An encoding of Interval Temporal Logic in Isabelle/HOL

Antonio Cau Ben Moszkowski David Smallwood

June 11, 2019

Abstract

These Isabelle theories introduce the semantics and syntax of Interval Temporal Logic (ITL). The ITL proof system, as introduced in [3], has been encoded and its soundness has been checked. An extensive library of ITL theorems, taken from [5], has been checked. The time reversal operator [4] has been defined and a collection of theorems has been provided.

Furthermore the new ITL operators first ▷ and last ◁ (introduced using time reversal) have been defined together with an extensive library of theorems. These were used to introduce the Runtime Verification monitor language RV [6] together with the algebraic properties of this language.

We also provide an algebraic characterisation of ITL based on [2] and link it with the work on Kleene Algebras [1].

Contents

theory *Interval*
 imports
  Main
begin

1 Intervals

An interval is a sequence of elements of a particular type. Intervals are similar to list in Isabelle/HOL, the difference is that intervals have instead of nil a single element at the end. So an interval of length zero is a single element whereas for Isabelle's list we have that an empty list is nil (no element present).

The usual operations on intervals are defined: length (*intlen*), *prefix*, *suffix*, *sub*, *nth*, *intfirst*, *intlast*, *intapp* and *intrev*.

In order to define the semantics of the ITL chopstar we introduce *index-sequence* which is a sequence of chop (fuse) points. This sequence is again of type interval but the elements are natural numbers. Two functions *shift* and *shiftm* are introduced that are used to add (shift) and subtract a natural number of each element in the sequence of chop (fuse) points.

1.1 Definitions

```
datatype 'a interval =
    St 'a (⌈-⌉)
  | Cons 'a 'a interval (infixr ⊙ 65)
for
  map: map
  rel: interval-all2
  pred: interval-all
```

```
type-synonym index = nat interval
```

```
syntax
  — interval Enumeration
  -interval :: args => 'a interval    (⟨(-)⟩)
```

```
translations
  ⟨x, xs⟩ == x⊙⟨xs⟩
  ⟨x⟩ == ⌈x⌉
```

```
primrec (nonexhaustive) intlen :: 'a interval ⇒ nat where
     intlen (St x) = 0
  |  intlen (x⊙xs) = 1+ (intlen xs)
```

```
primrec (nonexhaustive) nth :: 'a interval => nat => 'a where
    nth (St x) n       = x
  | nth (Cons x xs) n = (case n of 0 ⇒ x | Suc k ⇒ nth xs k)
```

```
primrec prefix:: nat ⇒ 'a interval ⇒ 'a interval where
    prefix n (St x) = (St x)
  | prefix n (Cons x xs) = (case n of 0 ⇒ (St x) | Suc m ⇒ (Cons x  (prefix m xs)))
```

```
primrec suffix:: nat ⇒ 'a interval ⇒ 'a interval where
   suffix n (St x) = (St x)
 | suffix n (Cons x  xs) = (case n of 0 ⇒ (Cons x  xs)  | Suc m ⇒ suffix m xs)
```

```
definition sub:: nat ⇒ nat ⇒ 'a interval ⇒ 'a interval
where
 sub n k xs = (if k<n then prefix 0 (suffix n xs)
                     else prefix (k−n) (suffix n xs)
              )
```

```
primrec intfirst :: 'a interval ⇒ 'a where
    intfirst (St x)     = x
  | intfirst (Cons x -) = x
```

primrec *intlast* :: *'a interval* ⇒ *'a* **where**
  intlast (*St x*) = *x*
| *intlast* (*Cons - xs*) = *intlast xs*

primrec *intapp* :: *'a interval* ⇒ *'a interval* ⇒ *'a interval* (**infixr** ⊖ *65*) **where**
intapp-St: (*St x*) ⊖ *ys* = *x* ⊙ *ys* |
intapp-Cons: (*x*⊙*xs*) ⊖ *ys* = *x* ⊙ (*xs* ⊖ *ys*)

primrec *intrev* :: *'a interval* ⇒ *'a interval* **where**
  intrev (*St x*) = (*St x*)
| *intrev* (*Cons x xs*) = (*intrev xs*) ⊖ (*St x*)

definition *index-sequence* :: *nat* ⇒ *index* ⇒ *bool* **where**
  index-sequence x idx ≡ (*nth idx 0* = *x*) ∧ (∀ *n*. *n*<*intlen idx* ⟶ *nth idx n* < *nth idx* (*Suc n*))

definition *shift* :: *nat* ⇒ *nat* ⇒ *nat* **where**
  shift k = (λ *x*. *x*+*k*)

definition *shiftm* :: *nat* ⇒ *nat* ⇒ *nat* **where**
  shiftm k = (λ *x*. (*if k*>*x then 0 else* (*x*−*k*)))

1.2 Lemmas

Basic lemmas are introduced for each of the above operations on intervals.

1.2.1 Interval Length

lemma *interval-intlen-gr-zero* [*simp*]:
  intlen xs ≥ *0*
by *auto*

lemma *interval-intlen-st* :
  intlen (*St x*) = *0*
by *simp*

lemma *interval-intlen-cons* [*simp*]:
  (*intlen* (*x*⊙*xs*)) = (*intlen xs*) +*1*
by *simp*

lemma *interval-intlen-cons-1* :
  intlen l > *0* ⟷ (∃ *x ls*. *l* = *x* ⊙*ls*)
by (*induct l*) *simp-all*

lemma *interval-intlen-map*:
  intlen (*map f xs*) = *intlen xs*
by (*induct xs*) *simp-all*

1.2.2 nth

lemma *interval-nth-zero* [*simp*]:
nth (*x*⊙*xs*) *0* = *x*
by *simp*

lemma *interval-nth-Suc* [*simp*]:
nth (*x*⊙*xs*) (*Suc n*) = *nth xs n*
by *auto*

lemma *interval-nth-last*:
nth (*x*⊙*xs*) (*intlen* (*x*⊙*xs*)) = *nth xs* (*intlen xs*)
by *simp*

lemma *interval-nth-cons*:
assumes *0*<*i* ∧ *i*<*1*+*intlen*(*xs*)
shows *nth*(*x*⊙*xs*) *i* =*nth xs* (*i*−*1*) ∧
nth(*x*⊙*xs*) (*i*+*1*) = *nth xs* ((*i*−*1*)+*1*)
by (*metis One-nat-def Suc-leI add.commute assms interval-nth-Suc le-add-diff-inverse2 plus-1-eq-Suc*)

lemma *interval-nth-zero-intfirst*:
nth xs 0 = *intfirst xs*
by (*induct xs*) *simp-all*

lemma *interval-nth-intlen-intlast*:
nth xs (*intlen xs*) = *intlast xs*
by (*induct xs*) *simp-all*

lemma *interval-st-intlen* :
(*xs* = (*St x*)) ⟷ *intlen xs* = *0* ∧ *nth xs 0* = *x*
by (*induct xs*) *simp-all*

lemma *interval-eq-nth-eq* :
(*xs* =*ys*) = (*intlen xs* =*intlen ys* ∧ (∀ *i*≤ *intlen xs*. *nth xs i* = *nth ys i*))
apply (*induct xs arbitrary*: *ys*)
apply (*metis interval-st-intlen le-numeral-extra*(*3*))
apply (*case-tac ys*, *simp*)
by *fastforce*

lemma *interval-nth-map* :
nth (*map f xs*) *i* = *f* (*nth xs i*)
apply (*induct xs arbitrary*: *i*, *simp*)
apply (*case-tac i*, *simp*, *simp*)
done

1.2.3 index sequence

lemma *interval-idx-less*:
assumes *iseq*: *index-sequence x idx*
shows (*n*<*intlen idx* ∧ *n*+*k*<*intlen idx*) ⟶ *nth idx n* < *nth idx* (*Suc*(*n*+*k*))
apply (*induct k*)

```
using index-sequence-def iseq apply auto[1]
using index-sequence-def iseq by auto

lemma interval-idx-less-last :
 assumes  index-sequence x idx
 shows    (i<intlen idx ∧ i+(intlen idx − (i+1))< intlen idx)
          ⟶ nth idx i < nth idx (Suc(i+(intlen idx −(i+1))))
using assms interval-idx-less by blast

lemma interval-idx-less-last-1:
 assumes  index-sequence x idx
 shows    i<intlen idx ⟶ nth idx i < nth idx (intlen idx)
using assms interval-idx-less-last by auto

lemma interval-idx-greater-first:
 assumes   index-sequence x idx
 shows     (i>0 ∧ i≤ intlen idx) ⟶ x < nth idx i
apply (induct i, simp)
using assms
by (metis One-nat-def Suc-le-lessD add-Suc index-sequence-def interval-idx-less
        less-le-trans plus-1-eq-Suc)

lemma interval-idx-cons:
    index-sequence 0 (x⊙ls) =
      (x=0 ∧ x<nth ls 0 ∧ index-sequence (nth ls 0) ls)
apply (simp add: index-sequence-def)
using less-Suc-eq-0-disj by auto

lemma interval-idx-shift-mono:
   mono (shift k)
by (simp add: Interval.shift-def mono-def)

lemma interval-idx-expand:
 index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧ 0≤i ∧ i< (intlen l)
  ⟹ 0≤ (nth l i) ∧ (nth l i) ≤  (nth l (i+1)) ∧ (nth l (i+1)) ≤ (intlen xs)
apply (simp add: index-sequence-def)
apply (induct l, simp)
by (metis Suc-lessI eq-imp-le index-sequence-def interval-idx-less-last-1 less-imp-le-nat)

lemma interval-idx-shift-idx [simp]:
    ( index-sequence (x+k) (map (shift k)  idx)) = (index-sequence x idx)
by (simp add: Interval.shift-def index-sequence-def interval-intlen-map interval-nth-map)

lemma interval-idx-shiftm :
   (index-sequence k (lsk) ∧ ls = map (shiftm k) lsk) ⟹
    index-sequence 0 (ls) ∧ (intlen ls) = (intlen lsk)
by (simp add: interval-eq-nth-eq index-sequence-def shift-def shiftm-def interval-nth-map )
  (smt Suc-leI diff-less-mono index-sequence-def interval-idx-greater-first interval-intlen-map
      le-less-trans less-Suc-eq-0-disj not-less order.asym)
```

lemma *interval-lsk-ls* :
(*index-sequence k* (*lsk*) ∧ *lsk* = *map* (*shift k*) *ls* ∧ *index-sequence 0* (*ls*)) =
(*index-sequence k* (*lsk*) ∧ *ls* = *map* (*shiftm k*) *lsk* ∧ *index-sequence 0* (*ls*))
apply (*simp add*: *interval-eq-nth-eq index-sequence-def shift-def shiftm-def interval-nth-map*)
apply *rule*
apply (*metis* (*no-types*, *lifting*) *add-diff-cancel-right′ interval-intlen-map not-add-less2*)
by (*metis* (*no-types*, *lifting*) *Suc-eq-plus1 add.commute add-cancel-right-left add-diff-inverse-nat ex-least-nat-less interval-intlen-map le-SucE le-zero-eq not-less-zero order-refl*)

lemma *interval-idx-link-shiftm*:
(*index-sequence k* (*lsk*) ∧ *ls* = *map* (*shiftm k*) *lsk*) =
(*index-sequence k* (*lsk*) ∧ *ls* = *map* (*shiftm k*) *lsk* ∧
index-sequence 0 (*ls*) ∧ (*intlen ls*) =(*intlen lsk*))
using *interval-idx-shiftm* **by** *blast*

lemma *interval-idx-link*:
(*lsk* = *map* (*shift k*) *ls* ∧ *index-sequence 0* (*ls*)) =
(*lsk* = *map* (*shift k*) *ls* ∧ *index-sequence k* (*lsk*) ∧ *index-sequence 0* (*ls*)∧
(*intlen ls*) =(*intlen lsk*))
by (*metis Interval.shift-def add-diff-cancel-left′ diff-diff-cancel diff-is-0-eq′ interval-idx-shift-idx interval-idx-shift-mono interval-intlen-map le-numeral-extra*(*3*) *mono-def*)

lemma *interval-idx-bound-0* :
assumes *index-sequence 0 ls* ∧ *Interval.nth ls* (*intlen ls*) = *intlen* (*suffix k xs*)
shows ((*i*≤*intlen ls*) ⟶ ((*nth ls* (*i*)) ≤ (*intlen* (*suffix k xs*))))
using *assms*
by (*metis add.commute add-eq-if eq-iff interval-idx-less le-add-diff-inverse2 le-neq-implies-less lessI less-imp-le-nat*)

lemma *interval-idx-bound-1*:
(*index-sequence 0* (*ls*) ∧ (*nth* (*ls*) (*intlen* (*ls*))) = (*intlen* (*suffix k xs*))) ⟷
(*index-sequence 0* (*ls*) ∧ (*nth* (*ls*) (*intlen* (*ls*))) = (*intlen* (*suffix k xs*)) ∧
(∀ *i*. (*i*≤*intlen ls*) ⟶ ((*nth ls* (*i*)) ≤ (*intlen* (*suffix k xs*)))))
using *interval-idx-bound-0* **by** *blast*

1.2.4 prefix, suffix and sub

lemma *interval-prefix-state* [*simp*]:
prefix m (*St x*) = (*St x*)
by *simp*

lemma *interval-prefix-suc* [*simp*]:
prefix (*Suc m*) (*x*⊙*xs*) = *x* ⊙ (*prefix m xs*)
by *auto*

lemma *interval-prefix-zero* [*simp*]:
prefix 0 (*x*⊙*xs*) = *St x*
by *auto*

lemma *interval-prefix-zero-intfirst* [*simp*]:

```
  prefix 0 xs = St (intfirst xs)
by (induct xs) simp-all

lemma interval-intfirst-prefix [simp]:
 i≤intlen xs ⟹  intfirst (prefix i xs) = intfirst xs
by (induct xs arbitrary: i, auto) (case-tac i, auto)

lemma interval-prefix-intlen [simp]:
    (prefix (intlen xs) xs) = xs
by (induct xs) simp-all

lemma interval-prefix-intlen-gr-1 [simp]:
   (prefix ((intlen xs)+i) xs) = xs
by (induct xs) simp-all

lemma interval-intlen-prefix-cons [simp]:
    intlen( prefix (Suc i) (x⊙xs)) = 1 + intlen(prefix i xs)
using interval-intlen-cons by auto

lemma interval-prefix-length :
    intlen (prefix i xs) = (if i≤ intlen xs then i else intlen xs)
by (induct xs arbitrary: i, simp) (case-tac i, auto)

lemma interval-prefix-length-good [simp]:
  assumes  i≤ intlen xs
  shows   (intlen ( prefix i xs)) = i
using assms by (simp add: interval-prefix-length)

lemma interval-prefix-length-bad [simp] :
 assumes   i > intlen xs
 shows      intlen (prefix i xs) = intlen xs
using assms by (simp add: interval-prefix-length)

lemma interval-pref-intlen-bound :
 assumes    i ≤ (intlen xs)
 shows      intlen (prefix i xs) ≤ intlen xs
using assms by (induct xs, simp) (metis interval-prefix-length)

lemma interval-suffix-length:
    intlen (suffix i xs ) = (if i≤ intlen xs then (intlen xs)−i else 0)
by (induct xs arbitrary: i, simp) (case-tac i, auto)

lemma interval-suffix-length-good [simp]:
  assumes  i≤ intlen xs
  shows    intlen (suffix i xs ) = (intlen xs)−i
using assms by (simp add: interval-suffix-length)

lemma interval-suffix-length-bad [simp]:
 assumes   i> intlen xs
 shows     intlen (suffix i xs ) = 0
```

```
using assms by (simp add: interval-suffix-length)

lemma interval-nth-prefix [simp]:
  i≤intlen xs ∧ k ≤ i ⟹ nth (prefix i xs) k = nth xs k
apply (induct xs arbitrary: i k, auto)
apply (case-tac i, auto)
apply (case-tac k, auto)
done

lemma interval-nth-suffix [simp]:
  i≤intlen xs ∧ k ≤ intlen xs −i ⟹ nth (suffix i xs) k = nth xs (i+k)
by (induct xs arbitrary: i k, auto) (case-tac i, auto)

lemma interval-suffix-prefix-help-1:
 assumes  ia+i ≤ intlen xs ∧ k ≤ ia
 shows  nth (prefix ia (suffix i xs)) k = nth (suffix i (prefix (ia +i) xs)) k
proof −
 have 1:  nth (prefix ia (suffix i xs)) k = nth (suffix i xs) k
 using interval-nth-prefix assms by (metis interval-prefix-intlen-gr-1 le-cases le-iff-add)
 have 2:  nth (suffix i xs) k = nth xs (i+k)
 using interval-nth-suffix assms by (simp add: add-le-imp-le-diff)
 have 3:  nth xs (i+k) = nth (prefix (ia+i) xs) (i+k)
 using interval-nth-prefix assms by simp
 have 4:  nth (prefix (ia+i) xs) (i+k) = nth (suffix i (prefix (ia +i) xs)) k
 using interval-nth-suffix assms by simp
 from 1 2 3 4 show ?thesis by auto
qed

lemma interval-suffix-prefix-help-2:
 assumes ia+i ≤ intlen xs
 shows    (∀ k ≤ ia . nth (prefix ia (suffix i xs)) k = nth (suffix i (prefix (ia +i) xs)) k)
using interval-suffix-prefix-help-1 using assms by fastforce

lemma interval-suffix-prefix-help-3:
 assumes ia+i ≤ intlen xs
 shows   intlen (prefix ia (suffix i xs)) = intlen (suffix i (prefix (ia +i) xs))
using assms interval-prefix-length-good interval-suffix-length-good by auto

lemma interval-suffix-prefix-swap:
 assumes ia+i ≤ intlen xs
 shows    prefix ia (suffix i xs) = suffix i (prefix (ia +i) xs)
by (simp add: interval-eq-nth-eq interval-suffix-prefix-help-2 interval-suffix-prefix-help-3 assms)

lemma interval-prefix-prefix-zero [simp]:
     prefix 0 ( prefix 0 xs ) = prefix 0 xs
by (induct xs) simp-all

lemma interval-pref-pref [simp]:
     (prefix i (prefix i xs)) = prefix i xs
by (metis interval-prefix-intlen interval-prefix-intlen-gr-1 interval-prefix-length
         less-imp-add-positive not-less)

lemma interval-pref-pref-3 [simp]:
    (prefix i (prefix (i+k) xs)) = prefix i xs
apply (induct xs arbitrary: i k, simp)
apply (case-tac i, auto)
by (simp add: Nitpick.case-nat-unfold)

lemma interval-pref-help:
 assumes i≤intlen (prefix (intlen xs − Suc 0) xs)
 shows    (prefix i (prefix (intlen xs − Suc 0) xs)) = (prefix i xs)
using assms
by (metis diff-le-self interval-pref-pref-3 interval-prefix-length
          ordered-cancel-comm-monoid-diff-class.add-diff-inverse)

lemma interval-pref-pref-help:
 assumes intlen xs >0 ∧ ia<intlen (xs)
 shows   (prefix ia (prefix (intlen xs − Suc 0) xs)) = (prefix ia  xs)
using assms
by (metis Suc-leI Suc-le-mono Suc-pred diff-le-self interval-pref-help interval-prefix-length-good)

lemma interval-pref-pref-help-1:
 assumes i>0 ∧ i≤ intlen xs
 shows   (prefix (intlen (prefix i xs) − Suc 0) (prefix i xs)) =
        (prefix (intlen (prefix i xs) − Suc 0) xs)
using assms interval-pref-pref-3 by (metis diff-le-self interval-prefix-length-good le-iff-add)

lemma interval-suffix-suc [simp]:
    suffix (Suc m) (x ⊙ xs) = suffix m xs
by auto

lemma interval-suffix-zero [simp]:
    suffix 0 xs = xs
by (induct xs) simp-all

lemma interval-suffix-intlen [simp]:
    suffix (intlen xs) xs = (St (nth xs (intlen xs)))
by (induct xs) simp-all

lemma interval-suffix-intlast [simp]:
   suffix (intlen xs) xs = St (intlast xs)
by (induct xs) simp-all

lemma interval-suffix-suffix [simp]:
    suffix i (suffix j xs) = suffix (i+j) xs
apply (induct xs arbitrary: i j, simp)
apply (case-tac i, auto)
by (simp add: Nitpick.case-nat-unfold)

lemma interval-prefix-suffix-intlen:
```

intlen (*prefix ia* (*suffix i xs*)) =
(*if i* ≤ *intlen xs then*
(*if ia*≤ *intlen xs* −*i* *then ia else* (*intlen xs*) −*i*)
else 0)
by (*metis interval-prefix-length interval-suffix-length le-zero-eq*)

lemma *interval-prefix-suffix-intlen-good* [*simp*]:
assumes *ia*≤ *intlen xs* −*i* ∧ *i* ≤ *intlen xs*
shows *intlen* (*prefix ia* (*suffix i xs*)) = *ia*
using *assms* **by** (*simp add*: *interval-prefix-suffix-intlen*)

lemma *interval-prefix-suffix-intlen-bad-0* [*simp*]:
assumes *i*> *intlen xs*
shows *intlen* (*prefix ia* (*suffix i xs*)) = *0*
using *assms* **by** (*simp add*: *interval-prefix-suffix-intlen*)

lemma *interval-prefix-suffix-intlen-bad-1* [*simp*] :
assumes *i* ≤ *intlen xs* ∧ *ia* > *intlen xs* −*i*
shows *intlen* (*prefix ia* (*suffix i xs*)) = (*intlen xs*) −*i*
using *assms* **by** (*simp add*: *interval-prefix-suffix-intlen*)

lemma *interval-suffix-suffix-3*:
assumes *i*>*0* ∧ *ia*<*i* ∧ *i*≤ *intlen xs*
shows (*suffix* (*i*−*ia*) (*suffix* ((*intlen xs*)−*i*) *xs*)) = (*suffix* (((*intlen xs*)−*ia*)) *xs*)
using *assms* **by** *simp*

lemma *interval-sub-zero-prefix* :
sub 0 k xs = *prefix k xs*
by (*simp add*: *Interval.sub-def*)

lemma *interval-sub-suffix* :
assumes (*i* < *j* ∧ *j*≤ (*intlen xs*)−*k*)
shows (*sub* (*i*+*k*) (*j*+*k*) *xs*) = (*sub i j* (*suffix k xs*))
using *assms* **by** (*simp add*: *Interval.sub-def*)

lemma *interval-sub-prefix-suffix-0*:
assumes (*0* ≤ *i* ∧ *ia*+*i* ≤ *intlen xs*)
shows (*sub i* (*i*+*ia*) *xs*) = (*prefix* (*ia*) (*suffix i xs*))
using *assms* **by** (*simp add*: *Interval.sub-def*)

lemma *interval-sub-prefix-suffix*:
assumes *0* ≤ *i* ∧ *i*≤*j* ∧ *j* ≤ *intlen xs*
shows (*sub i j xs*) = (*prefix* (*j*−*i*) (*suffix i xs*))
using *assms* **by** (*simp add*: *Interval.sub-def*)

1.2.5 Reverse

lemma *interval-intlen-intapp* [*simp*]:
intlen (*xs* ⊖ *ys*) = (*intlen xs*) + (*intlen ys*) +*1*
by (*induct xs arbitrary*: *ys*) *simp-all*

```
lemma interval-intrev-intlen [simp]:
  intlen (intrev xs) = intlen xs
by (induct xs, simp, simp)

lemma interval-suffix-intapp [simp]:
 suffix (Suc (intlen xs)) (xs ⊖ ys) = ys
by (induct xs) simp-all

lemma interval-suffix-intapp2 [simp]:
 suffix (intlen xs − k) ( xs) ⊖ ys = suffix (intlen xs − k) ( xs ⊖ ys)
by (induct xs, simp)
   (metis Suc-diff-le diff-is-0-eq′ intapp-Cons interval-suffix-suc interval-suffix-zero
          intlen.simps(2) not-less-eq-eq plus-1-eq-Suc)

lemma interval-intapp-assoc [simp]:
  (xs ⊖ ys) ⊖ zs = xs ⊖ ( ys ⊖ zs )
by (induct xs) simp-all

lemma interval-intapp-nth:
  nth (xs ⊖ ys) k = (if k ≤ intlen xs
                     then (nth xs k)
                     else (nth ys (k − (intlen xs) −1)) )
apply (induct xs arbitrary: k)
apply (case-tac k, simp, simp)
apply (case-tac k, simp, simp)
done

lemma interval-rev-intapp [simp]:
  intrev (xs ⊖ ys) = (intrev ys) ⊖ (intrev xs)
by (induct xs) simp-all

lemma interval-rev-rev-ident [simp]:
  intrev (intrev xs) = xs
by (induct xs) auto

lemma interval-rev-swap :
  ((intrev xs) = ys) = (xs = intrev ys)
by auto

lemma interval-intlast-intrev:
 intlast (intrev xs) = intfirst xs
by (induct xs, auto)
   (metis Suc-eq-plus1 add.right-neutral interval.inject(1) interval-intlen-intapp
          interval-intlen-st interval-suffix-intapp interval-suffix-intlast)

lemma interval-intfirst-intrev:
 intfirst (intrev xs) = intlast xs
by (induct xs, auto)
   (metis intapp-St interval-intlast-intrev interval-rev-intapp intlast.simps(2) intrev.simps(1))
```

```
lemma interval-intrev-nth:
 k≤ intlen (intrev xs) ⟹ (nth  (intrev xs) k) = (nth  xs ((intlen xs) −k))
apply (induct xs, simp)
apply simp
apply (case-tac k)
apply (simp add: interval-intapp-nth)
by (smt Interval.nth.simps(1) Suc-diff-Suc diff-Suc-Suc diff-is-0-eq' interval-intapp-nth
       interval-intrev-intlen le-SucE less-Suc-eq-le old.nat.simps(4) old.nat.simps(5))
```

```
lemma interval-intrev-prefix:
 k≤ intlen xs ⟹ intrev( prefix k xs) = suffix ((intlen xs) − k) (intrev xs)
apply (induct xs arbitrary: k, simp)
apply simp
apply (case-tac k)
apply (metis diff-zero interval-intrev-intlen interval-suffix-intapp intrev.simps(1) old.nat.simps(4))
by (metis Suc-le-mono diff-Suc-Suc interval-intrev-intlen interval-suffix-intapp2
        intrev.simps(2) old.nat.simps(5))
```

```
lemma interval-intrev-suffix:
 k≤ intlen xs ⟹ intrev( suffix k xs) = prefix ((intlen xs) − k) (intrev xs)
by (induct xs arbitrary: k, simp, simp add: interval-intrev-prefix interval-rev-swap)
```

```
lemma interval-intrev-sub1:
 assumes  0 ≤ i ∧ i≤j ∧ j ≤ intlen xs
 shows    intrev (sub i j xs) = intrev (prefix (j−i) (suffix i xs))
using assms interval-sub-prefix-suffix by (simp add: interval-sub-prefix-suffix)
```

```
lemma interval-intrev-sub2:
 assumes  0 ≤ i ∧ i≤j ∧ j ≤ intlen xs
 shows    intrev (prefix (j−i) (suffix i xs)) = suffix ((intlen xs) − j) (intrev (suffix i xs))
using assms interval-intrev-prefix[of j−i suffix i xs]  by auto
```

```
lemma interval-intrev-sub3:
 assumes  0 ≤ i ∧ i≤j ∧ j ≤ intlen xs
 shows    suffix ((intlen xs) − j) (intrev (suffix i xs)) =
          suffix ((intlen xs) −j) (prefix ((intlen xs) − i) (intrev xs))
using assms interval-intrev-suffix[of i xs] by auto
```

```
lemma interval-intrev-sub4:
 assumes  0 ≤ i ∧ i≤j ∧ j ≤ intlen xs
 shows   suffix ((intlen xs) −j) (prefix ((intlen xs) − i) (intrev xs)) =
         sub ((intlen xs) −j) ((intlen xs) − i) (intrev xs)
using assms by (simp add: diff-le-mono2 interval-sub-prefix-suffix interval-suffix-prefix-swap)
```

```
lemma interval-intrev-sub:
 assumes  0 ≤ i ∧ i≤j ∧ j ≤ intlen xs
 shows    intrev (sub i j xs) = sub ((intlen xs) −j) ((intlen xs) − i) (intrev xs)
using assms
by (simp add: interval-intrev-sub1 interval-intrev-sub2 interval-intrev-sub3 interval-intrev-sub4)
```

```
lemma interval-intrev-idx-2:
 assumes index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧
          0≤i ∧ i< (intlen l)
 shows   (intrev (sub (nth l i) (nth l (i+1)) xs)) =
   ( (sub ((intlen xs)− (nth l (i+1))) ((intlen xs) − (nth l i)) (intrev xs)))
using assms interval-idx-expand interval-intrev-sub[of (nth l i)  (nth l (i+1))  xs]
by blast

lemma interval-intrev-idx-3:
 assumes  index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧
           ls = map (λ x. (intlen xs) −x) (intrev l)
 shows   (nth ls 0) = 0 ∧ (nth ls (intlen ls)) = (intlen xs) ∧ intlen ls =intlen l
using assms
by (metis diff-self-eq-0 diff-zero index-sequence-def interval-intfirst-intrev
          interval-intlast-intrev interval-intlen-map interval-intrev-intlen
          interval-nth-intlen-intlast interval-nth-map interval-nth-zero-intfirst)

lemma interval-intrev-idx-4:
  index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧
             ls = map (λ x. (intlen xs) −x) (intrev l)
   ⟹ i≤ intlen ls ⟶ (nth ls i) = (intlen xs) − (nth l ((intlen l)−i))
apply (induct ls)
apply (metis diff-zero interval-intlen-st interval-intrev-idx-3 le-0-eq)
by (simp add: interval-intlen-map interval-intrev-nth interval-nth-map)

lemma interval-intrev-idx-5:
 assumes  (index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs))
 shows    (i< intlen l ⟶
           (intlen xs) − (nth l ((intlen l)−i)) < (intlen xs) − (nth l ((intlen l)−(i+1))))
using assms
by (smt Suc-diff-Suc Suc-eq-plus1 add-gr-0 add-less-cancel-left diff-less
        index-sequence-def le-add-diff-inverse2 le-numeral-extra(3) less-diff-conv
        less-imp-le-nat not-gr-zero interval-idx-expand)

lemma interval-intrev-idx-6:
 assumes  (index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧
           ls = map (λ x. (intlen xs) −x) (intrev l))
 shows    (i< intlen ls ⟶
           ((nth ls i) = (intlen xs) − (nth l ((intlen l)−i)) ∧
           (nth ls (i+1)) = (intlen xs) − (nth l ((intlen l)−(i+1))) ∧
           (nth ls i) < (nth ls (i+1))))
proof −
 have 1: (i< intlen ls ⟶ (nth ls i) = (intlen xs) − (nth l ((intlen l)−i)))
          using assms interval-intrev-idx-4 less-imp-le-nat by blast
 have 2: (i< intlen ls ⟶ (nth ls (i+1)) = (intlen xs) − (nth l ((intlen l)−(i+1))))
          using assms by (simp add: interval-intrev-idx-4)
 have 3: ( i< intlen ls ⟶
           ((nth ls i) = (intlen xs) − (nth l ((intlen l)−i)) ∧
           (nth ls (i+1)) = (intlen xs) − (nth l ((intlen l)−(i+1)))))
        using 1 2 by auto
  have 4: (i< intlen ls ⟶
            ((nth ls i) = (intlen xs) − (nth l ((intlen l)−i)) ∧
            (nth ls (i+1)) = (intlen xs) − (nth l ((intlen l)−(i+1))) ∧
            (nth ls i) < (nth ls (i+1))))
         using assms 3 index-sequence-def interval-intrev-idx-5
        by (metis interval-intlen-map interval-intrev-intlen)
  from 4 show ?thesis by blast
qed

lemma interval-intrev-idx-7:
  assumes  (index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧
           ls = map (λ x. (intlen xs) −x) (intrev l))
    shows  index-sequence 0 ls
using assms interval-intrev-idx-6 interval-intrev-idx-3
by (metis Suc-eq-plus1 index-sequence-def )

lemma interval-intrev-idx-8:
  assumes index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧
          ls = map (λ x. (intlen xs) −x) (intrev l) ∧ index-sequence 0 ls
  shows   i<intlen ls ⟶
          (intlen xs)− (nth l (i+1)) = nth ls ((intlen ls)−(i+1)) ∧
          (intlen xs) − (nth l i)    = (nth ls ((intlen ls) −i))
using assms interval-intrev-idx-4
by (smt Suc-eq-plus1 Suc-leI add-diff-cancel-right′ assms diff-diff-cancel diff-diff-left
      diff-le-self interval-intrev-idx-3)

lemma interval-intrev-idx-9:
  assumes index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs) ∧
          ls = map (λ x. (intlen xs) −x) (intrev l) ∧ index-sequence 0 ls
  shows   i<intlen ls ⟶
          sub ((intlen xs)− (nth l (i+1))) ((intlen xs) − (nth l i)) (intrev xs) =
          sub (nth ls ((intlen ls)−(i+1))) ((nth ls ((intlen ls) −i)) ) (intrev xs)

using interval-intrev-idx-8 using assms by fastforce

lemma interval-intrev-idx-11:
  assumes (index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs))
  shows    i≤intlen l ⟶
         (nth l i) = (nth (map (λ x.(intlen xs)−x) (intrev (map (λ x.(intlen xs)−x) (intrev l)))) i)
using assms index-sequence-def
by (smt diff-diff-cancel diff-is-0-eq diff-less diff-zero leD le-cases not-gr-zero
      interval-intrev-idx-3 interval-intrev-idx-6 interval-intrev-idx-7)

lemma interval-intrev-idx-12:
  assumes (index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen xs))
  shows    l = map (λ x. (intlen xs) −x  ) (intrev (map (λ x. (intlen xs) −x) (intrev l)))
using assms interval-intrev-idx-11
by (simp add: interval-intrev-idx-11 interval-eq-nth-eq interval-intlen-map)
```

end

theory *Syntax*
imports
Main
abbrevs &-*i* = $\wedge_i$ **and**
|-*i* = $\vee_i$ **and**
)-*i* = $\supset_i$ **and**
−-*i* = $\neg_i$ **and**
=-*i* = $\equiv_i$ **and**
false-i = $false_i$ **and**
true-i = $true_i$ **and**
if-i = if_i **and**
-∗ = *
begin

2 Syntax

The basic ITL syntax is introduced first followed by the derived operators including the time reversal operator ([4]).

2.1 Primitive formulae

datatype *'a pitl* =
false-d ($false_i$)
| *atom-d 'a* ⇒ *bool* (($atom_i$ -))
| *implies-d 'a pitl 'a pitl* ((- $\supset_i$ -) [*26*,*25*] *25*)
| *skip-d* (*skip*)
| *chop-d 'a pitl 'a pitl* ((- ; -) [*84*,*84*] *83*)
| *chopstar-d 'a pitl* ((- *) [*85*] *85*)

2.2 Derived Boolean Operators

definition *not-d* (($\neg_i$ -) [*90*] *90*)
where
$\neg_i\ f \equiv f \supset_i false_i$

definition *true-d* ($true_i$)
where
$true_i \equiv \neg_i\ false_i$

definition *or-d* ((- $\vee_i$ -) [*31*,*30*] *30*)
where
$f \vee_i g \equiv \neg_i\ f \supset_i g$

definition *and-d* ((- $\wedge_i$ -) [*36*,*35*] *35*)
where

$f \wedge_i g \equiv \neg_i (\neg_i f \vee_i \neg_i g)$

definition *iff-d* ((- $\equiv_i$ -) [*21*,*20*] *20*)
where
$f \equiv_i g \equiv ((f \supset_i g) \wedge_i (g \supset_i f))$

2.3 Next and Previous Operators

definition *next-d* (($\bigcirc$ -) [*88*] *87*)
where
$\bigcirc f \equiv skip ; f$

definition *wnext-d* ((*wnext* -) [*88*] *87*)
where
$wnext\ f \equiv \neg_i (\bigcirc (\neg_i f))$

definition *prev-d* ((*prev* -) [*88*] *87*)
where
$prev\ f \equiv f;skip$

definition *wprev-d* ((*wprev* -) [*88*] *87*)
where
$wprev\ f \equiv \neg_i(prev(\neg_i f))$

2.4 More and Empty

definition *more-d* (*more*)
where
$more \equiv \bigcirc true_i$

definition *empty-d* (*empty*)
where
$empty \equiv \neg_i more$

2.5 Box and Diamond Operators

definition *sometimes-d* (($\Diamond$ -) [*88*] *87*)
where
$\Diamond f \equiv true_i ; f$

definition *always-d* (($\Box$ -) [*88*] *87*)
where
$\Box f \equiv \neg_i (\Diamond (\neg_i f))$

definition *di-d* ((*di* -) [*88*] *87*)
where
$di\ f \equiv f ; true_i$

definition *bi-d* ((*bi* -) [*88*] *87*)
where

$\mathit{bi}\ f \equiv \neg_i\ (\mathit{di}\ (\neg_i\ f))$

definition *da-d* ((*da* -) [*88*] *87*)
where
$\mathit{da}\ f \equiv \mathit{true}_i\ ;\ (f\ ;\ \mathit{true}_i)$

definition *ba-d* ((*ba* -) [*88*] *87*)
where
$\mathit{ba}\ f \equiv \neg_i\ (\mathit{da}\ (\neg_i\ f))$

definition *dm-d* ((*dm* -) [*88*] *87*)
where
$\mathit{dm}\ f \equiv \diamond\ (\mathit{more} \wedge_i f)$

definition *bm-d* ((*bm* -) [*88*] *87*)
where
$\mathit{bm}\ f \equiv \neg_i\ (\mathit{dm}\ (\neg_i\ f))$

2.6 Initial and Final Operators

definition *init-d* ((*init* -) [*88*] *87*)
where
$\mathit{init}\ f \equiv ((\mathit{empty} \wedge_i f);\ \mathit{true}_i)$

definition *fin-d* ((*fin* -) [*88*] *87*)
where
$\mathit{fin}\ f \equiv \Box\ (\ \mathit{empty} \supset_i f)$

2.7 Programming Constructs

definition *halt-d* ((*halt* -) [*88*] *87*)
where
$\mathit{halt}\ f \equiv \Box\ (\ \mathit{empty} \equiv_i f)$

definition *keep-d* ((*keep* -) [*88*] *87*)
where
$\mathit{keep}\ f \equiv \mathit{ba}\ (\ \mathit{skip} \supset_i f)$

definition *yields-d* ((- *yields* -) [*88*,*88*] *87*)
where
$f\ \mathit{yields}\ g \equiv \neg_i\ (\ f;\ \neg_i\ g)$

definition *ifthenelse-d* ((*if*$_i$ - *then* - *else* -) [*88*,*88*,*88*] *87*)
where
$\mathit{if}_i\ f\ \mathit{then}\ g\ \mathit{else}\ h \equiv (f \wedge_i g) \vee_i (\ (\neg_i\ f) \wedge_i h)$

definition *ifthen-d* ((*if*$_i$ - *then* -) [*88*,*88*] *87*)
where
$\mathit{if}_i\ f\ \mathit{then}\ g\ \equiv \mathit{if}_i\ f\ \mathit{then}\ g\ \mathit{else}\ \mathit{true}_i$

definition *while-d* ((*while* - *do* -) [*88*,*88*] *87*)
where
while f do g $\equiv$ (*f* $\wedge_i$ *g*)* $\wedge_i$ *fin* ($\neg_i$ *f*)

definition *repeat-d* ((*repeat* - *until* -) [*88*,*88*] *87*)
where
repeat f until g $\equiv$ *f* ; (*while* ($\neg_i$ *g*) *do f*)

primrec *len-d* :: *nat* $\Rightarrow$ *'a pitl* ((*len* -) [*88*] *87*) **where**
(*len 0*) = *empty*
| (*len* (*Suc n*)) = (*skip* ; (*len n*))

primrec *power-chop-d* :: *'a pitl* $\Rightarrow$ *nat* $\Rightarrow$ *'a pitl* **where**
power-0 : *power-chop-d f 0* = *empty*
| *power-Suc*: *power-chop-d f* (*Suc n*) = ((*f* $\wedge_i$ *more*);(*power-chop-d f n*))

primrec *power-d* :: *'a pitl* $\Rightarrow$ *nat* $\Rightarrow$ *'a pitl* ((*power* - -) [*88*,*88*] *87*) **where**
pow-0 : *power-d f 0* = *empty*
| *pow-Suc*: *power-d f* (*Suc n*) = ((*f*);(*power-d f n*))

2.8 Time reversal

primrec *rev-d* :: *'a pitl* $\Rightarrow$ *'a pitl* ((- ^{r}) [*88*] *87*)
where
$\mathit{false}_i{}^r = \mathit{false}_i$
| $(\mathit{atom}_i\ p)^r = \mathit{fin}\ (\mathit{atom}_i\ p)$
| $(f1 \supset_i f2)^r = (\ f1^r \supset_i f2^r\ )$
| $\mathit{skip}^r = \mathit{skip}$
| $(f1\ ;\ f2)^r = (\ f2^r\ ;\ f1^r\ )$
| $(f^\star)^r = (f^r)^\star$

end

theory *Semantics*
imports
Interval
Syntax
begin

3 Semantics

The semantics of the basic ITL operators is defined. Then lemmas are introduced that define the derived ITL operators in terms of operations on intervals. Furthermore we prove the soundness of the ITL axioms. This is followed by the key time reversal lemma.

3.1 Semantics Primitive Operators

fun *semantics-pitl* :: [*'a interval*, *'a pitl*] $\Rightarrow$ *bool* ((- $\models$ -) [*80*,*10*] *10*)

```
where
  (σ ⊨ falseᵢ) = False
| (σ ⊨ atomᵢ p) = (p (intfirst σ))
| (σ ⊨ f ⊃ᵢ g) = ((σ ⊨ f) ⟶ (σ ⊨ g))
| (σ ⊨ skip)  = (intlen σ = 1)
| (σ ⊨ f ; g) =
   (∃ i. ((0 ≤ i) ∧ (i ≤ (intlen σ)) ∧ ((prefix i σ) ⊨ f) ∧ ((suffix i σ) ⊨ g)) )
| (σ ⊨ f⋆) = (∃ (l::index). index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen σ) ∧
                         (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                             ((sub (nth l i) (nth l (i+1)) σ) ⊨ f)
                         )
                        )
```

3.2 Semantics Boolean Operators

```
lemma not-defs [simp]:
  (σ ⊨ ¬ᵢ f ) = Not ( σ ⊨ f)
by (simp add: not-d-def)

lemma or-defs [simp]:
  (σ ⊨ f1 ∨ᵢ f2) = ( (σ ⊨ f1) ∨ (σ ⊨ f2))
by (metis not-defs or-d-def semantics-pitl.simps(3))

lemma and-defs [simp]:
  (σ ⊨ f1 ∧ᵢ f2) = ( (σ ⊨ f1) ∧ (σ ⊨ f2))
by (simp add: and-d-def)

lemma implies-defs [simp]:
  (σ ⊨ f1 ⊃ᵢ f2) = ( (σ ⊨ f1) ⟶ (σ ⊨ f2))
by auto

lemma iff-defs [simp]:
  (σ ⊨ f1 ≡ᵢ f2) = ( (σ ⊨ f1) = (σ ⊨ f2))
by (metis and-defs iff-d-def semantics-pitl.simps(3))

lemma true-defs [simp]:
  (σ ⊨ trueᵢ)
by (simp add: true-d-def)
```

3.3 Semantics Box and Diamond Operators

```
lemma sometimes-defs [simp]:
  (σ ⊨ ◇ f) = (∃ i ≤ intlen σ.  (suffix i σ ⊨ f))
by (simp add: sometimes-d-def)

lemma always-defs [simp]:
  (σ ⊨ □ f) = (∀ i ≤ intlen σ.  (suffix i σ ⊨ f))
by (simp add: always-d-def)

lemma di-defs [simp]:
```

$(\sigma \models \textit{di } f) = (\exists\ i \le \textit{intlen } \sigma.\ \ (\textit{prefix } i\ \sigma \models f))$
by (*simp add*: *di-d-def*)

lemma *bi-defs* [*simp*]:
$(\sigma \models \textit{bi } f) = (\forall\ i \le \textit{intlen } \sigma.\ (\textit{prefix } i\ \sigma \models f))$
by (*simp add*: *bi-d-def*)

lemma *da-defs* [*simp*]:
$(\sigma \models \textit{da } f) = (\exists\ i\ \textit{ia}\ .\ 0 \le i\ \wedge \textit{ia}+i \le \textit{intlen } \sigma \wedge (\textit{sub } i\ (\textit{ia} + i)\ \sigma \models f))$
apply (*simp add*: *da-d-def*)
using *interval-prefix-length-good interval-suffix-length-good*
by (*smt add.commute add-diff-cancel-left′ add-leD2 interval-sub-prefix-suffix-0 le-iff-add nat-add-left-cancel-le zero-le*)

lemma *ba-defs* [*simp*]:
$(\sigma \models \textit{ba } f) = (\forall\ i\ \textit{ia}\ .\ (0 \le i \wedge \textit{ia}+i \le \textit{intlen } \sigma) \longrightarrow (\textit{sub } i\ (\textit{ia}+i)\ \sigma \models f))$
by (*metis ba-d-def da-defs not-defs*)

3.4 Semantics Next and Previous Operators

lemma *skip-defs* :
$(\sigma\ \models \textit{skip}) = (\ (\textit{intlen } \sigma) = 1)$
by *simp*

lemma *next-defs* [*simp*]:
$(\sigma \models \bigcirc\ f) = (\ (\textit{intlen } \sigma) > 0 \wedge ((\textit{suffix } 1\ \sigma) \models f))$
using *Suc-le-eq* **by** (*simp add*: *next-d-def*) *force*

lemma *wnext-defs* [*simp*]:
$(\sigma \models \textit{wnext } f) = ((\textit{intlen } \sigma) = 0 \vee ((\textit{suffix } 1\ \sigma) \models f))$
by (*simp add*: *wnext-d-def*)

lemma *prev-defs* [*simp*]:
$(\sigma \models \textit{prev } f) = (\ (\textit{intlen } \sigma) > 0 \wedge ((\textit{prefix } ((\textit{intlen } \sigma) - 1)\ \sigma) \models f))$
by (*simp add*: *prev-d-def*)
(*metis One-nat-def Suc-leI diff-diff-cancel diff-is-0-eq′ diff-le-self interval-suffix-length-good neq0-conv zero-neq-one*)

lemma *wprev-defs* [*simp*]:
$(\sigma \models \textit{wprev } f) = (\ (\textit{intlen } \sigma) = 0 \vee ((\textit{prefix } ((\textit{intlen } \sigma) - 1)\ \sigma) \models f))$
by (*simp add*: *wprev-d-def*)

3.5 Semantics More and Empty

lemma *more-defs* [*simp*] :
$(\sigma \models \textit{more}) = (\textit{intlen } \sigma > 0)$
by (*simp add*: *more-d-def*)

lemma *empty-defs* [*simp*]:
$(\sigma \models \textit{empty}) = (\ (\textit{intlen } \sigma) = 0)$

by (*simp add*: *empty-d-def*)

3.6 Semantics Initial and Final Operators

lemma *init-defs* [*simp*]:
$(\sigma \models init\ f) = (\ (St\ (intfirst\ \sigma)) \models f)$
by (*simp add*: *init-d-def*) *auto*

lemma *fin-defs* [*simp*]:
$(\sigma \models fin\ f) = (\ (St\ (intlast\ \sigma\ )) \models f)$
by (*simp add*: *fin-d-def interval-nth-intlen-intlast*)

3.7 Semantics Programming Constructs

lemma *ifthenelse-defs* [*simp*]:
$(\sigma \models if_i\ f0\ then\ f1\ else\ f2) =$
$(\ ((\sigma \models f0) \wedge (\sigma \models f1)) \vee (\ \neg(\sigma \models f0) \wedge (\sigma \models f2)))$
by (*simp add*: *ifthenelse-d-def*)

lemma *len-defs* [*simp*]:
$(\sigma \models len\ n) = ((intlen\ \sigma) = n)$
by (*induct n arbitrary*: σ, *simp*, *simp*) *fastforce*

3.8 Soundness Axioms

3.8.1 ChopAssoc

lemma *ChopAssocSemHelp*:
$(\exists\, i\ ia\ .\ i \le intlen\ \sigma \wedge ia \le intlen\ \sigma - i \wedge (prefix\ i\ \sigma \models f) \wedge$
$(prefix\ ia\ (suffix\ i\ \sigma) \models g) \wedge (suffix\ (ia + i)\ \sigma \models h)) =$
$(\exists\, j\ ja\ .\ j \le intlen\ \sigma \wedge ja \le j \wedge (prefix\ ja\ (prefix\ j\ \sigma) \models f) \wedge$
$(suffix\ ja\ (prefix\ j\ \sigma) \models g) \wedge (suffix\ j\ \sigma \models h))$
by (*smt Nat.le-diff-conv2 add-diff-cancel-left′ interval-pref-pref-3 interval-suffix-prefix-swap le-add1 le-add-diff-inverse2 le-trans*)

lemma *ChopAssocSemHelp2*:
$(\sigma \models\ f\ ;\ (g\ ;\ h)) = (\sigma \models (f;g);h)$
proof –
have $(\sigma \models\ f\ ;\ (g\ ;\ h)) =$
$((\exists\, i \le intlen\ \sigma.\ (prefix\ i\ \sigma \models f) \wedge (\exists\, ia \le intlen\ (suffix\ i\ \sigma).$
$(prefix\ ia\ (suffix\ i\ \sigma) \models g) \wedge (suffix\ (ia + i)\ \sigma \models h))))$
by *simp-all*
also have ... =
$(\exists\, i\ ia\ .\ i \le intlen\ \sigma \wedge ia \le intlen\ \sigma - i \wedge (prefix\ i\ \sigma \models f) \wedge$
$(prefix\ ia\ (suffix\ i\ \sigma) \models g) \wedge (suffix\ (ia + i)\ \sigma \models h))$
by *fastforce*
also have ... =
$(\exists\, j\ ja\ .\ j \le intlen\ \sigma \wedge ja \le j \wedge (prefix\ ja\ (prefix\ j\ \sigma) \models f) \wedge$
$(suffix\ ja\ (prefix\ j\ \sigma) \models g) \wedge (suffix\ j\ \sigma \models h))$
using *ChopAssocSemHelp*[*of* σ *f g h*] **by** *blast*
also have ... =

$(\exists\, i \le intlen\ \sigma.\ (\exists\, ia \le intlen\ (prefix\ i\ \sigma).\ (prefix\ ia\ (prefix\ i\ \sigma) \models f) \wedge$
$(suffix\ ia\ (prefix\ i\ \sigma) \models g)) \wedge (suffix\ i\ \sigma \models h))$
by *fastforce*
also have ... =
$(\sigma \models (f;g);h)$ **by** *simp-all*
finally show $(\sigma \models\ f\ ;\ (g\ ;\ h)) = (\sigma \models (f;g);h)$.
qed

lemma *ChopAssocSem*:
$(\sigma \models\ f\ ;\ (g\ ;\ h) \equiv_i\ (f;g);h)$
using *ChopAssocSemHelp2* **using** *iff-defs* **by** *blast*

3.8.2 OrChopImp

lemma *OrChopImpSem*:
$(\sigma \models\ (\ f \vee_i g);h\ \ \supset_i\ f;h \vee_i g;h\ )$
by *simp-all blast*

3.8.3 ChopOrImp

lemma *ChopOrImpSem*:
$(\sigma \models f;(g \vee_i h) \supset_i\ \ f;g \vee_i\ f;h\ )$
by *simp-all blast*

3.8.4 EmptyChop

lemma *EmptyChopSem*:
$(\sigma \models empty\ ;\ f \equiv_i f\ )$
by *simp-all auto*

3.8.5 ChopEmpty

lemma *ChopEmptySem*:
$(\sigma \models f;empty \equiv_i f\ )$
by *simp-all auto*

3.8.6 StateImpBi

lemma *StateImpBiSem*:
$(\sigma \models init\ f \supset_i\ \ bi\ (init\ f)\ )$
by *simp-all*

3.8.7 NextImpNotNextNot

lemma *NextImpNotNextNotSem*:
$(\sigma \models \bigcirc\ f \supset_i\ \neg_i\ (\bigcirc\ \neg_i\ f)\ )$
by *auto*

3.8.8 BiBoxChopImpChop

lemma *BiBoxChopImpChopSem*:
$(\sigma \models bi\ (\ f \supset_i f1) \wedge_i \Box(g \supset_i g1) \supset_i f;g \supset_i f1;g1\ )$

by *fastforce*

3.8.9 BoxInduct

lemma *box-induct-help-1* :
$(\sigma \models f) \wedge (\forall i.\ \mathit{Suc}\ 0 \leq \mathit{intlen}\ \sigma - i \longrightarrow$
$i \leq \mathit{intlen}\ \sigma \longrightarrow (\mathit{suffix}\ i\ \sigma \models f) \longrightarrow (\mathit{suffix}\ (\mathit{Suc}\ i)\ \sigma \models f))$
$\Longrightarrow (\forall\ j.\ j \leq \mathit{intlen}\ \sigma \longrightarrow (\mathit{suffix}\ j\ \sigma \models f))$
proof
fix *j*
show $(\sigma \models f) \wedge (\forall i.\ \mathit{Suc}\ 0 \leq \mathit{intlen}\ \sigma - i \longrightarrow$
$i \leq \mathit{intlen}\ \sigma \longrightarrow (\mathit{suffix}\ i\ \sigma \models f) \longrightarrow (\mathit{suffix}\ (\mathit{Suc}\ i)\ \sigma \models f))$
$\Longrightarrow\ j \leq \mathit{intlen}\ \sigma \longrightarrow (\mathit{suffix}\ j\ \sigma \models f)$
proof
(*induct j arbitrary*: σ)
case *0*
then show *?case* **by** *simp*
next
case (*Suc j*)
then show *?case*
by (*metis Nat.le-diff-conv2 One-nat-def Suc-eq-plus1-left Suc-leD*)
qed
qed

lemma *BoxInductSem*:
$(\sigma \models \Box\ (f \supset_i \mathit{wnext}\ f) \wedge_i f \supset_i \Box\ f)$
apply (*simp*)
using *box-induct-help-1* **by** (*metis One-nat-def diff-self-eq-0 not-one-le-zero*)

3.8.10 ChopStarEqv

lemma *chopstar-help-1*:
$(\ \exists l.\ l = \langle 0 \rangle \wedge \mathit{index\text{-}sequence}\ 0\ l \wedge$
$\mathit{Interval.nth}\ l\ (\mathit{intlen}\ l) = (\mathit{intlen}\ \sigma) \wedge$
$(\forall i.\ (0 \leq i \wedge i < (\mathit{intlen}\ l)) \longrightarrow$
$((\mathit{sub}\ (\mathit{nth}\ l\ i)\ (\mathit{nth}\ l\ (i+1))\ \sigma) \models f)$
$)) \longleftrightarrow (\mathit{intlen}\ \sigma = 0)$
by (*simp add*: *index-sequence-def*)

lemma *chopstar-help-2*:
$(\forall i.\ (0 < i \wedge i < 1 + (\mathit{intlen}\ ls)) \longrightarrow$
$((\mathit{sub}\ (\mathit{nth}\ ls\ (i-1))\ (\mathit{nth}\ ls\ ((i-1)+1))\ \sigma) \models f)$
$) =$
$(\forall i.\ (0 \leq i \wedge i < (\mathit{intlen}\ ls)) \longrightarrow$
$((\mathit{sub}\ (\mathit{nth}\ ls\ (i))\ (\mathit{nth}\ ls\ ((i)+1))\ \sigma) \models f)$
$)$
by (*metis Suc-eq-plus1 Suc-pred add-diff-cancel-right' add-less-cancel-left add-nonneg-pos le-add2 le-add-same-cancel2 plus-1-eq-Suc zero-less-one*)

lemma *chop-power-chain*:
$(\exists\ (l{::}\mathit{index}).\ (\mathit{intlen}\ l) = (\mathit{Suc}\ n) \wedge \mathit{index\text{-}sequence}\ 0\ l \wedge (\mathit{nth}\ l\ (\mathit{intlen}\ l)) = (\mathit{intlen}\ \sigma) \wedge$

```
                    (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                        ((sub (nth l i) (nth l (i+1)) σ) ⊨ f)
                    )
         ) =
  (∃ k. 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
            (sub 0 k σ ⊨ f) ∧
            (∃ ls. (intlen ls) = n ∧index-sequence 0 (ls) ∧
                   (nth (ls) (intlen (ls))) = (intlen (suffix k σ))
              ∧ (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                        ((sub (nth ls (i)) (nth ls ((i)+1)) (suffix k σ)) ⊨ f)
                ))
        )
proof −
 have (∃ (l::index). (intlen l) = (Suc n) ∧ index-sequence 0 l ∧
                  (nth l (intlen l)) = (intlen σ) ∧
                    (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                        ((sub (nth l i) (nth l (i+1)) σ) ⊨ f)
                    )
          )
     =
     (∃ x ls l.  (intlen l) = (Suc n) ∧ l=x⊙ls ∧ index-sequence 0 l ∧
               (nth l (intlen l)) = (intlen σ) ∧
                    (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                        ((sub (nth l i) (nth l (i+1)) σ) ⊨ f)
                    )
          )
 using interval-intlen-cons-1 by (metis zero-less-Suc)
 also have ... =
          (∃ x ls l.  (intlen l) = (Suc n) ∧ l=x⊙ls ∧ index-sequence 0 (x⊙ls) ∧
                   (nth (x⊙ls) (intlen (x⊙ls))) = (intlen σ) ∧
                    (∀ i. (0≤i ∧ i< (intlen (x⊙ls))) ⟶
                        ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f)
                    )
               )
 by auto
 also have ... =
          (∃ x ls . (intlen ls) = n ∧ index-sequence 0 (x⊙ls) ∧
                  (nth (x⊙ls) (intlen (x⊙ls))) = (intlen σ) ∧
                    (∀ i. (0≤i ∧ i< (intlen (x⊙ls))) ⟶
                        ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f)
                    )
             )
 by auto
 also have ... =
          (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence 0 (x ⊙ ls) ∧
                  (nth (ls) (intlen (ls))) = (intlen σ) ∧
                    ((∀ i. (0≤i ∧ i< (intlen (x⊙ls))) ⟶
                        ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f))
                    )
            )
```

```
by (simp add: index-sequence-def )
also have ... =
      (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence (nth ls 0) (ls) ∧
             (nth (ls) (intlen (ls))) = (intlen σ) ∧
            (x < (nth ls 0) ∧
            ((∀ i. (0≤i ∧ i< (intlen (x⊙ls))) ⟶
                ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f))
              )
            )
          )
using interval-idx-cons by auto
also have ... =
      (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence (nth ls 0) (ls) ∧
             (nth (ls) (intlen (ls))) = (intlen σ) ∧
            (x < (nth ls 0) ∧
             ((sub (nth (x⊙ls) 0) (nth (x⊙ls) (1)) σ) ⊨ f)
            ∧
            ((∀ i. (0<i ∧ i< 1+ (intlen (ls))) ⟶
                ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f))
              )
            )
          )
by (metis (no-types, lifting) One-nat-def add.right-neutral add-Suc add-Suc-right
     add-cancel-right-left interval-intlen-cons not-gr-zero zero-le zero-less-Suc)
also have ... =
      (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence (nth ls 0) (ls) ∧
           (nth (ls) (intlen (ls))) = (intlen σ) ∧
            (x < (nth ls 0) ∧ (nth (x⊙ls) 0) = x ∧ (nth (x⊙ls) (1)) = (nth ls 0) ∧
             ((sub (nth (x⊙ls) 0) (nth (x⊙ls) (1)) σ) ⊨ f)
            ∧
            ((∀ i. (0<i ∧ i< 1+ (intlen (ls))) ⟶
                ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f))
              )
            )
          )
by auto
also have ... =
      (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence (nth ls 0) (ls) ∧
               (nth (ls) (intlen (ls))) = (intlen σ) ∧
            (x < (nth ls 0) ∧ (nth (x⊙ls) 0) = x ∧ (nth (x⊙ls) (1)) = (nth ls 0) ∧
             ((sub x (nth ls 0) σ) ⊨ f)
            ∧
            ((∀ i. (0<i ∧ i< 1+ (intlen (ls))) ⟶
                ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f))
              )
            )
          )
by auto
also have ... =
      (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence (nth ls 0) (ls) ∧
                  (nth (ls) (intlen (ls))) = (intlen σ) ∧
                   (x < (nth ls 0) ∧
                    ((sub x (nth ls 0) σ) ⊨ f)
                   ∧
                   ((∀ i. (0<i ∧ i< 1+ (intlen (ls))) ⟶
                          ((sub (nth (x⊙ls) i) (nth (x⊙ls) (i+1)) σ) ⊨ f))
                       )
                   )
               )
by auto
also have ... =
          (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence (nth ls 0) (ls) ∧
                  (nth (ls) (intlen (ls))) = (intlen σ) ∧
              ( x < (nth ls 0) ∧
              ((sub x (nth ls 0) σ) ⊨ f)
              ∧
               (∀ i. (0<i ∧ i< 1+(intlen ls)) ⟶
                          ((sub (nth ls (i−1)) (nth ls ((i−1)+1)) σ) ⊨ f)
                      )))
using interval-nth-cons by metis
also have ... =
          (∃ x ls . (intlen ls) = n ∧ x =0 ∧ index-sequence (nth ls 0) (ls) ∧
                  (nth (ls) (intlen (ls))) = (intlen σ) ∧
              (x < (nth ls 0) ∧
              ((sub x (nth ls 0) σ) ⊨ f))
              ∧ (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                          ((sub (nth ls (i)) (nth ls ((i)+1)) σ) ⊨ f)
                 )
            )
using chopstar-help-2 by metis
also have ... =
          (∃  ls . (intlen ls) = n ∧ index-sequence (nth ls 0) (ls) ∧
                 (nth (ls) (intlen (ls))) = (intlen σ) ∧
              (0 < (nth ls 0) ∧
              ((sub 0 (nth ls 0) σ) ⊨ f))
              ∧ (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                          ((sub (nth ls (i)) (nth ls ((i)+1)) σ) ⊨ f)
                 )
            )
by simp
also have ... =
          (∃  lsk .  (intlen lsk) = n ∧ (nth lsk 0) ≤ intlen σ ∧ (nth lsk 0) > 0 ∧
                ((sub 0 (nth lsk 0) σ) ⊨ f) ∧
                index-sequence (nth lsk 0) (lsk) ∧
                (nth (lsk) (intlen (lsk))) = (intlen σ) ∧
                (∀ i. (0≤i ∧ i< (intlen lsk)) ⟶
                          ((sub (nth lsk (i)) (nth lsk ((i)+1)) σ) ⊨ f)
                 )
            )
by (metis Suc-eq-plus1 Suc-pred add.left-neutral eq-iff interval-idx-less-last
        interval-intlen-gr-zero le-neq-implies-less lessI less-imp-le-nat)
also have ... =
        (∃ k lsk. (intlen lsk) = n ∧ (nth lsk 0) ≤ intlen σ ∧
                (nth lsk 0) > 0 ∧ k=(nth lsk 0) ∧
              (sub 0 (nth lsk 0) σ ⊨ f) ∧
              index-sequence (nth lsk 0) (lsk) ∧
              (nth (lsk) (intlen (lsk))) = (intlen (σ)) ∧
              (∀ i. (0≤i ∧ i< (intlen lsk)) ⟶
                    ((sub ((nth lsk (i))) ((nth lsk ((i)+1))) (σ)) ⊨ f)
              )
    )
by auto
also have ... =
        (∃ k lsk. (intlen lsk) = n ∧ 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧ k=(nth lsk 0) ∧
           (sub 0 k σ ⊨ f) ∧
           (index-sequence k (lsk) ∧
                  (nth (lsk) (intlen (lsk))) = ((intlen (suffix k σ))+k) ∧
              (∀ i. (0≤i ∧ i< (intlen lsk)) ⟶
                    ((sub ((nth lsk (i))) ((nth lsk ((i)+1))) (σ)) ⊨ f)
              ))
    )
apply (simp add: interval-prefix-suffix-intlen interval-suffix-length interval-prefix-length)
by auto
also have ... =
        (∃ k lsk. (intlen lsk) = n ∧  0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
           (sub 0 k σ ⊨ f) ∧
           (index-sequence k (lsk) ∧
                  (nth (lsk) (intlen (lsk))) = ((intlen (suffix k σ))+k) ∧
              (∀ i. (0≤i ∧ i< (intlen lsk)) ⟶
                    ((sub ((nth lsk (i))) ((nth lsk ((i)+1))) (σ)) ⊨ f)
              ))
    )
using index-sequence-def by auto
also have ... =
        (∃ k.  0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
           (sub 0 k σ ⊨ f) ∧
           (∃ ls lsk. (intlen lsk) = n ∧ index-sequence k (lsk) ∧
                   ls = map (shiftm k) lsk ∧
                  (nth (lsk) (intlen (lsk))) = ((intlen (suffix k σ))+k) ∧
              (∀ i. (0≤i ∧ i< (intlen lsk)) ⟶
                    ((sub ((nth lsk (i))) ((nth lsk ((i)+1))) (σ)) ⊨ f)
              ))
    )
by blast
also have ... =
        (∃ k.  0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
           (sub 0 k σ ⊨ f) ∧
           (∃ ls lsk. (intlen lsk) = n ∧ index-sequence k (lsk) ∧
                   ls = map (shiftm k) lsk ∧
                   index-sequence 0 (ls) ∧ (intlen ls) = n ∧
```

```
               (nth (lsk) (intlen (lsk))) = ((intlen (suffix k σ))+k) ∧
          (∀ i. (0≤i ∧ i< (intlen lsk)) ⟶
                   ((sub ((nth lsk (i))) ((nth lsk ((i)+1))) (σ)) ⊨ f)
          ))
    )
using interval-idx-link-shiftm by blast
also have ... =
       (∃ k.  0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
          (sub 0 k σ ⊨ f) ∧
          (∃ ls lsk. (intlen lsk) = n  ∧ index-sequence k (lsk) ∧
               lsk = map (shift k) ls ∧
               index-sequence 0 (ls) ∧ (intlen ls) = n  ∧
               (nth (lsk) (intlen (lsk))) = ((intlen (suffix k σ))+k) ∧
          (∀ i. (0≤i ∧ i< (intlen lsk)) ⟶
                   ((sub ((nth lsk (i))) ((nth lsk ((i)+1))) (σ)) ⊨ f)
          ))
    )
using interval-lsk-ls by blast
also have ... =
       (∃ k ls lsk .  0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
          (sub 0 k σ ⊨ f) ∧
          ( (intlen lsk) = n  ∧ lsk = map (shift k) ls ∧
                index-sequence 0 (ls) ∧
                index-sequence k (lsk) ∧
               (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
          (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                   ((sub ((nth ls (i))+k) ((nth ls ((i)+1))+k) (σ)) ⊨ f)
          ))
    )
apply (simp add: Interval.shift-def interval-intlen-map interval-nth-map) by blast
also have ... =
       (∃ k ls lsk.  0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
          (sub 0 k σ ⊨ f) ∧
          ( (intlen lsk) = n  ∧ lsk = map (shift k) ls ∧
               (intlen ls) = n  ∧ index-sequence 0 (ls) ∧
               (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
          (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                   ((sub ((nth ls (i))+k) ((nth ls ((i)+1))+k) (σ)) ⊨ f)
          ))
    )
using interval-idx-link by blast
also have ... =
       (∃ k. 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
          (sub 0 k σ ⊨ f) ∧
          (∃ ls. (intlen ls) = n  ∧ index-sequence 0 (ls) ∧
             (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
          (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                   ((sub ((nth ls (i))+k) ((nth ls ((i)+1))+k) (σ)) ⊨ f)
          ))
    )
```

```
by (simp add: interval-intlen-map)
also have ... =
        (∃ k . 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
          (sub 0 k σ ⊨ f) ∧
          (∃ ls. (intlen ls) = n  ∧ index-sequence 0 (ls) ∧
               (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
             (∀ i≤intlen ls. Interval.nth ls i ≤ intlen (suffix k σ)) ∧
             (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                     ((sub ((nth ls (i))+k) ((nth ls ((i)+1))+k) (σ)) ⊨ f)
             )
          )
      )
using interval-idx-bound-1 by blast
also have ... =
        (∃ k. 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
          (sub 0 k σ ⊨ f) ∧
          (∃ ls. (intlen ls) = n  ∧ index-sequence 0 (ls) ∧
               (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
            (∀ i≤intlen ls. Interval.nth ls i ≤ intlen (suffix k σ))
           ∧ (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                     ((sub (nth ls (i)) (nth ls ((i)+1)) (suffix k σ)) ⊨ f)
             )
          )
      )
by (smt add.commute index-sequence-def interval-idx-expand interval-sub-suffix
      interval-suffix-length-good plus-1-eq-Suc)
also have ... =
        (∃ k. 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
          (sub 0 k σ ⊨ f) ∧
          (∃ ls. (intlen ls) = n  ∧ index-sequence 0 (ls) ∧
               (nth (ls) (intlen (ls))) = (intlen (suffix k σ))
           ∧ (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                     ((sub (nth ls (i)) (nth ls ((i)+1)) (suffix k σ)) ⊨ f)
             ))
      )
using interval-idx-bound-1 by blast
finally show (∃ (l::index). (intlen l) = (Suc n) ∧ index-sequence 0 l ∧
              (nth l (intlen l)) = (intlen σ) ∧
              (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                   ((sub (nth l i) (nth l (i+1)) σ) ⊨ f)
              )
             ) =
             (∃ k. 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧ (sub 0 k σ ⊨ f) ∧
              (∃ ls. (intlen ls) = n ∧index-sequence 0 (ls) ∧
                (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
                (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                   ((sub (nth ls (i)) (nth ls ((i)+1)) (suffix k σ)) ⊨ f)
                )
              )
             )
```

```
               .
qed

lemma chop-power-eqv-sem:
  (∃ n. (σ ⊨ (power-chop-d f n))) =
   ((σ ⊨ empty) ∨ (∃ n. (σ ⊨ (f ∧_i more); (power-chop-d f n))))
by (metis not0-implies-Suc power-chop-d.power-0 power-chop-d.power-Suc)

lemma chopstar-eqv-power-chop-help:
   ( σ ⊨ power-chop-d f n) =
    (∃ (l::index). intlen(l) = n ∧ index-sequence 0 l ∧
      (nth l (intlen l)) = (intlen ( σ)) ∧
      (∀ i. (0≤i ∧ i< (intlen l)) ⟶
            ((sub (nth l i) (nth l (i+1)) (σ)) ⊨ f)
      )
    )
proof
 (induct n arbitrary: σ)
 case 0
 then show ?case using index-sequence-def chopstar-help-1 by fastforce
 next
 case (Suc n)
 then show ?case
  proof −
   have 1: (σ ⊨ power-chop-d f (Suc n)) = (σ ⊨ ((f ∧_i more);(power-chop-d f n)))
   by simp
   have 2: (σ ⊨ ((f ∧_i more);(power-chop-d f n))) =
            (∃ k. 0 ≤k ∧ k ≤ intlen (σ) ∧ k > 0 ∧
              (prefix k (σ) ⊨ f) ∧
              (suffix k (σ) ⊨ power-chop-d f n)
            )

  by auto
  have 3: (∃ k. 0 ≤k ∧ k ≤ intlen (σ) ∧ k > 0 ∧
              (prefix k (σ) ⊨ f) ∧
              (suffix k (σ) ⊨ power-chop-d f n)
          ) =
          (∃ k. 0 ≤k ∧ k ≤ intlen (σ) ∧ k > 0 ∧
              (sub 0 k (σ) ⊨ f) ∧
              (suffix k (σ) ⊨ power-chop-d f n)
          )
  by (simp add: interval-sub-zero-prefix)
  have 4: (∃ k. 0 ≤k ∧ k ≤ intlen (σ) ∧ k > 0 ∧
              (sub 0 k (σ) ⊨ f) ∧
              (suffix k (σ) ⊨ power-chop-d f n)
          ) =
          (∃ k. 0 ≤k ∧ k ≤ intlen (σ) ∧ k > 0 ∧
              (sub 0 k (σ) ⊨ f) ∧
              (∃ (l::index). intlen(l) = n ∧ index-sequence 0 l ∧
                 (nth l (intlen l)) = (intlen (suffix k σ)) ∧
              (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                  ((sub (nth l i) (nth l (i+1)) (suffix k σ)) ⊨ f)
              )
            )
          )
  using Suc.hyps by auto
  have 5:
        (∃ (l::index). (intlen l) = (Suc n) ∧ index-sequence 0 l ∧
          (nth l (intlen l)) = (intlen σ) ∧
          (∀ i. (0≤i ∧ i< (intlen l)) ⟶
              ((sub (nth l i) (nth l (i+1)) σ) ⊨ f)
          )
        ) =
        (∃ k. 0 ≤k ∧ k ≤ intlen σ ∧ k > 0 ∧
           (sub 0 k σ ⊨ f) ∧
           (∃ ls. (intlen ls) = n ∧index-sequence 0 (ls) ∧
              (nth (ls) (intlen (ls))) = (intlen (suffix k σ)) ∧
              (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                   ((sub (nth ls (i)) (nth ls ((i)+1)) (suffix k σ)) ⊨ f)
              )
           )
        )
   using chop-power-chain by simp
   from 1 2 3 4 5 show ?thesis by auto
  qed
qed

lemma chopstar-eqv-power-chop:
 (σ ⊨ f⋆) = (∃ k. (σ ⊨ power-chop-d f k))
by (simp add: chopstar-eqv-power-chop-help)

lemma ChopstarEqvSem:
  (σ ⊨ f⋆ ≡ᵢ  empty ∨ᵢ  (f ∧ᵢ more); f⋆ )
using chopstar-eqv-power-chop
by (smt chop-power-eqv-sem iff-defs or-defs semantics-pitl.simps(5))
```

3.9 Time Reversal

```
lemma time-reverse-help-1:
 assumes index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen σ) ∧
         ls = map (λ x. (intlen σ) −x) (intrev l) ∧ index-sequence 0 ls
 shows  (∀ i.  0≤ i ∧ i<intlen ls ⟶
         (sub (nth ls ((intlen ls)−(i+1))) ((nth ls ((intlen ls) −i)) ) (intrev σ) ⊨ fʳ))
         =
         (∀ i. 0≤i ∧ i<intlen ls ⟶ (sub (nth ls (i)) ((nth ls (i+1)) ) (intrev σ) ⊨ fʳ))
by (smt Suc-diff-Suc Suc-eq-plus1 add-gr-0 diff-diff-cancel diff-less le-add-diff-inverse2
        le-simps(1) zero-less-one zero-order(1))

lemma TimeReverseSem:
 (σ ⊨ f) ⟷ ((intrev σ) ⊨ fʳ)
```

```
proof
 (induct f arbitrary: σ)
 case false-d
 then show ?case by auto
 next
 case (atom-d x)
 then show ?case by (simp add: interval-intlast-intrev)
 next
 case (implies-d f1 f2)
 then show ?case by simp
 next
 case skip-d
 then show ?case by simp
 next
 case (chop-d f1 f2)
 then show ?case using interval-intrev-prefix interval-intrev-suffix
 by (smt interval-intlen-gr-zero interval-intrev-intlen interval-prefix-length
        interval-rev-rev-ident interval-suffix-length-good order-refl rev-d.simps(5)
        semantics-pitl.simps(5))
 next
 case (chopstar-d f)
 then show ?case
  proof −
   have (σ ⊨  f ⋆) =
          (∃ l. index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen σ) ∧
                         (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                            ((sub (nth l i) (nth l (i+1)) σ) ⊨ f)
                         )
                        )
   by simp
   also have  ... =
            (∃ l. index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen σ) ∧
                         (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                            (intrev (sub (nth l i) (nth l (i+1)) σ) ⊨ f^r)
                         )
                        )
   using chopstar-d.hyps by simp
   also have  ... =
           (∃ l. index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen σ) ∧
                         (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                    ( (sub ((intlen σ)− (nth l (i+1))) ((intlen σ) − (nth l i)) (intrev σ)) ⊨ f^r)
                         )
                        )
   using interval-intrev-idx-2 by metis
   also have  ... =
          (∃ l ls. ls = map (λ x. (intlen σ) −x) (intrev l) ∧ index-sequence 0 ls ∧
                  index-sequence 0 l  ∧ (nth l (intlen l)) = (intlen σ) ∧
                         (∀ i. (0≤i ∧ i< (intlen l)) ⟶
                    ( (sub ((intlen σ)− (nth l (i+1))) ((intlen σ) − (nth l i)) (intrev σ)) ⊨ f^r)
                         )
                    )
  using interval-intrev-idx-7 by auto
  also have ... =
       (∃ l ls. ls = map (λ x. (intlen σ) −x) (intrev l) ∧
            index-sequence 0 ls ∧ index-sequence 0 l
         ∧ (nth l (intlen l)) = (intlen σ) ∧ (nth ls (intlen ls)) = (intlen σ) ∧
                 (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
             ( (sub ((intlen σ)− (nth l (i+1))) ((intlen σ) − (nth l i)) (intrev σ)) ⊨ f^r)
                    )
                    )
  by (metis interval-intrev-idx-3)
  also have ... =
        (∃ l ls. ls = map (λ x. (intlen σ) −x) (intrev l) ∧
             index-sequence 0 ls ∧ index-sequence 0 l
         ∧ (nth l (intlen l)) = (intlen σ) ∧ (nth ls (intlen ls)) = (intlen σ) ∧
                 (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
            ( sub (nth ls ((intlen ls)−(i+1))) ((nth ls ((intlen ls) −i)) ) (intrev σ) ⊨ f^r)
                    )
                    )

  using interval-intrev-idx-9 by metis
  also have ... =
        (∃ l ls. ls = map (λ x. (intlen σ) −x) (intrev l) ∧
             index-sequence 0 ls ∧ index-sequence 0 l
         ∧ (nth l (intlen l)) = (intlen σ) ∧ (nth ls (intlen ls)) = (intlen σ) ∧
                 (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                    ( (sub (nth ls (i)) ((nth ls (i+1)) ) (intrev σ)) ⊨ f^r)
                    )
                   )

  using time-reverse-help-1 by metis
  also have ... =
        (∃ ls. index-sequence 0 ls ∧
           (nth ls (intlen ls)) = (intlen σ) ∧
                 (∀ i. (0≤i ∧ i< (intlen ls)) ⟶
                    ( (sub (nth ls (i)) ((nth ls (i+1)) ) (intrev σ)) ⊨ f^r)
                    )
                   )
  using interval-intrev-idx-12 by (smt interval-intrev-idx-3 interval-intrev-idx-7)
  also have ... = (intrev σ ⊨ ( f^r ⋆) ) by simp
  also have ... = (intrev σ ⊨ ( f ⋆) ^r) by simp
  finally show (σ ⊨ f ⋆) = (intrev σ ⊨ ( f ⋆) ^r) .
 qed
qed

end


theory ITL
imports
```

Semantics
begin

4 Axioms and Rules

The ITL axiom and proof rules are introduced (taken from [3]) together with the validity operation. The soundness of the rules and axioms are checked using the lemmas of Semantics.thy.

definition *valid* :: $'a\ pitl \Rightarrow bool$ $((\vdash\ \text{-})\ 5)$
where $(\vdash f) = (\forall\ \sigma.\ (\sigma \models f))$

lemma *itl-valid* [*simp*] :
$(\vdash f) = (\forall\ \sigma.\ (\sigma \models f))$
by (*simp add*: *valid-def*)

lemma *itl-eq*:
$(\vdash f \equiv_i g) = (\forall\ \sigma.\ (\sigma \models f) = (\sigma \models g))$
by *simp*

lemma *EqvReverseReverse*:
$\vdash\ (f^r)^r \equiv_i f$
using *TimeReverseSem* **by** (*metis interval-rev-rev-ident itl-eq*)

lemma *ReverseEqv*:
$(\vdash f) \longleftrightarrow (\vdash f^r)$
by (*metis TimeReverseSem interval-rev-swap valid-def*)

4.1 Rules

lemma *MP* :
assumes $\vdash f \supset_i g$
$\vdash f$
shows $\vdash g$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *BoxGen* :
assumes $\vdash f$
shows $\vdash \Box\ f$
using *assms* **by** *auto*

lemma *BiGen*:
assumes $\vdash f$
shows $\vdash bi\ f$
using *assms* **by** *auto*

4.2 Axioms

lemma *ChopAssoc* :
$\vdash\ f\ ;\ (g\ ;\ h) \equiv_i\ (f;g);h$
using *ChopAssocSem valid-def* **by** *blast*

lemma *OrChopImp* :
  $\vdash (f \lor_i g);h \ \supset_i \ f;h \lor_i g;h$
using *OrChopImpSem valid-def* **by** *blast*

lemma *ChopOrImp* :
  $\vdash \ f;(g \lor_i h) \supset_i \ f;g \lor_i \ f;h$
using *ChopOrImpSem valid-def* **by** *blast*

lemma *EmptyChop* :
  $\vdash \ empty \ ; \ f \equiv_i f$
using *EmptyChopSem valid-def* **by** *blast*

lemma *ChopEmpty* :
  $\vdash f;empty \equiv_i f$
using *ChopEmptySem valid-def* **by** *blast*

lemma *StateImpBi* :
  $\vdash init \ f \supset_i \ bi \ (init \ f)$
using *StateImpBiSem valid-def* **by** *blast*

lemma *NextImpNotNextNot* :
  $\vdash \ \bigcirc f \supset_i \ \neg_i (\bigcirc \neg_i f)$
using *NextImpNotNextNotSem valid-def* **by** *blast*

lemma *BiBoxChopImpChop* :
  $\vdash \ bi \ (f \supset_i f1) \land_i \Box(g \supset_i g1) \supset_i f;g \supset_i f1;g1$
using *BiBoxChopImpChopSem valid-def* **by** *blast*

lemma *BoxInduct* :
  $\vdash \Box \ (f \supset_i wnext \ f) \land_i f \supset_i \Box \ f$
using *BoxInductSem valid-def* **by** *blast*

lemma *ChopstarEqv* :
  $\vdash \ f^\star \equiv_i \ empty \lor_i \ (f \land_i more); f^\star$
using *ChopstarEqvSem valid-def* **by** *blast*

end

theory *Theorems*
 imports
  ITL
begin

5 ITL theorems

We give the proofs of a list of ITL theorems. These proofs and theorems were from [5].

5.1 Propositional reasoning

This is a list of propositional logic theorems used in the proofs of the ITL theorems.

lemma *itl-prop*:
$\vdash (f \equiv_i f) \equiv_i \mathit{true}_i$
$\vdash (\neg_i\ \mathit{true}_i) \equiv_i \mathit{false}_i$
$\vdash (\neg_i\ \mathit{false}_i) \equiv_i \mathit{true}_i$
$\vdash (\neg_i \neg_i f) \equiv_i f$
$\vdash (\neg_i f \equiv_i f) \equiv_i \mathit{false}_i$
$\vdash (f \equiv_i \neg_i f) \equiv_i \mathit{false}_i$
$\vdash (\neg_i (f \equiv_i g)) \equiv_i (f \equiv_i \neg_i g)$
$\vdash (\mathit{true}_i \equiv_i f) \equiv_i f$
$\vdash (f \equiv_i \mathit{true}_i) \equiv_i f$
$\vdash (\mathit{true}_i \supset_i f) \equiv_i f$
$\vdash (\mathit{false}_i \supset_i f) \equiv_i \mathit{true}_i$
$\vdash (f \supset_i \mathit{true}_i) \equiv_i \mathit{true}_i$
$\vdash (f \supset_i f) \equiv_i \mathit{true}_i$
$\vdash (f \supset_i \mathit{false}_i) \equiv_i \neg_i f$
$\vdash (f \supset_i \neg_i f) \equiv_i \neg_i f$
$\vdash (f \wedge_i \mathit{true}_i) \equiv_i f$
$\vdash (\mathit{true}_i \wedge_i f) \equiv_i f$
$\vdash (f \wedge_i \mathit{false}_i) \equiv_i \mathit{false}_i$
$\vdash (\mathit{false}_i \wedge_i f) \equiv_i \mathit{false}_i$
$\vdash (f \wedge_i f) \equiv_i f$
$\vdash (f \wedge_i \neg_i f) \equiv_i \mathit{false}_i$
$\vdash (\neg_i f \wedge_i f) \equiv_i \mathit{false}_i$
$\vdash (f \vee_i \mathit{true}_i) \equiv_i \mathit{true}_i$
$\vdash (\mathit{true}_i \vee_i f) \equiv_i \mathit{true}_i$
$\vdash (f \vee_i \mathit{false}_i) \equiv_i f$
$\vdash (\mathit{false}_i \vee_i f) \equiv_i f$
$\vdash (f \vee_i f) \equiv_i f$
$\vdash (f \vee_i \neg_i f) \equiv_i \mathit{true}_i$
$\vdash (\neg_i f \vee_i f) \equiv_i \mathit{true}_i$
$(\vdash f \equiv_i f1) = (\vdash f1 \equiv_i f)$
$(\vdash f \equiv_i f1) = ((\vdash f \supset_i f1) \wedge (\vdash f1 \supset_i f))$
$(\vdash f \supset_i (f1 \wedge_i f2)) = ((\vdash f \supset_i f1) \wedge (\vdash f \supset_i f2))$
$(\vdash f \equiv_i f1) = (\vdash \neg_i f \equiv_i \neg_i f1)$
$(\vdash \neg_i (f \vee_i f1)) = (\vdash \neg_i f \wedge_i \neg_i f1)$
$(\vdash (f \supset_i f1)) = (\vdash (\neg_i f \vee_i f1))$
by *auto*

lemma *prop01*:
assumes $\vdash f \equiv_i g$
shows $\vdash \neg_i f \equiv_i \neg_i g$
using *assms itl-prop*(*33*) **by** *blast*

lemma *prop02*:
assumes $\vdash f \supset_i g$
$\vdash g \supset_i h$
shows $\vdash f \supset_i h$

using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop03*:
assumes $\vdash f \equiv_i g$
$\vdash g \equiv_i h$
shows $\vdash f \equiv_i h$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop04*:
$\vdash \neg_i\ (f \wedge_i g \wedge_i \neg_i h) \equiv_i (f \supset_i (g \supset_i h))$
by *auto*

lemma *prop05*:
assumes $\vdash f \equiv_i g$
shows $\vdash h \wedge_i f \equiv_i h \wedge_i g$
using *assms* **by** *auto*

lemma *prop06*:
assumes $\vdash f \equiv_i g$
shows $\vdash f \wedge_i h \equiv_i g \wedge_i h$
using *assms* **by** *auto*

lemma *prop07*:
assumes $\vdash f \equiv_i f1$
$\vdash g \equiv_i g1$
shows $\vdash (\mathit{if}_i\ \ (\mathit{init}\ w)\ \ \mathit{then}\ f\ \ \mathit{else}\ g) \equiv_i (\mathit{if}_i\ \ (\mathit{init}\ w)\ \ \mathit{then}\ f1\ \ \mathit{else}\ g1)$
using *assms*(*1*) *assms*(*2*) **by** *simp*

lemma *prop08*:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash f \wedge_i g \wedge_i f1 \supset_i h$
using *assms* **by** *auto*

lemma *prop09*:
assumes $\vdash f \wedge_i g \wedge_i h \supset_i\ f1$
$\vdash f \wedge_i h \wedge_i g \supset_i \neg_i f1$
shows $\vdash \neg_i (f \wedge_i h \wedge_i g)$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop10*:
assumes $\vdash f \wedge_i f1 \supset_i h$
shows $\vdash f \wedge_i g \wedge_i f1 \supset_i h$
using *assms* **by** *auto*

lemma *prop11*:
$\vdash (\mathit{if}_i\ \ g\ \ \mathit{then}\ \ f\ \ \mathit{else}\ f1) \supset_i ((g \supset_i f) \wedge_i (\neg_i g \supset_i f1))$
by *simp*

lemma *prop12*:
assumes $\vdash f \supset_i g$

shows $\vdash h \wedge_i f \supset_i h \wedge_i g$
using *assms* **by** *auto*

lemma *prop13*:
assumes $\vdash f \supset_i \neg_i g \vee_i h$
shows $\vdash g \wedge_i f \supset_i h$
using *assms* **by** *auto*

lemma *prop14*:
assumes $\vdash f \equiv_i g \vee_i h$
$\vdash h \supset_i h1$
shows $\vdash f \supset_i g \vee_i h1$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop15*:
assumes $\vdash f \equiv_i g$
$\vdash h \supset_i g$
shows $\vdash h \supset_i f$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop16*:
assumes $\vdash f \supset_i g \vee_i h$
$\vdash g \supset_i h1 \vee_i h$
shows $\vdash f \supset_i h1 \vee_i h$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop17*:
assumes $\vdash f \supset_i g$
$\vdash f1 \supset_i g$
shows $\vdash f \vee_i f1 \supset_i g$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop18*:
assumes $\vdash g \wedge_i h \supset_i h1$
$\vdash f \equiv_i g$
shows $\vdash f \wedge_i h \supset_i h1$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop19*:
assumes $\vdash f \equiv_i g \vee_i h$
shows $\vdash h \supset_i f$
using *assms* **by** *auto*

lemma *prop20*:
assumes $\vdash f \equiv_i g \vee_i h$
shows $\vdash \neg_i f \equiv_i \neg_i g \wedge_i \neg_i h$
using *assms* **by** *auto*

lemma *prop21*:
assumes $\vdash f \equiv_i h$
$\vdash f \equiv_i h1$
shows $\vdash h1 \equiv_i h$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop22*:
assumes $\vdash f \equiv_i h$
$\vdash g \equiv_i g1$
shows $\vdash f \wedge_i g \supset_i h \wedge_i g1$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop23*:
assumes $\vdash f \equiv_i g \vee_i h$
shows $\vdash f \wedge_i f1 \equiv_i (g \wedge_i f1) \vee_i (h \wedge_i f1)$
using *assms* **by** *auto*

lemma *prop24*:
assumes $\vdash f \equiv_i g \vee_i (h \wedge_i h1)$
$\vdash g \equiv_i g1$
$\vdash h \equiv_i h2 \wedge_i h3$
$\vdash h3 \wedge_i h1 \equiv_i h4$
shows $\vdash f \equiv_i g1 \vee_i (h2 \wedge_i h4)$
using *assms*(1) *assms*(2) *assms*(3) *assms*(4) **by** *auto*

lemma *prop25*:
assumes $\vdash f \supset_i false_i$
shows $\vdash g \vee_i f \equiv_i g$
using *assms* **by** *auto*

lemma *prop26*:
assumes $\vdash f \supset_i g$
shows $\vdash f \supset_i h \vee_i g$
using *assms* **by** *auto*

lemma *prop27*:
assumes $\vdash f \supset_i g$
shows $\vdash \neg_i g \supset_i \neg_i f$
using *assms* **by** *auto*

lemma *prop28*:
assumes $\vdash f \equiv_i g \vee_i h$
$\vdash h \equiv_i h1$
shows $\vdash f \equiv_i g \vee_i h1$
using *assms*(1) *assms*(2) **by** *auto*

lemma *prop29*:
assumes $\vdash f \supset_i g \vee_i h$
shows $\vdash f \wedge_i \neg_i g \supset_i h$
using *assms* **by** *auto*

lemma *prop30*:
assumes $\vdash f0 \supset_i g$
$\vdash f1 \supset_i g$
shows $\vdash f0 \vee_i f1 \supset_i g$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop31*:
$\vdash (f \supset_i (g \equiv_i h)) \equiv_i ((f \wedge_i g) \equiv_i (f \wedge_i h))$
by *auto*

lemma *prop32*:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash g \supset_i (f \supset_i h)$
using *assms* **by** *auto*

lemma *prop33*:
assumes $\vdash f0 \wedge_i g \supset_i h$
$\vdash f1 \wedge_i g \supset_i h$
shows $\vdash (f0 \vee_i f1) \wedge_i g \supset_i h$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop34*:
assumes $\vdash f \wedge_i g \supset_i h$
$\vdash f$
shows $\vdash g \supset_i h$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop35*:
assumes $\vdash f \supset_i g \vee_i h$
$\vdash h \supset_i h1$
shows $\vdash f \supset_i g \vee_i h1$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop36*:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash f \supset_i (g \supset_i h)$
using *assms* **by** *auto*

lemma *prop37*:
assumes $\vdash f1 \supset_i f$
$\vdash f \wedge_i f1 \wedge_i g \supset_i h$
shows $\vdash f1 \wedge_i g \supset_i h$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop38*:
assumes $\vdash f \supset_i g$
shows $\vdash f \equiv_i f \wedge_i g$
using *assms* **by** *auto*

lemma *prop39*:
 assumes $\vdash f \equiv_i f1$
 $\vdash g \equiv_i g1$
 shows $\vdash (f \supset_i g) \equiv_i (f1 \supset_i g1)$
using *assms*(*1*) *assms*(*2*) **by** *auto*

lemma *prop40*:
 assumes $\vdash f \equiv_i f1$
 $\vdash g \equiv_i g1$
 $\vdash h \equiv_i h1$
 shows $\vdash (f \equiv_i g \wedge_i h) \equiv_i (f1 \equiv_i g1 \wedge_i h1)$
using *assms*(*1*) *assms*(*2*) *assms*(*3*) **by** *auto*

5.2 State formulas

The *init* operator denotes state formulas, i.e., ITL formula that only constrain the first state of an interval.

lemma *Initprop* :
 $\vdash ((init\ f) \wedge_i (init\ g)) \equiv_i init(f \wedge_i g)$
 $\vdash (\neg_i\ (init\ f)) \equiv_i init\ (\ \neg_i\ f\ )$
 $\vdash ((init\ f) \vee_i (init\ g)) \equiv_i init\ (f \vee_i g)$
 $\vdash init\ true_i$
by *auto*

lemma *Finprop* :
 $\vdash (true_i;(f \wedge_i empty)) \wedge_i (true_i;(g \wedge_i empty)) \equiv_i (true_i;(f \wedge_i g \wedge_i empty))$
 $\vdash (true_i;(f \wedge_i empty)) \vee_i (true_i;(g \wedge_i empty)) \equiv_i (true_i;((f \vee_i g) \wedge_i empty))$
 $\vdash (true_i;(true_i \wedge_i empty))$
 $\vdash \neg_i\ (true_i;(f \wedge_i empty)) \equiv_i (true_i;(\neg_i f \wedge_i empty))$
apply *simp-all*
apply *auto*[*1*]
apply *auto*[*2*]
using *dual-order.order-iff-strict* **by** *fastforce*

5.3 Basic Theorems

lemma *BiChopImpChop* :
 $\vdash bi\ (f \supset_i f1) \supset_i f;g \supset_i f1;g$
 proof −
 have *1*: $\vdash g \supset_i g$ **by** *auto*
 hence *2*: $\vdash \Box\ (\ g \supset_i g)$ **by** (*rule BoxGen*)
 have *3*: $\vdash bi\ (\ f \supset_i f1) \wedge_i \Box(g \supset_i g) \supset_i f;g \supset_i f1;g$ **by** (*rule BiBoxChopImpChop*)
 from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *AndChopA*:
 $\vdash (f \wedge_i f1);g \supset_i f;g$
proof −
 have *1*: $\vdash f \wedge_i f1 \supset_i f$ **by** *auto*
 hence *2*: $\vdash bi\ (f \wedge_i f1 \supset_i f)$ **by** (*rule BiGen*)
 have *3*: $\vdash bi\ (f \wedge_i f1 \supset_i f) \supset_i (f \wedge_i f1);g \supset_i f;g$ **by** (*rule BiChopImpChop*)

from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *AndChopB*:
$\vdash (f \wedge_i f1);g \supset_i f1;g$
proof −
have *1*: $\vdash f \wedge_i f1 \supset_i f1$ **by** *auto*
hence *2*: $\vdash$ *bi* $(f \wedge_i f1 \supset_i f1)$ **by** (*rule BiGen*)
have *3*: $\vdash$ *bi* $(f \wedge_i f1 \supset_i f1) \supset_i (f \wedge_i f1);g \supset_i f1;g$ **by** (*rule BiChopImpChop*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *NextChop*:
$\vdash (\bigcirc f);g \equiv_i \bigcirc(f;g)$
proof −
have *1*: $\vdash$ *skip*;$(f;g) \equiv_i$ (*skip*;f);g **by** (*rule ChopAssoc*)
show *?thesis* **by** (*metis 1 itl-prop*(*30*) *next-d-def*)
qed

lemma *BoxChopImpChop* :
$\vdash \Box (g \supset_i g1) \supset_i f;g \supset_i f;g1$
proof −
have *1*: $\vdash g \supset_i g$ **by** *auto*
hence *2*: $\vdash$ *bi* $(g \supset_i g)$ **by** (*rule BiGen*)
have *3*: $\vdash$ *bi* $(f \supset_i f) \wedge_i \Box(g \supset_i g1) \supset_i f;g \supset_i f;g1$ **by** (*rule BiBoxChopImpChop*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *LeftChopImpChop*:
assumes $\vdash f \supset_i f1$
shows $\vdash f;g \supset_i f1;g$
proof −
have *1*: $\vdash f \supset_i f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash$ *bi* $(f \supset_i f1)$ **by** (*rule BiGen*)
have *3*: $\vdash$ *bi* $(f \supset_i f1) \supset_i f;g \supset_i f1;g$ **by** (*rule BiChopImpChop*)
from *2 3* **show** *?thesis* **using** *MP* **by** *blast*
qed

lemma *RightChopImpChop*:
assumes $\vdash g \supset_i g1$
shows $\vdash f;g \supset_i f;g1$
proof −
have *1*: $\vdash g \supset_i g1$ **using** *assms* **by** *auto*
hence *2*: $\vdash \Box (g \supset_i g1)$ **by** (*rule BoxGen*)
have *3*: $\vdash \Box (g \supset_i g1) \supset_i f;g \supset_i f;g1$ **by** (*rule BoxChopImpChop*)
from *2 3* **show** *?thesis* **using** *MP* **by** *blast*
qed

lemma *RightChopEqvChop*:
assumes $\vdash g \equiv_i g1$

```
 shows  ⊢ f;g ≡_i f;g1
proof −
 have 1: ⊢ g ≡_i g1 using assms by auto
 have 2: (⊢ g ⊃_i g1) ⟹ (⊢ f;g ⊃_i f;g1) by (rule RightChopImpChop)
 have 3: (⊢ g1 ⊃_i g) ⟹ (⊢ f;g1 ⊃_i f;g) by (rule RightChopImpChop)
 from 1 2 3 show ?thesis using itl-prop(31) by blast
qed

lemma ChopOrEqv:
 ⊢ f;(g ∨_i g1) ≡_i f;g ∨_i f;g1
proof −
 have  1: ⊢ g ⊃_i g ∨_i g1 by auto
 hence 2: ⊢ f;g ⊃_i f;(g ∨_i g1) by (rule RightChopImpChop)
 have  3: ⊢ g1 ⊃_i g ∨_i g1 by auto
 hence 4: ⊢ f;g1 ⊃_i f;(g ∨_i g1) by (rule RightChopImpChop)
 from 2 4 show ?thesis by auto
qed

lemma OrChopEqv:
 ⊢ (f ∨_i f1);g ≡_i f;g ∨_i f1;g
proof −
 have  1: ⊢ f ⊃_i f ∨_i f1 by auto
 hence 2: ⊢ f;g ⊃_i (f∨_i f1);g by (rule LeftChopImpChop)
 have  3: ⊢ f1 ⊃_i f ∨_i f1 by auto
 hence 4: ⊢ f1;g ⊃_i (f∨_i f1);g by (rule LeftChopImpChop)
 from 2 4 show ?thesis by auto
qed

lemma OrChopImpRule:
 assumes ⊢ f ⊃_i f1 ∨_i f2
 shows   ⊢ f;g ⊃_i (f1;g) ∨_i (f2;g)
proof −
 have  1: ⊢ f ⊃_i f1 ∨_i f2 using assms by auto
 hence 2: ⊢ f;g ⊃_i (f1 ∨_i f2);g by (rule LeftChopImpChop)
 have  3: ⊢ (f1 ∨_i f2); g ≡_i f1;g ∨_i f2;g by (rule OrChopEqv)
 from 2 3 show ?thesis by auto
qed

lemma  LeftChopEqvChop:
 assumes ⊢ f ≡_i f1
 shows   ⊢ f;g ≡_i f1;g
proof −
 have  1: ⊢ f ≡_i f1 using assms by auto
 hence 2: ⊢ f ⊃_i f1 by auto
 hence 3: ⊢ f;g ⊃_i f1;g by (rule LeftChopImpChop)
 from  1 have ⊢ f1 ⊃_i f by auto
 hence 4: ⊢ f1;g ⊃_i f;g by (rule LeftChopImpChop)
 from 3 4 show ?thesis by auto
qed
```

lemma *OrChopEqvRule*:
assumes $\vdash f \equiv_i f1 \vee_i f2$
shows $\vdash f;g \equiv_i (f1;g) \vee_i (f2;g)$
proof –
have *1*: $\vdash f \equiv_i f1 \vee_i f2$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;g \equiv_i (f1 \vee_i f2);g$ **by** (*rule LeftChopEqvChop*)
have *3*: $\vdash (f1 \vee_i f2);g \equiv_i f1;g \vee_i f2;g$ **by** (*rule OrChopEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *NextImpNext*:
assumes $\vdash f \supset_i g$
shows $\vdash \bigcirc f \supset_i \bigcirc g$
proof –
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash \Box (f \supset_i g)$ **by** (*rule BoxGen*)
have *3*: $\vdash \Box (f \supset_i g) \supset_i (skip;f) \supset_i (skip;g)$ **by** (*rule BoxChopImpChop*)
have *4*: $\vdash (skip;f) \supset_i (skip;g)$ **by** (*metis 2 3 MP*)
from *4* **show** *?thesis* **by** (*metis next-d-def*)
qed

lemma *ChopOrImpRule*:
assumes $\vdash g \supset_i g1 \vee_i g2$
shows $\vdash f;g \supset_i (f;g1) \vee_i (f;g2)$
proof –
have *1*: $\vdash g \supset_i g1 \vee_i g2$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;g \supset_i f;(g1 \vee_i g2)$ **by** (*rule RightChopImpChop*)
have *3*: $\vdash f;(g1 \vee_i g2) \equiv_i f;g1 \vee_i f;g2$ **by** (*rule ChopOrEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *NextImpDist*:
$\vdash \bigcirc (f \supset_i g) \supset_i \bigcirc f \supset_i \bigcirc g$
proof –
have *1*: $\vdash \neg_i (f \supset_i g) \equiv_i f \wedge_i \neg_i g$ **by** *auto*
hence *2*: $\vdash skip;\neg_i (f \supset_i g) \equiv_i skip;(f \wedge_i \neg_i g)$ **by** (*rule RightChopEqvChop*)
have *3*: $\vdash f \supset_i g \vee_i (f \wedge_i \neg_i g)$ **by** *auto*
hence *4*: $\vdash skip;f \supset_i (skip;g) \vee_i (skip;(f \wedge_i \neg_i g))$ **by** (*rule ChopOrImpRule*)
hence *5*: $\vdash \neg_i (skip;(f \wedge_i \neg_i g)) \supset_i (skip;f) \supset_i (skip;g)$ **by** *auto*
have *6*: $\vdash \neg_i (skip;\neg_i(f \supset_i g)) \supset_i (skip;f) \supset_i (skip;g)$ **by** *auto*
hence *7*: $\vdash \neg_i (\bigcirc\neg_i(f \supset_i g)) \supset_i (\bigcirc f) \supset_i (\bigcirc g)$ **by** (*simp add*: *next-d-def*)
have *8*: $\vdash \bigcirc(f \supset_i g) \supset_i \neg_i (\bigcirc\neg_i(f \supset_i g))$ **by** (*rule NextImpNotNextNot*)
from *7 8* **show** *?thesis* **by** *auto*
qed

lemma *ChopImpDiamond*:
$\vdash f;g \supset_i \Diamond g$
proof –
have *1*: $\vdash f \supset_i true_i$ **by** *auto*

hence *2*: $\vdash f;g \supset_i true_i;g$ **by** (*rule LeftChopImpChop*)
from *2* **show** *?thesis* **by** *auto*
qed

lemma *NowImpDiamond*:
$\vdash f \supset_i \Diamond f$
proof −
have *1*: $\vdash empty;f \equiv_i f$ **by** (*rule EmptyChop*)
have *2*: $\vdash empty \supset_i true_i$ **by** *auto*
hence *3*: $\vdash empty;f \supset_i true_i;f$ **by** (*rule LeftChopImpChop*)
have *4*: $\vdash f \supset_i true_i;f$ **using** *1 3* **by** *auto*
from *4* **show** *?thesis* **by** *auto*
qed

lemma *BoxElim*:
$\vdash \Box f \supset_i f$
proof −
have *1*: $\vdash \neg_i f \supset_i \Diamond \neg_i f$ **by** (*rule NowImpDiamond*)
hence *2*: $\vdash \neg_i (\Diamond \neg_i f) \supset_i f$ **by** *auto*
from *2* **show** *?thesis* **by** (*metis always-d-def*)
qed

lemma *NextDiamondImpDiamond*:
$\vdash \bigcirc (\Diamond f) \supset_i \Diamond f$
proof −
have *1*: $\vdash skip;(true_i;f) \equiv_i (skip;true_i);f$ **by** (*rule ChopAssoc*)
hence *2*: $\vdash (skip;true_i);f \equiv_i skip;(true_i;f)$ **by** *auto*
hence *3*: $\vdash (skip;true_i);f \equiv_i \bigcirc(\Diamond f)$ **by** (*simp add*: *next-d-def*)
have *4*: $\vdash (skip;true_i);f \supset_i \Diamond f$ **by** (*rule ChopImpDiamond*)
from *3 4* **show** *?thesis* **by** *auto*
qed

lemma *BoxImpNowAndWeakNext*:
$\vdash \Box f \supset_i (f \wedge_i wnext\ (\Box f))$
proof −
have *1*: $\vdash \neg_i f \supset_i \Diamond \neg_i f$ **by** (*rule NowImpDiamond*)
hence *2*: $\vdash \neg_i (\Diamond \neg_i f) \supset_i f$ **by** *auto*
hence *3*: $\vdash \Box f \supset_i f$ **by** (*metis always-d-def*)
have *4*: $\vdash \bigcirc (\Diamond \neg_i f) \supset_i \Diamond (\neg_i f)$ **by** (*rule NextDiamondImpDiamond*)
have *5*: $\vdash \neg_i \neg_i (\Diamond \neg_i f) \supset_i \Diamond(\neg_i f)$ **by** *auto*
hence *6*: $\vdash \bigcirc (\neg_i \neg_i (\Diamond \neg_i f)) \supset_i \bigcirc (\Diamond(\neg_i f))$ **by** (*rule NextImpNext*)
have *7*: $\vdash \bigcirc (\neg_i \neg_i (\Diamond \neg_i f)) \supset_i \Diamond (\neg_i f)$ **using** *4 6* **by** *auto*
hence *8*: $\vdash \bigcirc (\neg_i(\Box f)) \supset_i \Diamond (\neg_i f)$ **by** (*simp add*: *always-d-def*)
hence *9*: $\vdash \neg_i (\Diamond (\neg_i f)) \supset_i \neg_i (\bigcirc (\neg_i(\Box f)))$ **by** *auto*
hence *10*: $\vdash \Box f \supset_i wnext\ (\Box f)$ **by** (*simp add*: *always-d-def wnext-d-def*)
from *3 10* **show** *?thesis* **using** *itl-prop*(*32*) **by** *blast*
qed

lemma *BoxImpBoxRule*:

assumes $\vdash f \supset_i g$
shows $\vdash \Box f \supset_i \Box g$
proof −
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i g \supset_i \neg_i f$ **by** *auto*
hence *3*: $\vdash \Box(\neg_i g \supset_i \neg_i f)$ **by** (*rule BoxGen*)
have *4*: $\vdash \Box(\neg_i g \supset_i \neg_i f) \supset_i (true_i;\neg_i g) \supset_i (true_i;\neg_i f)$ **by** (*rule BoxChopImpChop*)
have *5*: $\vdash (true_i;\neg_i g) \supset_i (true_i;\neg_i f)$ **using** *3 4 MP* **by** *auto*
hence *6*: $\vdash \Diamond \neg_i g \supset_i \Diamond \neg_i f$ **by** (*simp add*: *sometimes-d-def*)
hence *7*: $\vdash \neg_i (\Diamond \neg_i f) \supset_i \neg_i(\Diamond \neg_i g)$ **by** *auto*
from *7* **show** *?thesis* **by** (*simp add*: *always-d-def*)
qed

lemma *BoxImpDist*:
$\vdash \Box(f \supset_i g) \supset_i \Box f \supset_i \Box g$
proof −
have *1*: $\vdash (f \supset_i g) \supset_i (\neg_i g \supset_i \neg_i f)$ **by** *auto*
hence *2*: $\vdash \Box(f \supset_i g) \supset_i \Box(\neg_i g \supset_i \neg_i f)$ **by** (*rule BoxImpBoxRule*)
have *3*: $\vdash \Box(\neg_i g \supset_i \neg_i f) \supset_i (true_i;\neg_i g) \supset_i (true_i;\neg_i f)$ **by** (*rule BoxChopImpChop*)
have *4*: $\vdash \Box(f \supset_i g) \supset_i (true_i;\neg_i g) \supset_i (true_i;\neg_i f)$ **using** *2 3 prop02* **by** *blast*
hence *5*: $\vdash \Box(f \supset_i g) \supset_i \Diamond\neg_i g \supset_i \Diamond\neg_i f$ **by** (*simp add*: *sometimes-d-def*)
hence *6*: $\vdash \Box(f \supset_i g) \supset_i \neg_i(\Diamond\neg_i f) \supset_i \neg_i(\Diamond\neg_i g)$ **by** *auto*
from *6* **show** *?thesis* **by** (*simp add*: *always-d-def*)
qed

lemma *DiamondEmpty*:
$\vdash \Diamond\ empty$
proof −
have *1*: $\vdash true_i$ **by** *auto*
have *2*: $\vdash true_i;\ empty \equiv_i true_i$ **by** (*rule ChopEmpty*)
have *3*: $\vdash true_i;\ empty$ **using** *1 2* **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *sometimes-d-def*)
qed

lemma *NextEqvNext*:
assumes $\vdash f \equiv_i g$
shows $\vdash \bigcirc f \equiv_i \bigcirc g$
proof −
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash skip;f \equiv_i skip;g$ **by** (*rule RightChopEqvChop*)
from *1* **show** *?thesis* **by** (*simp add*: *next-d-def*)
qed

lemma *NextAndNextImpNextRule*:
assumes $\vdash (f \wedge_i g) \supset_i h$
shows $\vdash (\bigcirc f \wedge_i \bigcirc g) \supset_i \bigcirc h$
using *assms* **by** *auto*

lemma *NextAndNextEqvNextRule*:
assumes $\vdash f \wedge_i g \equiv_i h$

```
 shows  ⊢ ○ f ∧ᵢ ○ g ≡ᵢ ○ h
using assms by auto

lemma WeakNextEqvWeakNext:
 assumes ⊢ f≡ᵢ g
 shows  ⊢ wnext  f ≡ᵢ wnext  g
using assms by auto

lemma DiamondImpDiamond:
 assumes ⊢ f⊃ᵢ g
 shows  ⊢ ◇ f ⊃ᵢ ◇ g
using assms by auto

lemma DiamondEqvDiamond:
 assumes ⊢ f≡ᵢ g
 shows  ⊢ ◇ f ≡ᵢ ◇ g
using assms by auto

lemma BoxEqvBox:
 assumes ⊢ f≡ᵢ g
 shows  ⊢ □ f ≡ᵢ □ g
using assms by auto

lemma BoxAndBoxImpBoxRule:
 assumes ⊢ f ∧ᵢ g ⊃ᵢ h
 shows  ⊢ □ f ∧ᵢ □ g ⊃ᵢ □ h
using assms by auto

lemma BoxAndBoxEqvBoxRule:
 assumes ⊢ f ∧ᵢ g ≡ᵢ h
 shows  ⊢ □ f ∧ᵢ □ g ≡ᵢ □ h
using assms by auto

lemma ImpBoxRule:
 assumes ⊢  f ⊃ᵢ g
 shows  ⊢ □ f ⊃ᵢ □ g
using assms by auto

lemma BoxIntro:
 assumes ⊢  f⊃ᵢ g
         ⊢  more ∧ᵢ f ⊃ᵢ○ f
 shows  ⊢ f⊃ᵢ□ g
proof −
 have  1: ⊢  more ∧ᵢ f ⊃ᵢ○ f using assms by auto
 hence 2: ⊢ f⊃ᵢ (empty ∨ᵢ ○f) by auto
 hence 3: ⊢ f ⊃ᵢ wnext f by auto
 hence 4: ⊢ □(f ⊃ᵢ wnext f) by (rule BoxGen)
 have  5: ⊢ (□ (f ⊃ᵢ wnext f)) ∧ᵢ f ⊃ᵢ □ f  by (rule BoxInduct)
 hence 6: ⊢ (□ (f ⊃ᵢ wnext f)) ⊃ᵢ (f ⊃ᵢ □f)  using prop36 by blast
 have  7: ⊢ f ⊃ᵢ □f using 4 6 MP by blast
 have  8: ⊢ □f ⊃_i f by (rule BoxElim)
 have  9: ⊢ f ≡_i □ f using 7 8 itl-prop(31) by blast
 have 10: ⊢ f ⊃_i g using assms by auto
 hence 11: ⊢ □f ⊃_i □ g by (rule ImpBoxRule)
 from 7 9 11 show ?thesis  using prop02 by blast
qed

lemma NextLoop:
 assumes ⊢ f⊃_i ○ f
 shows  ⊢ ¬_i  f
proof −
 have  1: ⊢ f⊃_i ○ f  using assms by auto
 hence  2: ⊢ f ⊃_i (more ∧_i wnext f) by auto
 hence  3: ⊢ f ⊃_i wnext f  by auto
 hence  4: ⊢ □(f ⊃_i wnext f) by (rule BoxGen)
 have  5: ⊢ □ (f ⊃_i wnext f) ∧_i f ⊃_i □ f  by (rule BoxInduct)
 hence  6: ⊢ □ (f ⊃_i wnext f) ⊃_i (f ⊃_i □f) using prop36 by blast
 have  7: ⊢ f ⊃_i □f  using 4 6 MP by blast
 have  8: ⊢ □f ⊃_i f by (rule BoxElim)
 have  9: ⊢ f ≡_i □ f using 7 8 itl-prop(31) by blast
 have 10: ⊢ f ⊃_i more using 2 by auto
 hence 11: ⊢ □ f ⊃_i □ more by (rule ImpBoxRule)
 have 12: ⊢ ¬_i(□ more)  by auto
 from 7 9 11 12 show ?thesis by (metis not-d-def prop02)
qed

lemma WnextEqvEmptyOrNext:
⊢ wnext f ≡_i empty ∨_i ○ f
by auto

lemma NotEmptyAndNext:
⊢ ¬_i(empty ∧_i ○ f)
by auto

lemma BoxEqvAndWnextBox:
⊢ □ f ≡_i f ∧_i wnext ( □ f)
proof −
 have 1: ⊢ □ f ⊃_i f ∧_i wnext ( □ f)
      using BoxImpNowAndWeakNext by blast
 have 2: ⊢ f ∧_i wnext ( □ f) ⊃_i f
      by simp
 have 3: ⊢ more ∧_i (f ∧_i wnext ( □ f) ) ⊃_i ○ (f ∧_i wnext ( □ f) )
      by (metis 1 NextImpNext WnextEqvEmptyOrNext empty-d-def prop10 prop13 prop14)
 have 4: ⊢ f ∧_i wnext ( □ f) ⊃_i □ f
      using 2 3 BoxIntro by blast
 from 1 4 show ?thesis using itl-prop(31) by blast
qed

lemma BoxEqvAndEmptyOrNextBox:
⊢ □f ≡_i f ∧_i (empty  ∨_i  ○(□ f))
```

using *BoxEqvAndWnextBox WnextEqvEmptyOrNext* **using** *prop03 prop05* **by** *blast*

lemma *BoxEqvBoxBox*:
$\vdash \square f \equiv_i \square\ (\square\ f)$
by *auto*

lemma *BoxBoxImpBox*:
$\vdash \square(\square h) \supset_i \square\ h$
using *BoxEqvBoxBox itl-prop*(*31*) **by** *blast*

lemma *BoxImpBoxBox*:
$\vdash \square\ h\ \supset_i\ \square(\square h)$
by *simp*

lemma *DiamondIntro*:
assumes $\vdash (f \wedge_i \neg_i\ g) \supset_i \bigcirc f$
shows $\vdash f \supset_i \Diamond\ g$
proof −
have *1*: $\vdash f \wedge_i \neg_i\ g \supset_i \bigcirc f$
using *assms* **by** *auto*
hence *2*: $\vdash f \wedge_i \neg_i\ g \wedge_i (\square \neg_i g) \supset_i (\bigcirc f) \wedge_i (\square \neg_i g)$
by *auto*
have *3*: $\vdash (\square \neg_i g) \supset_i \neg_i\ g$
by (*rule BoxElim*)
hence *4*: $\vdash \square \neg_i g \equiv_i (\square \neg_i g) \wedge_i \neg_i\ g$
using *BoxImpBoxBox BoxBoxImpBox itl-prop*(*31*) *itl-prop*(*32*) *prop02 prop26 prop29* **by** *blast*
have *5*: $\vdash f \wedge_i (\square \neg_i g) \supset_i \bigcirc f \wedge_i \square \neg_i g$
using *2 4* **by** *auto*
have *6*: $\vdash \square \neg_i g \equiv_i (\neg_i g) \wedge_i \mathit{wnext}(\square \neg_i g)$
using *BoxEqvAndWnextBox* **by** *blast*
have *7*: $\vdash \bigcirc f \wedge_i \square \neg_i g \supset_i\ \bigcirc f \wedge_i \mathit{wnext}(\square \neg_i g)$
using *6* **by** *auto*
have *8*: $\vdash f \wedge_i (\square \neg_i g) \supset_i \bigcirc f \wedge_i \mathit{wnext}(\square \neg_i g)$
using *5 7* **by** *auto*
hence *9*: $\vdash f \wedge_i (\square \neg_i g) \supset_i \mathit{more} \wedge_i \mathit{wnext}\ f \wedge_i \mathit{wnext}(\square \neg_i g)$
by *auto*
hence *10*: $\vdash f \wedge_i (\square \neg_i g) \supset_i \mathit{wnext}\ f \wedge_i \mathit{wnext}(\square \neg_i g)$
by *auto*
hence *11*: $\vdash f \wedge_i (\square \neg_i g) \supset_i \mathit{wnext}\ (f \wedge_i \square \neg_i g)$
by *auto*
hence *12*: $\vdash \square(f \wedge_i (\square \neg_i g) \supset_i \mathit{wnext}\ (f \wedge_i \square \neg_i g))$
by (*rule BoxGen*)
have *13*: $\vdash \square(f \wedge_i (\square \neg_i g) \supset_i \mathit{wnext}\ (f \wedge_i \square \neg_i g)) \wedge_i f \wedge_i (\square \neg_i g) \supset_i \square(f \wedge_i (\square \neg_i g))$
by (*rule BoxInduct*)
hence *14*: $\vdash \square(f \wedge_i (\square \neg_i g) \supset_i \mathit{wnext}\ (f \wedge_i \square \neg_i g)) \supset_i ((f \wedge_i (\square \neg_i g)) \supset_i \square(f \wedge_i (\square \neg_i g)))$
using *prop36* **by** *blast*
have *15*: $\vdash ((f \wedge_i (\square \neg_i g)) \supset_i \square(f \wedge_i (\square \neg_i g)))$
using *12 14 MP* **by** *blast*
have *16*: $\vdash \square(f \wedge_i (\square \neg_i g)) \supset_i (f \wedge_i (\square \neg_i g))$
by (*rule BoxElim*)

have *17*: $\vdash \Box(f \wedge_i (\Box \neg_i g)) \equiv_i (f \wedge_i (\Box \neg_i g))$
  using *16 15 itl-prop*(*31*) **by** *blast*
have *18*: $\vdash (f \wedge_i (\Box \neg_i g)) \supset_i \mathit{more}$
  using *9* **by** *auto*
hence *19*: $\vdash \Box(f \wedge_i (\Box \neg_i g)) \supset_i \Box\, \mathit{more}$
  by (*rule ImpBoxRule*)
have *20*: $\vdash \neg_i(\Box\, \mathit{more})$
  by *auto*
have *21*: $\vdash \neg_i(f \wedge_i (\Box \neg_i g))$
  using *17 19 20* **by** *auto*
hence *22*: $\vdash \neg_i f \vee_i \neg_i (\Box \neg_i g)$
  by *auto*
have *23*: $\vdash \neg_i (\Box \neg_i g) \equiv_i \Diamond g$
  by *auto*
from *22 23* **show** *?thesis* **by** *auto*
qed

lemma *DiamondIntroB*:
assumes $\vdash (f \wedge_i \neg_i g) \supset_i \bigcirc (f \wedge_i \neg_i g)$
shows $\vdash f \supset_i \Diamond g$
proof –
have *1*: $\vdash (f \wedge_i \neg_i g) \supset_i \bigcirc (f \wedge_i \neg_i g)$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i(f \wedge_i \neg_i g)$ **by** (*rule NextLoop*)
hence *3*: $\vdash f \supset_i g$ **by** *auto*
have *4*: $\vdash g \supset_i \Diamond g$ **by** (*rule NowImpDiamond*)
from *3 4* **show** *?thesis* **by** *auto*
qed

lemma *NextContra* :
assumes $\vdash (f \wedge_i \neg_i g) \supset_i (\bigcirc f \wedge_i \neg_i(\bigcirc g))$
shows $\vdash f \supset_i g$
proof –
have *1*: $\vdash (f \wedge_i \neg_i g) \supset_i (\bigcirc f \wedge_i \neg_i(\bigcirc g))$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i(f \supset_i g) \supset_i \bigcirc (\neg_i(f \supset_i g))$ **by** *auto*
hence *3*: $\vdash \neg_i \neg_i(f \supset_i g)$ **by** (*rule NextLoop*)
from *3* **show** *?thesis* **by** *auto*
qed

lemma *DiamondDiamondEqvDiamond*:
$\vdash \Diamond(\Diamond f) \equiv_i \Diamond f$
proof –
have *1*: $\vdash \mathit{true}_i;\mathit{true}_i \equiv_i \mathit{true}_i$ **by** *auto*
hence *2*: $\vdash (\mathit{true}_i;\mathit{true}_i);f \equiv_i \mathit{true}_i;f$ **using** *LeftChopEqvChop* **by** *blast*
have *3*: $\vdash (\mathit{true}_i;\mathit{true}_i);f \equiv_i \mathit{true}_i;(\mathit{true}_i;f)$ **using** *ChopAssoc itl-prop*(*30*) **by** *blast*
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *WeakNextDiamondInduct*:
assumes $\vdash \mathit{wnext}\ (\Diamond f) \supset_i f$

shows $\vdash f$
proof –
have *1*: $\vdash \mathit{wnext}\ (\Diamond f) \supset_i f$ **using** *assms* **by** *blast*
hence *2*: $\vdash \neg_i f \supset_i \neg_i(\mathit{wnext}\ (\Diamond f))$ **using** *prop27* **by** *blast*
hence *3*: $\vdash \neg_i f \supset_i \bigcirc(\neg_i (\Diamond f))$ **by** *auto*
have *4*: $\vdash f \supset_i \Diamond f$ **by** (*rule NowImpDiamond*)
hence *5*: $\vdash \neg_i(\Diamond f) \supset_i \neg_i f$ **by** *auto*
have *6*: $\vdash \neg_i f \supset_i \bigcirc(\neg_i f)$ **using** *3 5* **using** *NextImpNext prop02* **by** *blast*
hence *7*: $\vdash \neg_i \neg_i f$ **by** (*rule NextLoop*)
from *7* **show** *?thesis* **by** *auto*
qed

lemma *EmptyNextInducta*:
assumes $\vdash \mathit{empty} \supset_i f$
$\vdash \bigcirc f \supset_i f$
shows $\vdash f$
proof –
have *1*: $\vdash \mathit{empty} \supset_i f$ **using** *assms* **by** *auto*
have *2*: $\vdash \bigcirc f \supset_i f$ **using** *assms* **by** *blast*
have *3*: $\vdash (\mathit{empty} \vee_i \bigcirc f) \supset_i f$ **using** *1 2 prop17* **by** *blast*
have *4*: $\vdash \mathit{wnext}\ f \equiv_i (\mathit{empty} \vee_i \bigcirc f)$ **by** (*rule WnextEqvEmptyOrNext*)
hence *5*: $\vdash \mathit{wnext}\ f \supset_i f$ **using** *3* **using** *itl-prop*(*31*) *prop02* **by** *blast*
hence *6*: $\vdash \neg_i f \supset_i \neg_i (\mathit{wnext}\ f)$ **by** *auto*
hence *7*: $\vdash \neg_i f \supset_i \bigcirc(\neg_i f)$ **by** *auto*
hence *8*: $\vdash \neg_i \neg_i f$ **by** (*rule NextLoop*)
from *8* **show** *?thesis* **by** *auto*
qed

lemma *EmptyNextInductb*:
assumes $\vdash \mathit{empty} \wedge_i f \supset_i g$
$\vdash \bigcirc(f \supset_i g) \wedge_i f \supset_i g$
shows $\vdash f \supset_i g$
proof –
have *1*: $\vdash \mathit{empty} \wedge_i f \supset_i g$ **using** *assms* **by** *auto*
have *2*: $\vdash \bigcirc(f \supset_i g) \wedge_i f \supset_i g$ **using** *assms* **by** *blast*
have *3*: $\vdash (\mathit{empty} \vee_i \bigcirc(f \supset_i g)) \wedge_i f \supset_i g$ **using** *1 2 prop33* **by** *blast*
hence *4*: $\vdash \mathit{wnext}\ (f \supset_i g) \wedge_i f \supset_i g$ **using** *prop36* **by** *auto*
hence *5*: $\vdash \mathit{wnext}\ (f \supset_i g) \supset_i (f \supset_i g)$ **using** *prop36* **by** *blast*
hence *6*: $\vdash \neg_i (f \supset_i g) \supset_i \neg_i (\mathit{wnext}\ (f \supset_i g))$ **using** *prop27* **by** *blast*
hence *7*: $\vdash \neg_i (f \supset_i g) \supset_i \bigcirc (\neg_i(f \supset_i g))$ **by** *simp*
hence *8*: $\vdash \neg_i \neg_i (f \supset_i g)$ **by** (*rule NextLoop*)
from *8* **show** *?thesis* **by** *auto*
qed

lemma *FinImpFin*:
assumes $\vdash f \supset_i g$
shows $\vdash \mathit{fin}\ f \supset_i \mathit{fin}\ g$
using *ImpBoxRule assms* **by** *auto*

lemma *FinEqvFin*:
assumes $\vdash f \equiv_i g$
shows $\vdash$ *fin* $f \equiv_i$ *fin* g
using *FinImpFin assms itl-prop*(*31*) **by** *blast*

lemma *FinAndFinImpFinRule*:
assumes $\vdash f \wedge_i g \supset_i h$
shows $\vdash$ *fin* $f \wedge_i$ *fin* $g \supset_i$ *fin* h
proof −
have $\vdash f \wedge_i g \supset_i h$ **using** *assms* **by** *auto*
then show *?thesis* **by** *simp*
qed

lemma *FinAndFinEqvFinRule*:
assumes $\vdash f \wedge_i g \equiv_i h$
shows $\vdash$ *fin* $f \wedge_i$ *fin* $g \equiv_i$ *fin* h
by (*meson FinAndFinImpFinRule FinImpFin assms itl-prop*(*31*) *itl-prop*(*32*))

lemma *HaltEqvHalt*:
assumes $\vdash f \equiv_i g$
shows $\vdash$ *halt* $f \equiv_i$ *halt* g
proof −
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash$ (*empty* $\equiv_i f$) $\equiv_i$ (*empty* $\equiv_i g$) **by** *auto*
hence *3*: $\vdash \Box$(*empty* $\equiv_i f$) $\equiv_i \Box$ (*empty* $\equiv_i g$) **by** (*rule BoxEqvBox*)
from *3* **show** *?thesis* **by** (*simp add*: *halt-d-def*)
qed

lemma *BiImpDiImpDi*:
$\vdash$ *bi* $(f \supset_i g) \supset_i$ *di* $f \supset_i$ *di* g
proof −
have *1*: $\vdash$ *bi* $(f \supset_i g) \supset_i (f;$ *true*$_i) \supset_i (g;$ *true*$_i)$ **by** (*rule BiChopImpChop*)
from *1* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiImpDi*:
assumes $\vdash f \supset_i g$
shows $\vdash$ *di* $f \supset_i$ *di* g
proof −
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;$ *true*$_i \supset_i g;$ *true*$_i$ **by** (*rule LeftChopImpChop*)
from *2* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *BiImpBiRule*:
assumes $\vdash f \supset_i g$
shows $\vdash$ *bi* $f \supset_i$ *bi* g

proof −
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i\ g \supset_i \neg_i\ f$ **by** *auto*
hence *3*: $\vdash\ \mathit{di}\ \neg_i\ g \supset_i\ \mathit{di}\ \neg_i\ f$ **by** (*rule DiImpDi*)
hence *4*: $\vdash \neg_i\ (\ \mathit{di}\ \neg_i\ f) \supset_i \neg_i\ (\ \mathit{di}\ \neg_i\ g)$ **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*: *bi-d-def*)
qed

lemma *DiEqvDi*:
assumes $\vdash f \equiv_i g$
shows $\vdash\ \mathit{di}\ f \equiv_i\ \mathit{di}\ g$
proof −
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;\ \mathit{true}_i \equiv_i g;\ \mathit{true}_i$ **by** (*rule LeftChopEqvChop*)
from *2* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *BiEqvBi*:
assumes $\vdash f \equiv_i g$
shows $\vdash\ \mathit{bi}\ f \equiv_i\ \mathit{bi}\ g$
proof −
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i\ f \equiv_i \neg_i\ g$ **by** *auto*
hence *3*: $\vdash\ \mathit{di}\ \neg_i\ f \equiv_i\ \mathit{di}\ \neg_i\ g$ **by** (*rule DiEqvDi*)
hence *4*: $\vdash \neg_i\ (\mathit{di}\ \neg_i\ f) \equiv_i \neg_i\ (\ \mathit{di}\ \neg_i\ g)$ **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*: *bi-d-def*)
qed

lemma *LeftChopChopImpChopRule*:
assumes $\vdash (f;\ g) \supset_i g$
shows $\vdash (f;\ g);\ h \supset_i (g;\ h)$
proof −
have *1*: $\vdash (f;\ g) \supset_i g$ **using** *assms* **by** *blast*
hence *2*: $\vdash (f;\ g);\ h \supset_i g;\ h$ **by** (*rule LeftChopImpChop*)
have *3*: $\vdash f;\ (g;\ h) \equiv_i (f;\ g);\ h$ **by** (*rule ChopAssoc*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *AndChopCommute* :
$\vdash (f \wedge_i \mathit{f1});\ g \equiv_i (\mathit{f1} \wedge_i f);\ g$
proof −
have *1*: $\vdash f \wedge_i \mathit{f1} \equiv_i \mathit{f1} \wedge_i f$ **by** *auto*
from *1* **show** *?thesis* **by** (*rule LeftChopEqvChop*)
qed

lemma *BiAndChopImport*:
$\vdash\ \mathit{bi}\ f \wedge_i (\mathit{f1};\ g) \supset_i (f \wedge_i \mathit{f1});\ g$
proof −
have *1*: $\vdash f \supset_i (\mathit{f1} \supset_i f \wedge_i \mathit{f1})$ **by** *auto*
hence *2*: $\vdash \mathit{bi}\ f \supset_i \mathit{bi}\ (\mathit{f1} \supset_i f \wedge_i \mathit{f1})$ **by** (*rule BiImpBiRule*)

have *3*: $\vdash bi\ \ (f1 \supset_i (f \wedge_i f1)) \supset_i\ \ f1;\ g \supset_i (f \wedge_i f1);\ g$ **by** (*rule BiChopImpChop*)
from *1 3* **show** *?thesis* **using** *MP* **by** *auto*
qed

lemma *StateAndChopImport*:
$\vdash (init\ w) \wedge_i (f;\ g) \supset_i ((init\ w) \wedge_i f);\ g$
proof −
have *1*: $\vdash (init\ w) \supset_i bi\ \ (init\ w)$ **by** (*rule StateImpBi*)
hence *2*: $\vdash (init\ w) \wedge_i (f;\ g) \supset_i bi\ (init\ w) \wedge_i (f;\ g)$ **by** *auto*
have *3*: $\vdash bi\ \ (init\ w) \wedge_i (f;\ g) \supset_i ((init\ w) \wedge_i f);\ g$ **by** (*rule BiAndChopImport*)
from *2 3* **show** *?thesis* **using** *MP* **by** *auto*
qed

5.4 Further Properties Di and Bi

lemma *ImpDi*:
$\vdash f \supset_i\ di\ \ f$
proof −
have *1*: $\vdash f;\ \ empty\ \equiv_i f$ **by** (*rule ChopEmpty*)
have *2*: $\vdash empty\ \supset_i true_i$ **by** *auto*
hence *3*: $\vdash f;\ \ empty\ \supset_i f;\ true_i$ **by** (*rule RightChopImpChop*)
have *4* : $\vdash f \supset_i f;\ true_i$ **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiState*:
$\vdash di\ (init\ w) \equiv_i (init\ w)$
proof −
have *0*: $\vdash (init\ \neg_i w) \supset_i bi\ \ \ (init\ \neg_i w)$ **using** *StateImpBi* **by** *fastforce*
hence *1*: $\vdash \neg_i (init\ w) \supset_i bi\ \ \neg_i\ (init\ w)$ **using** *Initprop* **by** *auto*
hence *2*: $\vdash \neg_i\ \ (init\ w) \supset_i \neg_i\ (\ di\ \neg_i\ \neg_i\ \ (init\ w))$ **by** (*simp add*: *bi-d-def*)
have *3*: $\vdash (\neg_i\ \ (init\ w) \supset_i \neg_i\ \ (di\ \neg_i\ \neg_i\ \ (init\ w))) \supset_i (\ di\ \neg_i\ \neg_i\ \ (init\ w) \supset_i (init\ w))$ **by** *auto*
have *4*: $\vdash\ \ di\ \neg_i\ \neg_i\ \ (init\ w) \supset_i (init\ w)$ **using** *2 3 MP* **by** *blast*
have *5*: $\vdash (init\ w) \supset_i \neg_i\ \neg_i\ \ (init\ w)$ **by** *auto*
hence *6*: $\vdash\ \ di\ \ (init\ w) \supset_i\ \ di\ \neg_i\ \neg_i\ \ (init\ w)$ **by** (*rule DiImpDi*)
have *7*: $\vdash\ \ di\ \ (init\ w) \supset_i (init\ w)$ **using** *6 4 MP* **using** *prop02* **by** *blast*
have *8*: $\vdash (init\ w) \supset_i\ \ di\ \ (init\ w)$ **by** (*rule ImpDi*)
from *7 8* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *StateChop*:
$\vdash (init\ w);\ f \supset_i (init\ w)$
using *DiState* **by** *auto*

lemma *StateChopExportA*:
$\vdash ((init\ w) \wedge_i f);\ g \supset_i (init\ w)$
using *DiState* **by** *auto*

lemma *StateAndChop*:
$\vdash ((init\ w) \wedge_i f);\ g \equiv_i (init\ w) \wedge_i (f;\ g)$

using *StateAndChopImport StateChopExportA AndChopB itl-prop*(*31*) *itl-prop*(*32*) **by** *blast*

lemma *StateAndChopImpChopRule*:
assumes $\vdash (init\ w) \wedge_i f \supset_i f1$
shows $\vdash (init\ w) \wedge_i (f;\, g) \supset_i (f1;\, g)$
proof −
have *1*: $\vdash (init\ w) \wedge_i f \supset_i f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash ((init\ w) \wedge_i f);\, g \supset_i f1;\, g$ **by** (*rule LeftChopImpChop*)
have *3*: $\vdash ((init\ w) \wedge_i f);\, g \equiv_i (init\ w) \wedge_i (f;\, g)$ **by** (*rule StateAndChop*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *StateImpChopEqvChop* :
assumes $\vdash (init\ w) \supset_i (f \equiv_i f1)$
shows $\vdash (init\ w) \supset_i ((f;\, g) \equiv_i (f1;\, g))$
proof −
have *1*: $\vdash (init\ w) \supset_i (f \equiv_i f1)$ **using** *assms* **by** *auto*
hence *2*: $\vdash (init\ w) \wedge_i f \supset_i f1$ **by** *auto*
hence *3*: $\vdash (init\ w) \wedge_i (f;\, g) \supset_i (f1;\, g)$ **by** (*rule StateAndChopImpChopRule*)
have *4*: $\vdash (init\ w) \wedge_i f1 \supset_i f$ **using** *1* **by** *auto*
hence *5*: $\vdash (init\ w) \wedge_i (f1;\, g) \supset_i (f;\, g)$ **by** (*rule StateAndChopImpChopRule*)
from *3 5* **show** *?thesis* **by** *auto*
qed

lemma *ChopEqvStateAndChop*:
assumes $\vdash f \equiv_i (init\ w) \wedge_i f1$
shows $\vdash (f;\, g) \equiv_i (init\ w) \wedge_i (f1;\, g)$
proof −
have *1*: $\vdash f \equiv_i (init\ w) \wedge_i f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;\, g \equiv_i ((init\ w) \wedge_i f1);\, g$ **by** (*rule LeftChopEqvChop*)
have *3*: $\vdash ((init\ w) \wedge_i f1);\, g \equiv_i (init\ w) \wedge_i (f1;\, g)$ **by** (*rule StateAndChop*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *DiIntro*:
$\vdash f \supset_i\ di\ f$
proof −
have *1*: $\vdash f;\ empty\ \equiv_i f$ **by** (*rule ChopEmpty*)
have *2*: $\vdash\ empty\ \supset_i true_i$ **by** *auto*
hence *3*: $\vdash \Box(\ empty\ \supset_i true_i)$ **by** (*rule BoxGen*)
have *4*: $\vdash \Box(\ empty\ \supset_i true_i) \supset_i\ (f;\ empty\ \supset_i f;\, true_i)$ **by** (*rule BoxChopImpChop*)
have *5*: $\vdash f;\ empty\ \supset_i f;\, true_i$ **using** *3 4 MP* **by** *auto*
hence *6*: $\vdash f;\ empty\ \supset_i\ di\ f$ **by** (*simp add*: *di-d-def*)
from *1 6* **show** *?thesis* **by** *auto*
qed

lemma *BiElim*:
$\vdash\ bi\ f \supset_i f$
proof −
have *1*: $\vdash \neg_i\ f \supset_i\ di\ \neg_i\ f$ **by** (*rule DiIntro*)

have *2*: $\vdash (\neg_i\ f \supset_i\ di\ \neg_i\ f) \supset_i (\neg_i\ (\ di\ \neg_i\ f) \supset_i f)$ **by** *auto*
have *3*: $\vdash \neg_i\ (di\ \neg_i\ f) \supset_i f$ **using** *1 2 MP* **by** *blast*
from *3* **show** *?thesis* **by** (*metis bi-d-def*)
qed

lemma *BiContraPosImpDist*:
$\vdash bi\ (\neg_i\ g \supset_i \neg_i\ f) \supset_i (bi\ f) \supset_i (bi\ g)$
proof −
have *1*: $\vdash bi\ (\neg_i\ g \supset_i \neg_i\ f) \supset_i (\ di\ \neg_i\ g) \supset_i (\ di\ \neg_i\ f)$ **by** (*rule BiImpDiImpDi*)
hence *2*: $\vdash bi\ (\neg_i\ g \supset_i \neg_i\ f) \supset_i (\neg_i\ (\ di\ \neg_i\ f)) \supset_i (\neg_i\ (\ di\ \neg_i\ g))$ **by** *auto*
from *2* **show** *?thesis* **by** (*metis bi-d-def*)
qed

lemma *BiImpDist*:
$\vdash bi\ (f \supset_i g) \supset_i (bi\ f) \supset_i (bi\ g)$
proof −
have *1*: $\vdash (f \supset_i g) \supset_i (\neg_i\ g \supset_i \neg_i\ f)$ **by** *auto*
hence *2*: $\vdash \neg_i\ (\neg_i\ g \supset_i \neg_i\ f) \supset_i \neg_i\ (f \supset_i g)$ **by** *auto*
hence *3*: $\vdash bi\ (\neg_i\ (\neg_i\ g \supset_i \neg_i\ f) \supset_i \neg_i\ (f \supset_i g))$ **by** (*rule BiGen*)
have *4*: $\vdash bi\ (\neg_i\ (\neg_i\ g \supset_i \neg_i\ f) \supset_i \neg_i\ (f \supset_i g))$
$\supset_i$
$bi\ (f \supset_i g) \supset_i bi\ (\neg_i\ g \supset_i \neg_i\ f)$ **by** (*rule BiContraPosImpDist*)
have *5*: $\vdash bi\ (f \supset_i g) \supset_i bi\ (\neg_i\ g \supset_i \neg_i\ f)$ **using** *3 4 MP* **by** *blast*
have *6*: $\vdash bi\ (\neg_i\ g \supset_i \neg_i\ f) \supset_i (bi\ f) \supset_i (bi\ g)$ **by** (*rule BiContraPosImpDist*)
from *5 6* **show** *?thesis* **using** *prop02* **by** *blast*
qed

lemma *IfChopEqvRule*:
assumes $\vdash f \equiv_i\ if_i\ (init\ w)\ then\ f1\ else\ f2$
shows $\vdash f; g \equiv_i\ if_i\ (init\ w)\ then\ (f1; g)\ else\ (f2; g)$
proof −
have *1*: $\vdash f \equiv_i\ if_i\ (init\ w)\ then\ f1\ else\ f2$ **using** *assms* **by** *auto*
hence *2*: $\vdash f \equiv_i ((init\ w) \wedge_i f1) \vee_i\ (\ (init\ \neg_i\ w) \wedge_i f2)$ **by** (*simp add*: *ifthenelse-d-def*)
hence *3*: $\vdash f; g \equiv_i ((init\ w) \wedge_i f1); g \vee_i\ (\ \ (init\ \neg_i\ w) \wedge_i f2); g$ **by** (*rule OrChopEqvRule*)
have *4*: $\vdash ((init\ w) \wedge_i f1); g \equiv_i (init\ w) \wedge_i (f1; g)$ **by** (*rule StateAndChop*)
have *5*: $\vdash (\ \ (init\ \neg_i\ w) \wedge_i f2); g \equiv_i\ \ (init\ \neg_i\ w) \wedge_i (f2; g)$ **by** (*rule StateAndChop*)
have *6*: $\vdash f; g \equiv_i ((init\ w) \wedge_i f1; g)\ \vee_i\ (\ \ (init\ \neg_i\ w) \wedge_i f2; g)$ **using** *3 4 5* **by** *auto*
from *6* **show** *?thesis* **by** (*simp add*: *ifthenelse-d-def*)
qed

lemma *ChopOrEqvRule*:
assumes $\vdash g \equiv_i g1 \vee_i\ g2$
shows $\vdash f; g \equiv_i (f; g1) \vee_i\ (f; g2)$
proof −
have *1*: $\vdash g \equiv_i g1 \vee_i\ g2$ **using** *assms* **by** *auto*
hence *2*: $\vdash f; g \equiv_i f; (g1 \vee_i\ g2)$ **by** (*rule RightChopEqvChop*)
have *3*: $\vdash f; (g1 \vee_i\ g2) \equiv_i f; g1 \vee_i\ f; g2$ **by** (*rule ChopOrEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *EmptyOrChopEqv*:
$\vdash (\mathit{empty} \ \vee_i \ f); g \equiv_i g \vee_i \ (f; g)$
proof $-$
have *1*: $\vdash (\mathit{empty} \ \vee_i \ f); g \equiv_i (\mathit{empty} \ ; g) \vee_i \ (f; g)$ **by** (*rule OrChopEqv*)
have *2*: $\vdash \ \mathit{empty} \ ; g \equiv_i g$ **by** (*rule EmptyChop*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *EmptyOrNextChopEqv*:
$\vdash (\mathit{empty} \ \vee_i \ \bigcirc f); g \equiv_i g \vee_i \ \bigcirc(f; g)$
proof $-$
have *1*: $\vdash (\mathit{empty} \ \vee_i \ \bigcirc f); g \equiv_i g \vee_i \ ((\bigcirc f); g)$ **by** (*rule EmptyOrChopEqv*)
have *2*: $\vdash (\bigcirc f); g \equiv_i \bigcirc(f; g)$ **by** (*rule NextChop*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *EmptyOrChopImpRule*:
assumes $\vdash \ f \supset_i \ \mathit{empty} \ \vee_i \ \mathit{f1}$
shows $\vdash f; g \supset_i g \vee_i \ (\mathit{f1}; g)$
proof $-$
have *1*: $\vdash f \supset_i \ \mathit{empty} \ \vee_i \ \mathit{f1}$ **using** *assms* **by** *auto*
hence *2*: $\vdash f; g \supset_i (\mathit{empty} \ \vee_i \ \mathit{f1}); g$ **by** (*rule LeftChopImpChop*)
have *3*: $\vdash (\mathit{empty} \ \vee_i \ \mathit{f1}); g \equiv_i g \vee_i \ (\mathit{f1}; g)$ **by** (*rule EmptyOrChopEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *EmptyOrChopEqvRule*:
assumes $\vdash \ f \equiv_i \ \mathit{empty} \ \vee_i \ \mathit{f1}$
shows $\vdash f; g \equiv_i g \vee_i \ (\mathit{f1}; g)$
proof $-$
have *1*: $\vdash f \equiv_i \ \mathit{empty} \ \vee_i \ \mathit{f1}$ **using** *assms* **by** *auto*
hence *2*: $\vdash f; g \equiv_i (\mathit{empty} \ \vee_i \ \mathit{f1}); g$ **by** (*rule LeftChopEqvChop*)
have *3*: $\vdash (\mathit{empty} \ \vee_i \ \mathit{f1}); g \equiv_i g \vee_i \ (\mathit{f1}; g)$ **by** (*rule EmptyOrChopEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *EmptyOrNextChopImpRule*:
assumes $\vdash f \supset_i \ \mathit{empty} \ \vee_i \ \bigcirc \mathit{f1}$
shows $\vdash f; g \supset_i g \vee_i \ \bigcirc(\mathit{f1}; g)$
proof $-$
have *1*: $\vdash f \supset_i \ \mathit{empty} \ \vee_i \ \bigcirc \mathit{f1}$ **using** *assms* **by** *auto*
hence *2*: $\vdash f; g \supset_i (\mathit{empty} \ \vee_i \ \bigcirc \mathit{f1}); g$ **by** (*rule LeftChopImpChop*)
have *3*: $\vdash (\mathit{empty} \ \vee_i \ \bigcirc \mathit{f1}); g \equiv_i g \vee_i \ \bigcirc(\mathit{f1}; g)$ **by** (*rule EmptyOrNextChopEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *EmptyOrNextChopEqvRule*:
assumes $\vdash f \equiv_i \ \mathit{empty} \ \vee_i \ \bigcirc \mathit{f1}$
shows $\vdash f; g \equiv_i g \vee_i \ \bigcirc(\mathit{f1}; g)$
proof $-$

have *1*: $\vdash f \equiv_i empty \ \vee_i \ \bigcirc f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;\, g \equiv_i (\ empty \ \vee_i \ \bigcirc f1);\, g$ **by** (*rule LeftChopEqvChop*)
have *3*: $\vdash (\ empty \ \vee_i \ \bigcirc f1);\, g \equiv_i g \vee_i \ \bigcirc(f1;\, g)$ **by** (*rule EmptyOrNextChopEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *ChopEmptyOrImpRule*:
assumes $\vdash \ g \supset_i \ empty \ \vee_i \ g1$
shows $\vdash f;\, g \supset_i f \vee_i \ (f;\, g1)$
proof –
have *1*: $\vdash g \supset_i \ empty \ \vee_i \ g1$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;\, g \supset_i (f;\ empty\ ) \vee_i \ (f;\, g1)$ **by** (*rule ChopOrImpRule*)
have *3*: $\vdash f;\ empty \ \equiv_i f$ **by** (*rule ChopEmpty*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *StateAndEmptyImpBoxState*:
$\vdash (init\ w) \wedge_i empty \supset_i \Box\ (init\ w)$
by *simp*

lemma *BoxEqvAndBox*:
$\vdash \Box\ f \equiv_i f \wedge_i \Box\ f$
by *fastforce*

lemma *NotBoxImpNotOrNotNextBox*:
$\vdash \neg_i(\ \Box\ f) \supset_i \neg_i f \vee_i \ \neg_i(\ \bigcirc (\Box\ f)\ )$
proof –
have *1*: $\vdash f \wedge_i wnext\ (\Box\ f) \equiv_i f \wedge_i \Box\ f$
by (*meson BoxEqvAndBox BoxEqvAndWnextBox prop21*)
have *2*: $\vdash \neg_i\ (wnext\ (\Box\ f)) \supset_i \neg_i\ (\bigcirc\ (\Box\ f))$
by (*metis* (*full-types*) *NextImpNotNextNot prop27 wnext-d-def*)
then show *?thesis* **using** *1* **by** (*metis* (*no-types*) *and-d-def itl-prop*(*33*) *prop19 prop35*)
qed

lemma *BoxStateChopBoxEqvBox*:
$\vdash \ \ \Box\ (init\ w);\ \Box\ (init\ w) \equiv_i \Box\ (init\ w)$
proof –
have *1*: $\vdash \Box\ (init\ w) \equiv_i (init\ w) \wedge_i (\ empty \ \vee_i \ \bigcirc(\Box\ (init\ w)))$
by (*rule BoxEqvAndEmptyOrNextBox*)
hence *2*: $\vdash \Box\ (init\ w);\ \Box\ (init\ w) \equiv_i$
$(init\ w) \wedge_i ((\ empty \ \vee_i \ \bigcirc(\Box\ (init\ w)));\ \Box\ (init\ w))$
by (*rule ChopEqvStateAndChop*)
have *3*: $\vdash (\ empty \ \vee_i \ \bigcirc(\Box\ (init\ w)));\ \Box\ (init\ w) \equiv_i$
$\Box\ (init\ w) \vee_i \ \bigcirc(\Box\ (init\ w);\ \Box\ (init\ w))$
by (*rule EmptyOrNextChopEqv*)
have *4*: $\vdash \Box\ (init\ w);\ \Box\ (init\ w) \equiv_i$
$(init\ w) \wedge_i (\Box\ (init\ w) \vee_i \ \bigcirc(\Box\ (init\ w);\ \Box\ (init\ w)))$
using *2 3* **by** *auto*
have *5*: $\vdash \neg_i\ (\Box\ (init\ w)) \supset_i \neg_i \ (init\ w) \vee_i \ \neg_i(\ \bigcirc(\Box\ (init\ w)))$

by (*rule NotBoxImpNotOrNotNextBox*)
have *6*: $\vdash (\Box\ (\mathit{init}\ w); \Box\ (\mathit{init}\ w)) \wedge_i \neg_i(\Box\ (\mathit{init}\ w)) \supset_i$
$\bigcirc(\Box\ (\mathit{init}\ w); \Box\ (\mathit{init}\ w)) \wedge_i \neg_i(\bigcirc(\Box\ (\mathit{init}\ w)))$
using *4 5* **by** *auto*
hence *7*: $\vdash \Box\ (\mathit{init}\ w); \Box\ (\mathit{init}\ w) \supset_i \Box\ (\mathit{init}\ w)$
by (*rule NextContra*)
have *11*: $\vdash \Box\ (\mathit{init}\ w) \equiv_i (\mathit{init}\ w) \wedge_i \Box\ (\mathit{init}\ w)$
by (*rule BoxEqvAndBox*)
have *12*: $\vdash \mathit{empty}\ ; \Box\ (\mathit{init}\ w) \equiv_i \Box\ (\mathit{init}\ w)$
by (*rule EmptyChop*)
have *13*: $\vdash ((\mathit{init}\ w) \wedge_i\ \mathit{empty}\ ); \Box\ (\mathit{init}\ w) \equiv_i (\mathit{init}\ w) \wedge_i (\ \mathit{empty}\ ; \Box\ (\mathit{init}\ w))$
by (*rule StateAndChop*)
have *14*: $\vdash \Box\ (\mathit{init}\ w) \equiv_i ((\mathit{init}\ w) \wedge_i\ \mathit{empty}\ ); \Box\ (\mathit{init}\ w)$
using *11 12 13* **by** *auto*
have *15*: $\vdash (\mathit{init}\ w) \wedge_i\ \mathit{empty}\ \supset_i \Box\ (\mathit{init}\ w)$
by (*rule StateAndEmptyImpBoxState*)
hence *16*: $\vdash ((\mathit{init}\ w) \wedge_i\ \mathit{empty}\ ); \Box\ (\mathit{init}\ w) \supset_i \Box\ (\mathit{init}\ w); \Box\ (\mathit{init}\ w)$
by (*rule LeftChopImpChop*)
have *17*: $\vdash \Box\ (\mathit{init}\ w) \supset_i \Box\ (\mathit{init}\ w); \Box\ (\mathit{init}\ w)$
using *14 16* **by** *auto*
from *7 17* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *NotBoxStateImpBoxYieldsNotBox*:
$\vdash\ \ \neg_i(\Box\ (\mathit{init}\ w)) \supset_i (\Box\ (\mathit{init}\ w))\ \mathit{yields}\ \neg_i(\Box\ (\mathit{init}\ w))$
proof –
have *1*: $\vdash \Box\ (\mathit{init}\ w); \Box\ (\mathit{init}\ w) \equiv_i \Box\ (\mathit{init}\ w)$ **by** (*rule BoxStateChopBoxEqvBox*)
have *2*: $\vdash \Box\ (\mathit{init}\ w) \equiv_i \neg_i\ \neg_i(\Box\ (\mathit{init}\ w))$ **by** *auto*
hence *3*: $\vdash \Box\ (\mathit{init}\ w); \Box\ (\mathit{init}\ w) \equiv_i \Box\ (\mathit{init}\ w); \neg_i\ \neg_i(\Box\ (\mathit{init}\ w))$ **by** (*rule RightChopEqvChop*)
have *4*: $\vdash \neg_i(\Box\ (\mathit{init}\ w)) \supset_i \neg_i\ (\Box\ (\mathit{init}\ w); \neg_i\ \neg_i\ (\Box\ (\mathit{init}\ w)))$ **using** *1 3* **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *StateEqvBi*:
$\vdash\ \ (\mathit{init}\ w) \equiv_i \mathit{bi}\ \ (\mathit{init}\ w)$
proof –
have *1*: $\vdash (\mathit{init}\ w) \supset_i \mathit{bi}\ \ (\mathit{init}\ w)$ **by** (*rule StateImpBi*)
have *2*: $\vdash \mathit{bi}\ \ (\mathit{init}\ w) \supset_i (\mathit{init}\ w)$ **by** (*rule BiElim*)
from *1 2* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *TrueChopEqvDiamond*:
$\vdash\ \ \mathit{true}_i; f \equiv_i \Diamond\ f$
by *simp*

5.5 Properties of Da and Ba

lemma *DaEqvDtDi*:
$\vdash\ \ \ \mathit{da}\ \ f \equiv_i \Diamond\ (\mathit{di}\ \ f)$

proof −
have *1*: $\vdash true_i; (f; true_i) \equiv_i true_i; (f; true_i)$ **by** *auto*
hence *2*: $\vdash true_i; (f; true_i) \equiv_i true_i;\ di\ \ f$ **by** (*simp add*: *di-d-def*)
have *3*: $\vdash true_i;\ di\ \ f \equiv_i \Diamond(\ di\ \ f)$ **by** (*rule TrueChopEqvDiamond*)
have *4*: $\vdash true_i; (f; true_i) \equiv_i \Diamond(\ di\ \ f)$ **using** *2 3* **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*:*da-d-def*)
qed

lemma *DaEqvDiDt*:
$\vdash\ \ da\ \ f \equiv_i\ di\ (\Diamond f)$
proof −
have *1*: $\vdash true_i; f \equiv_i \Diamond f$ **by** (*rule TrueChopEqvDiamond*)
hence *2*: $\vdash (true_i; f); true_i \equiv_i (\Diamond f); true_i$ **by** (*rule LeftChopEqvChop*)
hence *3*: $\vdash (true_i; f); true_i \equiv_i\ di(\Diamond f)$ **by** (*simp add*: *di-d-def*)
have *4*: $\vdash true_i; (f; true_i) \equiv_i (true_i; f); true_i$ **by** (*rule ChopAssoc*)
have *5*: $\vdash true_i; (f; true_i) \equiv_i\ di\ (\Diamond f)$ **using** *3 4* **by** *auto*
from *5* **show** *?thesis* **by** (*simp add*: *da-d-def*)
qed

lemma *DtDiEqvDiDt*:
$\vdash\ \ \Diamond (di\ \ f) \equiv_i\ di\ (\Diamond f)$
by (*metis ChopAssoc di-d-def sometimes-d-def*)

lemma *DiamondNotEqvNotBox*:
$\vdash \Diamond \neg_i\ f \equiv_i \neg_i (\Box f)$
by *simp*

lemma *BaEqvBiBt*:
$\vdash\ \ ba\ \ f \equiv_i bi(\Box f)$
proof −
have *1*: $\vdash\ da \neg_i\ f \equiv_i\ di(\Diamond \neg_i\ f)$ **by** (*rule DaEqvDiDt*)
have *2*: $\vdash \Diamond \neg_i\ f \equiv_i \neg_i(\Box f)$ **by** (*rule DiamondNotEqvNotBox*)
hence *3*: $\vdash\ di\ (\Diamond\neg_i\ f) \equiv_i\ di \neg_i (\Box f)$ **by** (*rule DiEqvDi*)
have *4*: $\vdash\ da \neg_i\ f \equiv_i\ di \neg_i(\Box f)$ **using** *1 3* **by** *auto*
hence *5*: $\vdash \neg_i\ (da \neg_i\ f) \equiv_i \neg_i (\ di \neg_i(\Box f))$ **by** *auto*
hence *6*: $\vdash \neg_i (\ da \neg_i\ f) \equiv_i bi(\Box f)$ **by** (*simp add*: *bi-d-def*)
from *6* **show** *?thesis* **by** (*simp add*: *ba-d-def*)
qed

lemma *DiNotEqvNotBi*:
$\vdash\ \ di \neg_i\ f \equiv_i \neg_i(\ bi\ \ f)$
proof −
have *1*: $\vdash bi\ \ f \equiv_i \neg_i (\ di \neg_i\ f)$ **by** (*simp add*: *bi-d-def*)
from *1* **show** *?thesis* **by** *auto*
qed

lemma *NotDiamondNotEqvBox*:
$\vdash \neg_i (\Diamond\neg_i f) \equiv_i \Box f$
by *simp*

lemma *BaEqvBtBi*:
$\vdash \quad ba \;\; f \equiv_i \Box\, (bi \;\; f)$
proof –
have *1*: $\vdash \;\; da \neg_i \;\; f \equiv_i \Diamond\, (di \neg_i \;\; f)$ **by** (*rule DaEqvDtDi*)
have *2*: $\vdash \;\; di \neg_i \;\; f \equiv_i \neg_i\, (bi \;\; f)$ **by** (*rule DiNotEqvNotBi*)
hence *3*: $\vdash \Diamond\, (di \neg_i \;\; f) \equiv_i \Diamond\neg_i\, (bi \;\; f)$ **by** (*rule DiamondEqvDiamond*)
have *4*: $\vdash \neg_i\, (\Diamond\neg_i\, (bi \;\; f)) \equiv_i \Box(bi \;\; f)$ **by** (*rule NotDiamondNotEqvBox*)
have *5*: $\vdash \neg_i\, (\; da \neg_i \;\; f) \equiv_i \Box(bi \;\; f)$ **using** *1 2 3* **by** *auto*
from *5* **show** *?thesis* **by** (*simp add*: *ba-d-def*)
qed

lemma *BtBiEqvBiBt*:
$\vdash \quad \Box\, (bi \;\; f) \equiv_i bi(\; \Box\, f)$
proof –
have *1*: $\vdash \quad ba \;\; f \equiv_i \Box\, (bi \;\; f)$ **by** (*rule BaEqvBtBi*)
have *2*: $\vdash \quad ba \;\; f \equiv_i bi(\; \Box\, f)$ **by** (*rule BaEqvBiBt*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *BoxStateEqvBaBoxState*:
$\vdash \quad \Box\, (init\ w) \equiv_i \;\; ba\, (\Box\, (init\ w))$
proof –
have *1*: $\vdash (init\ w) \equiv_i bi \;\; (init\ w)$ **by** (*rule StateEqvBi*)
hence *2*: $\vdash \Box\, (init\ w) \equiv_i \Box\, (bi \;\; (init\ w))$ **by** (*rule BoxEqvBox*)
have *3*: $\vdash \Box\, (bi \;\; (init\ w)) \equiv_i bi(\; \Box\, (init\ w))$ **by** (*rule BtBiEqvBiBt*)
have *4*: $\vdash \Box\, (init\ w) \equiv_i \Box(\Box\, (init\ w))$ **by** (*rule BoxEqvBoxBox*)
hence *5*: $\vdash bi(\; \Box\, (init\ w)) \equiv_i bi\, (\Box(\Box\, (init\ w)))$ **by** (*rule BiEqvBi*)
have *6*: $\vdash \;\; ba(\; \Box\, (init\ w)) \equiv_i bi(\; \Box(\Box\, (init\ w)))$ **by** (*rule BaEqvBiBt*)
from *2 3 5 6* **show** *?thesis* **by** *simp*
qed

lemma *BaImpBi*:
$\vdash \quad ba \;\; f \supset_i bi \;\; f$
proof –
have *1*: $\vdash \;\; ba \;\; f \equiv_i \Box(bi \;\; f)$ **by** (*rule BaEqvBtBi*)
have *2*: $\vdash \Box(bi \;\; f) \supset_i bi \;\; f$ **by** (*rule BoxElim*)
from *1 2* **show** *?thesis* **using** *MP* **using** *itl-prop*(*31*) *prop02* **by** *blast*
qed

lemma *BaImpBt*:
$\vdash \quad ba \;\; f \supset_i \Box\, f$
proof –
have *1*: $\vdash \;\; ba \;\; f \equiv_i bi(\; \Box\, f)$ **by** (*rule BaEqvBiBt*)
have *2*: $\vdash bi(\; \Box\, f) \supset_i \Box\, f$ **by** (*rule BiElim*)
from *1 2* **show** *?thesis* **using** *MP* **using** *itl-prop*(*31*) *prop02* **by** *blast*
qed

lemma *DiamondImpDa*:
$\vdash \quad \Diamond\, f \supset_i \;\; da \;\; f$
by (*metis DiIntro DiamondImpDiamond da-d-def di-d-def sometimes-d-def*)

lemma *DiImpDa*:
$\vdash\ \ di\ \ f \supset_i\ da\ \ f$
by (*metis NowImpDiamond da-d-def di-d-def sometimes-d-def*)

lemma *BoxAndChopImport*:
$\vdash\ \ \Box\ h \wedge_i f;\ g \supset_i f;\ (h \wedge_i g)$
proof −
have *1*: $\vdash h \supset_i g \supset_i (h\wedge_i g)$ **by** *auto*
hence *2*: $\vdash \Box\ h \supset_i \Box(g \supset_i (h\wedge_i g))$ **by** (*rule ImpBoxRule*)
have *3*: $\vdash \Box(g \supset_i (h\wedge_i g)) \supset_i f;\ g \supset_i f;\ (h\wedge_i g)$ **by** (*rule BoxChopImpChop*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *BaAndChopImport*:
$\vdash\ \ ba\ \ f \wedge_i (g;\ g1) \supset_i (f \wedge_i g);\ (f \wedge_i g1)$
proof −
have *1*: $\vdash ba\ \ f \supset_i bi\ \ f$ **by** (*rule BaImpBi*)
have *2*: $\vdash bi\ \ f \wedge_i (g;\ g1) \supset_i (f \wedge_i g);\ g1$ **by** (*rule BiAndChopImport*)
have *3*: $\vdash ba\ \ f \supset_i \Box\ f$ **by** (*rule BaImpBt*)
have *4*: $\vdash \Box\ f \wedge_i (f \wedge_i g);\ g1 \supset_i (f \wedge_i g);\ (f \wedge_i g1)$ **by** (*rule BoxAndChopImport*)
from *1 2 3 4* **show** *?thesis* **by** *simp*
qed

lemma *ChopAndCommute*:
$\vdash\ \ f;\ (g \wedge_i g1) \equiv_i f;\ (g1 \wedge_i g)$
proof −
have *1*: $\vdash (g \wedge_i g1) \equiv_i (g1 \wedge_i g)$ **by** *auto*
from *1* **show** *?thesis* **by** (*rule RightChopEqvChop*)
qed

lemma *ChopAndA*:
$\vdash\ \ f;\ (g \wedge_i g1) \supset_i f;\ g$
proof −
have *1*: $\vdash (g \wedge_i g1) \supset_i\ g$ **by** *auto*
from *1* **show** *?thesis* **by** (*rule RightChopImpChop*)
qed

lemma *ChopAndB*:
$\vdash\ \ f;\ (g \wedge_i g1) \supset_i f;\ g1$
proof −
have *1*: $\vdash (g \wedge_i g1) \supset_i\ g1$ **by** *auto*
from *1* **show** *?thesis* **by** (*rule RightChopImpChop*)
qed

lemma *BoxStateAndChopEqvChop*:
$\vdash\ \Box\ (init\ w) \wedge_i (f;\ g) \equiv_i (\Box\ (init\ w) \wedge_i f);\ (\Box\ (init\ w) \wedge_i g)$
proof −
have *1*: $\vdash \Box\ (init\ w) \equiv_i\ ba(\ \Box\ (init\ w))$
by (*rule BoxStateEqvBaBoxState*)

```
 have  2: ⊢ ba( □ (init w)) ∧ᵢ (f; g) ⊃ᵢ (□ (init w) ∧ᵢ f); (□ (init w) ∧ᵢ g)
      by (rule BaAndChopImport)
 have  3: ⊢ □ (init w) ∧ᵢ (f; g) ⊃ᵢ (□ (init w) ∧ᵢ f); (□ (init w) ∧ᵢ g)
      using 1 2 prop18 by blast
 have 11: ⊢ (□ (init w) ∧ᵢ f); (□ (init w) ∧ᵢ g) ⊃ᵢ (□ (init w)); (□ (init w) ∧ᵢ g)
      by (rule AndChopA)
 have 12: ⊢ (□ (init w)); (□ (init w) ∧ᵢ g) ⊃ᵢ (□ (init w)); (□ (init w))
      by (rule ChopAndA)
 have 13: ⊢ (□ (init w)); (□ (init w)) ≡ᵢ □ (init w)
      by (rule BoxStateChopBoxEqvBox)
 have 14: ⊢ (□ (init w) ∧ᵢ f); (□ (init w) ∧ᵢ g) ⊃ᵢ f; (□ (init w) ∧ᵢ g)
      by (rule AndChopB)
 have 15: ⊢ f; (□ (init w) ∧ᵢ g) ⊃ᵢ f; g
      by (rule ChopAndB)
 have 16: ⊢ (□ (init w) ∧ᵢ f); (□ (init w) ∧ᵢ g) ⊃ᵢ □ (init w) ∧ᵢ (f; g)
      using  11 12 13 14 15  using itl-prop(31) itl-prop(32) prop02 by metis
 from 3 16 show ?thesis using itl-prop(31) by blast
qed

lemma DiEqvNotBiNot:
 ⊢   di  f ≡ᵢ ¬ᵢ( bi ¬ᵢ  f)
proof −
 have  1: ⊢ bi ¬ᵢ  f ≡ᵢ ¬ᵢ ( di ¬ᵢ ¬ᵢ  f)  by (simp add: bi-d-def)
 hence 2: ⊢  di ¬ᵢ ¬ᵢ  f ≡ᵢ ¬ᵢ( bi ¬ᵢ  f)  by auto
 have  3: ⊢ f ≡ᵢ ¬ᵢ ¬ᵢ  f by auto
 hence 4: ⊢  di  f ≡ᵢ  di  ¬ᵢ ¬ᵢ  f by (rule DiEqvDi)
 from 2 4 show ?thesis by auto
qed

lemma ChopAndBoxImport:
⊢   f; g ∧ᵢ □ h ⊃ᵢ f; (g ∧ᵢ h)
proof −
 have 1: ⊢ □ h ∧ᵢ f; g ⊃ᵢ f; (h ∧ᵢ g)  by (rule BoxAndChopImport)
 have 2: ⊢ f; (h ∧ᵢ g) ≡ᵢ f; (g ∧ᵢ h)  by (rule ChopAndCommute)
 from 1 2 show ?thesis by auto
qed

lemma AndChopAndCommute:
⊢   (f ∧ᵢ g); (f1 ∧ᵢ g1) ≡ᵢ (g ∧ᵢ f); (g1 ∧ᵢ f1)
proof −
 have 1: ⊢ (f ∧ᵢ g); (f1 ∧ᵢ g1) ≡ᵢ (g ∧ᵢ f); (f1 ∧ᵢ g1)  by (rule AndChopCommute)
 have 2: ⊢ (g ∧ᵢ f); (f1 ∧ᵢ g1) ≡ᵢ (g ∧ᵢ f); (g1 ∧ᵢ f1)   by (rule ChopAndCommute)
 from 1 2 show ?thesis by auto
qed

lemma ChopImpChop:
 assumes ⊢   f⊃ᵢ f1  ⊢ g⊃ᵢ g1
 shows   ⊢ f; g ⊃ᵢ f1; g1
proof −
 have  1: ⊢ f ⊃ᵢ f1  using assms by auto
```

hence *2*: $\vdash f ; g \supset_i f1 ; g$ **by** (*rule LeftChopImpChop*)
have *3*: $\vdash g \supset_i g1$ **using** *assms* **by** *auto*
hence *4*: $\vdash f1 ; g \supset_i f1 ; g1$ **by** (*rule RightChopImpChop*)
from *2 4* **show** *?thesis* **by** *auto*
qed

lemma *ChopEqvChop*:
assumes $\vdash f \equiv_i f1$ $\vdash g \equiv_i g1$
shows $\vdash f ; g \equiv_i f1 ; g1$
proof −
have *1*: $\vdash f \equiv_i f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash f ; g \equiv_i f1 ; g$ **by** (*rule LeftChopEqvChop*)
have *3*: $\vdash g \equiv_i g1$ **using** *assms* **by** *auto*
hence *4*: $\vdash f1 ; g \equiv_i f1 ; g1$ **by** (*rule RightChopEqvChop*)
from *2 4* **show** *?thesis* **by** *auto*
qed

lemma *BoxImpBoxImpBox*:
$\vdash \Box\, h \supset_i \Box(g \supset_i \Box\, h \wedge_i g)$
proof −
have *1*: $\vdash \Box\, h \supset_i (g \supset_i \Box\, h \wedge_i g)$ **by** *simp*
hence *2*: $\vdash \Box(\Box\, h) \supset_i \Box(g \supset_i \Box\, h \wedge_i g)$ **by** (*rule ImpBoxRule*)
have *3*: $\vdash \Box\, h \equiv_i \Box(\Box h)$ **by** (*rule BoxEqvBoxBox*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *BoxChopImpChopBox*:
$\vdash \Box\, h \supset_i f ; g \supset_i f ; (\Box\, h \wedge_i g)$
proof −
have *1*: $\vdash \Box\, h \supset_i \Box(g \supset_i \Box\, h \wedge_i g)$ **by** (*rule BoxImpBoxImpBox*)
have *2*: $\vdash \Box(g \supset_i \Box\, h \wedge_i g) \supset_i f ; g \supset_i f ; (\Box\, h \wedge_i g)$ **by** (*rule BoxChopImpChop*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *NotChopEqvYieldsNot*:
$\vdash \neg_i (f ; g) \equiv_i f$ *yields* $\neg_i g$
proof −
have *1*: $\vdash g \equiv_i \neg_i \neg_i g$ **by** *auto*
hence *2*: $\vdash f ; g \equiv_i f ; \neg_i \neg_i g$ **by** (*rule RightChopEqvChop*)
hence *3*: $\vdash \neg_i (f ; g) \equiv_i \neg_i (f ; \neg_i \neg_i g)$ **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *NotDiFalse*:
$\vdash \neg_i (\mathit{di}\ \mathit{false}_i)$
proof −
have *1*: $\vdash (\mathit{init}\ \mathit{true}_i) \supset_i \mathit{bi}\ (\mathit{init}\ \mathit{true}_i)$ **by** (*rule StateImpBi*)
hence *2*: $\vdash \mathit{true}_i \supset_i \mathit{bi}\ \mathit{true}_i$ **by** *auto*
have *3*: $\vdash \mathit{true}_i$ **by** *auto*
have *4*: $\vdash \mathit{bi}\ \mathit{true}_i$ **using** *2 3 MP* **by** *auto*

hence *5*: $\vdash \neg_i\ (\ \mathit{di}\ \neg_i\ \mathit{true}_i)$ **by** *simp*
have *6*: $\vdash \neg_i\ \mathit{true}_i \equiv_i\ \mathit{false}_i$ **by** *auto*
hence *7*: $\vdash\ \mathit{di}\ \neg_i\ \mathit{true}_i \equiv_i\ \mathit{di}\ \mathit{false}_i$ **by** (*rule DiEqvDi*)
from *5 7* **show** *?thesis* **by** *auto*
qed

lemma *StateAndEmptyChop*:
$\vdash\ ((\mathit{init}\ w) \wedge_i\ \mathit{empty}\ );\ f \equiv_i (\mathit{init}\ w) \wedge_i f$
proof –
have *1*: $\vdash ((\mathit{init}\ w) \wedge_i\ \mathit{empty}\ );\ f \equiv_i (\mathit{init}\ w) \wedge_i\ \mathit{empty}\ ;\ f$ **by** (*rule StateAndChop*)
have *2*: $\vdash\ \mathit{empty}\ ;\ f \equiv_i f$ **by** (*rule EmptyChop*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *StateAndNextChop*:
$\vdash\ ((\mathit{init}\ w) \wedge_i \bigcirc f);\ g \equiv_i (\mathit{init}\ w) \wedge_i \bigcirc(f;\ g)$
proof –
have *1*: $\vdash ((\mathit{init}\ w) \wedge_i \bigcirc f);\ g \equiv_i (\mathit{init}\ w) \wedge_i (\bigcirc f);\ g$ **by** (*rule StateAndChop*)
have *2*: $\vdash (\bigcirc f);\ g \equiv_i \bigcirc(f;\ g)$ **by** (*rule NextChop*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *NextAndEqvNextAndNext*:
$\vdash \bigcirc (f \wedge_i g) \equiv_i \bigcirc f \wedge_i \bigcirc g$
by *auto*

lemma *NextStateAndChop*:
$\vdash\ \bigcirc(((\mathit{init}\ w) \wedge_i f);\ g) \equiv_i \bigcirc (\mathit{init}\ w) \wedge_i \bigcirc(f;\ g)$
proof –
have *1*: $\vdash ((\mathit{init}\ w) \wedge_i f);\ g \equiv_i (\mathit{init}\ w) \wedge_i f;\ g$ **by** (*rule StateAndChop*)
hence *2*: $\vdash \bigcirc(((\mathit{init}\ w) \wedge_i f);\ g) \equiv_i \bigcirc((\mathit{init}\ w) \wedge_i f;\ g)$ **by** (*rule NextEqvNext*)
have *3*: $\vdash \bigcirc((\mathit{init}\ w) \wedge_i f;\ g) \equiv_i \bigcirc (\mathit{init}\ w) \wedge_i \bigcirc(f;\ g)$ **by** (*rule NextAndEqvNextAndNext*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *StateYieldsEqv*:
$\vdash\ ((\mathit{init}\ w) \supset_i (f\ \mathit{yields}\ \ g)) \equiv_i ((\mathit{init}\ w) \wedge_i f)\ \mathit{yields}\ \ g$
proof –
have *1*: $\vdash ((\mathit{init}\ w) \wedge_i f);\ \neg_i\ \ g \equiv_i (\mathit{init}\ w) \wedge_i f;\ (\neg_i\ \ g)$ **by** (*rule StateAndChop*)
hence *2*: $\vdash ((\mathit{init}\ w) \supset_i \neg_i (f;\ \neg_i\ \ g)) \equiv_i \neg_i (((\mathit{init}\ w) \wedge_i f);\ \neg_i\ \ g\ )$ **by** *auto*
from *2* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *StateAndDi*:
$\vdash\ (\mathit{init}\ w) \wedge_i\ \mathit{di}\ \ f \equiv_i\ \mathit{di}\ ((\mathit{init}\ w) \wedge_i f)$
proof –
have *1*: $\vdash ((\mathit{init}\ w) \wedge_i f);\ \mathit{true}_i \equiv_i\ (\mathit{init}\ w) \wedge_i f;\ \mathit{true}_i$ **by** (*rule StateAndChop*)
from *1* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiNext*:
$\vdash \quad di(\bigcirc f) \equiv_i \bigcirc (di\ \ f)$
proof −
have *1*: $\vdash (\bigcirc f);\ true_i \equiv_i \bigcirc(f;\ true_i)$ **by** (*rule NextChop*)
from *1* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiNextState*:
$\vdash \quad di(\bigcirc (init\ w)) \equiv_i \bigcirc (init\ w)$
proof −
have *1*: $\vdash\ di(\bigcirc (init\ w)) \equiv_i \bigcirc(di\ \ (init\ w))$ **by** (*rule DiNext*)
have *2*: $\vdash\ di\ \ (init\ w) \equiv_i (init\ w)$ **by** (*rule DiState*)
hence *3*: $\vdash \bigcirc(di\ \ (init\ w)) \equiv_i \bigcirc (init\ w)$ **by** (*rule NextEqvNext*)
from *1 3* **show** *?thesis* **by** *auto*
qed

lemma *StateImpBiGen*:
assumes $\vdash \quad (init\ w) \supset_i f$
shows $\vdash (init\ w) \supset_i bi\ \ f$
proof −
have *1*: $\vdash (init\ w) \supset_i f$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i\ \ f \supset_i \neg_i\ \ (init\ w)$ **by** *auto*
hence *3*: $\vdash\ di \neg_i\ \ f \supset_i\ di \neg_i\ \ (init\ w)$ **by** (*rule DiImpDi*)
hence *4*: $\vdash\ di \neg_i\ \ f \supset_i\ di\ \ (init \neg_i w)$ **by** *auto*
have *5*: $\vdash\ di\ (init \neg_i\ w) \equiv_i\ \ (init \neg_i\ w)$ **by** (*rule DiState*)
have *6*: $\vdash\ di \neg_i\ \ f \supset_i \neg_i\ \ (init\ w)$ **using** *4 5* **by** *auto*
hence *7*: $\vdash (init\ w) \supset_i \neg_i\ (\ di \neg_i\ \ f)$ **by** *auto*
from *7* **show** *?thesis* **by** (*simp add*: *bi-d-def*)
qed

lemma *ChopAndNotChopImp*:
$\vdash \quad f;\ g \wedge_i \neg_i (f;\ g1) \supset_i f;\ (g \wedge_i \neg_i\ \ g1)$
proof −
have *1*: $\vdash g \supset_i (g \wedge_i \neg_i\ \ g1) \vee_i\ \ g1$ **by** *auto*
hence *2*: $\vdash f;\ g \supset_i f;\ ((g \wedge_i \neg_i\ \ g1) \vee_i\ \ g1)$ **by** (*rule RightChopImpChop*)
have *3*: $\vdash f;\ ((g \wedge_i \neg_i\ \ g1) \vee_i\ \ g1) \supset_i (f;\ (g \wedge_i \neg_i\ \ g1)) \vee_i\ \ (f;\ g1)$ **by** (*rule ChopOrImp*)
have *4*: $\vdash f;\ g \supset_i f;\ (g \wedge_i \neg_i\ \ g1) \vee_i\ \ f;\ g1$ **using** *2 3 MP* **by** *auto*
from *4* **show** *?thesis* **by** *auto*
qed

lemma *ChopAndYieldsImp*:
$\vdash \quad f;\ g \wedge_i f\ yields\ \ g1 \supset_i f;\ (g \wedge_i g1)$
proof −
have *1*: $\vdash g \supset_i (g \wedge_i g1) \vee_i\ \ \neg_i\ \ g1$ **by** *auto*
hence *2*: $\vdash f;\ g \supset_i f;\ ((g \wedge_i\ \ g1) \vee_i\ \ \neg_i g1)$ **by** (*rule RightChopImpChop*)
have *3*: $\vdash f;\ ((g \wedge_i\ \ g1) \vee_i\ \ \neg_i g1) \supset_i (f;\ (g \wedge_i\ \ g1)) \vee_i\ \ (f;\ \neg_i g1)$ **by** (*rule ChopOrImp*)
have *4*: $\vdash f;\ g \supset_i f;\ (g \wedge_i\ \ g1) \vee_i\ \ f;\ \neg_i g1$ **using** *2 3 MP* **by** *auto*
hence *5*: $\vdash f;\ g \wedge_i \neg_i\ \ (f;\ \neg_i\ \ g1) \supset_i f;\ (g \wedge_i g1)$ **by** *auto*
from *5* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *ChopAndYieldsMP*:
$\vdash\ f; g \wedge_i f\ \mathit{yields}\ (g \supset_i g1) \supset_i f; g1$
proof –
have *1*: $\vdash f; g \wedge_i f\ \mathit{yields}\ (g \supset_i g1) \supset_i f; (g \wedge_i (g \supset_i g1))$ **by** (*rule ChopAndYieldsImp*)
have *2*: $\vdash g \wedge_i (g \supset_i g1) \supset_i g1$ **by** *auto*
hence *3*: $\vdash f; (g \wedge_i (g \supset_i g1)) \supset_i f; g1$ **by** (*rule RightChopImpChop*)
from *1 3* **show** *?thesis* **by** *auto*
qed

lemma *OrYieldsImp*:
$\vdash\ (f \vee_i f1)\ \mathit{yields}\ g \equiv_i (f\ \mathit{yields}\ g) \wedge_i (f1\ \mathit{yields}\ g)$
proof –
have *1*: $\vdash ((f \vee_i f1); \neg_i g) \equiv_i ((f; \neg_i g) \vee_i (f1; \neg_i g))$ **by** (*rule OrChopEqv*)
hence *2*: $\vdash \neg_i ((f \vee_i f1); \neg_i g) \equiv_i \neg_i (f; \neg_i g) \wedge_i \neg_i (f1; \neg_i g)$ **by** *auto*
from *2* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *LeftYieldsImpYields*:
assumes $\vdash f \supset_i f1$
shows $\vdash (f1\ \mathit{yields}\ g) \supset_i (f\ \mathit{yields}\ g)$
proof –
have *1*: $\vdash f \supset_i f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash f; \neg_i g \supset_i f1; \neg_i g$ **by** (*rule LeftChopImpChop*)
hence *3*: $\vdash \neg_i (f1; \neg_i g) \supset_i \neg_i (f; \neg_i g)$ **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *LeftYieldsEqvYields*:
assumes $\vdash f \equiv_i f1$
shows $\vdash (f\ \mathit{yields}\ g) \equiv_i (f1\ \mathit{yields}\ g)$
proof –
have *1*: $\vdash f \equiv_i f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash f; \neg_i g \equiv_i f1; \neg_i g$ **by** (*rule LeftChopEqvChop*)
hence *3*: $\vdash \neg_i (f; \neg_i g) \equiv_i \neg_i (f1; \neg_i g)$ **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

5.6 Properties of Fin

lemma *FinEqvTrueChopAndEmpty*:
$\vdash \mathit{fin}\ f \equiv_i \mathit{true}_i;(f \wedge_i \mathit{empty})$
proof –
have *1*: $\vdash \mathit{fin}\ f \equiv_i \Box(\mathit{empty} \supset_i f)$ **by** (*simp add*: *fin-d-def*)
have *2*: $\vdash \Box(\mathit{empty} \supset_i f) \equiv_i \neg_i(\Diamond(\neg_i(\mathit{empty} \supset_i f)))$ **by** (*simp add*: *always-d-def*)
have *3*: $\vdash (\neg_i(\mathit{empty} \supset_i f)) \equiv_i (\neg_i f \wedge_i \mathit{empty})$ **by** *auto*
hence *4*: $\vdash \Diamond(\neg_i(\mathit{empty} \supset_i f)) \equiv_i \Diamond(\neg_i f \wedge_i \mathit{empty})$ **using** *DiamondEqvDiamond* **by** *blast*
hence *5*: $\vdash \neg_i(\Diamond(\neg_i(\mathit{empty} \supset_i f))) \equiv_i \neg_i(\Diamond(\neg_i f \wedge_i \mathit{empty}))$ **by** *auto*
have *6*: $\vdash \neg_i(\Diamond(\neg_i f \wedge_i \mathit{empty})) \equiv_i \mathit{true}_i;(f \wedge_i \mathit{empty})$ **using** *Finprop* **by** *auto*
from *1 2 5 6* **show** *?thesis* **by** *auto*

qed

lemma *DiamondFin*:
$\vdash \Diamond(\mathit{fin}\ w) \equiv_i \mathit{fin}\ w$
by (*metis DiamondDiamondEqvDiamond DiamondEqvDiamond FinEqvTrueChopAndEmpty*
itl-prop(*30*) *prop03 sometimes-d-def*)

lemma *ChopFinExportA*:
$\vdash f;(g \wedge_i \mathit{fin}\ w) \supset_i \mathit{fin}\ w$
using *DiamondFin* **by** *auto*

lemma *FinImpBox*:
$\vdash \mathit{fin}\ w \supset_i \Box(\mathit{fin}\ w)$
by (*metis BoxImpBoxBox fin-d-def*)

lemma *FinAndChopImport*:
$\vdash (\mathit{fin}\ w) \wedge_i (f;g) \supset_i f;((\mathit{fin}\ w) \wedge_i g)$
proof −
have *1*: $\vdash \mathit{fin}\ w \supset_i \Box(\mathit{fin}\ w)$ **by** (*rule FinImpBox*)
hence *2*: $\vdash \mathit{fin}\ w \wedge_i f;g \supset_i \Box(\mathit{fin}\ w) \wedge_i (f;g)$ **by** *auto*
have *3*: $\vdash \Box(\mathit{fin}\ w) \wedge_i (f;g) \supset_i f;((\mathit{fin}\ w) \wedge_i g)$ **using** *BoxAndChopImport* **by** *blast*
from *2 3* **show** *?thesis* **using** *MP* **by** *auto*
qed

lemma *FinAndChop*:
$\vdash f;(g \wedge_i \mathit{fin}\ w) \equiv_i \mathit{fin}\ w \wedge_i f;g$
using *FinAndChopImport ChopFinExportA ChopAndA ChopAndCommute itl-prop*(*31*) *itl-prop*(*32*) *prop15*
by *blast*

lemma *ChopAndEmptyEqvEmptyChopEmpty*:
$\vdash ((f;g) \wedge_i \mathit{empty}) \equiv_i (f \wedge_i \mathit{empty});(g \wedge_i \mathit{empty})$
by *auto*

lemma *FinAndEmpty*:
$\vdash (\mathit{fin}\ w) \wedge_i \mathit{empty} \equiv_i w \wedge_i \mathit{empty}$
proof −
have *1*: $\vdash (\mathit{fin}\ w) \wedge_i \mathit{empty} \equiv_i \mathit{true}_i;(w \wedge_i \mathit{empty}) \wedge_i \mathit{empty}$
using *FinEqvTrueChopAndEmpty* **by** *auto*
have *2*: $\vdash \mathit{true}_i;(w \wedge_i \mathit{empty}) \wedge_i \mathit{empty} \equiv_i (\mathit{true}_i \wedge_i \mathit{empty});(w \wedge_i \mathit{empty})$
using *ChopAndEmptyEqvEmptyChopEmpty* **by** *auto*
have *3*: $\vdash (\mathit{true}_i \wedge_i \mathit{empty});(w \wedge_i \mathit{empty}) \equiv_i \mathit{empty};(w \wedge_i \mathit{empty})$
using *LeftChopEqvChop itl-prop*(*17*) **by** *blast*
have *4*: $\vdash \mathit{empty};(w \wedge_i \mathit{empty}) \equiv_i w \wedge_i \mathit{empty}$
using *EmptyChop* **by** *blast*
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *AndFinEqvChopAndEmpty*:
$\vdash f \wedge_i \mathit{fin}\ g \equiv_i f;(g \wedge_i \mathit{empty})$

proof −

have *1*: $\vdash f \wedge_i \mathit{fin}\ g \equiv_i f;\mathit{empty} \wedge_i \mathit{fin}\ g$ **using** *ChopEmpty itl-prop*(*30*) *prop06* **by** *blast*

have *2*: $\vdash \mathit{fin}\ g \wedge_i f;\mathit{empty} \equiv_i f;(\mathit{empty} \wedge_i \mathit{fin}\ g)$ **using** *FinAndChop* **using** *itl-prop*(*30*) **by** *blast*

have *3*: $\vdash \mathit{empty} \wedge_i \mathit{fin}\ g \equiv_i \mathit{fin}\ g \wedge_i \mathit{empty}$ **by** *auto*

have *4*: $\vdash \mathit{fin}\ g \wedge_i \mathit{empty} \equiv_i g \wedge_i \mathit{empty}$ **using** *FinAndEmpty* **by** *auto*

have *5*: $\vdash \mathit{empty} \wedge_i \mathit{fin}\ g \equiv_i g \wedge_i \mathit{empty}$ **using** *3 4* **by** *auto*

hence *6*: $\vdash f;(\mathit{empty} \wedge_i \mathit{fin}\ g) \equiv_i f;(g \wedge_i \mathit{empty})$ **using** *RightChopEqvChop* **by** *blast*

from *1 2 5* **show** *?thesis* **by** *auto*

qed

lemma *AndFinEqvChopStateAndEmpty*:

$\vdash f \wedge_i \mathit{fin}\ (\mathit{init}\ w) \equiv_i f; ((\mathit{init}\ w) \wedge_i \mathit{empty})$

using *AndFinEqvChopAndEmpty* **by** *blast*

lemma *FinStateEqvStateAndEmptyOrNextFinState*:

$\vdash \mathit{fin}\ (\mathit{init}\ w) \equiv_i ((\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i \bigcirc(\mathit{fin}\ (\mathit{init}\ w))$

proof −

have *1*: $\vdash \mathit{fin}\ (\mathit{init}\ w) \equiv_i \Box(\mathit{empty} \supset_i \mathit{init}\ w)$

by (*simp add*: *fin-d-def*)

have *2* : $\vdash \Box(\mathit{empty} \supset_i \mathit{init}\ w) \equiv_i$

$(\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \mathit{wnext}\ (\Box\ (\mathit{empty} \supset_i \mathit{init}\ w))$

by (*rule BoxEqvAndWnextBox*)

have *3*: $\vdash \mathit{fin}\ (\mathit{init}\ w) \equiv_i (\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \mathit{wnext}\ (\mathit{fin}\ (\mathit{init}\ w))$

using *1 2* **by** (*simp add*: *fin-d-def*)

have *4*: $\vdash \mathit{wnext}\ (\mathit{fin}\ (\mathit{init}\ w)) \equiv_i \mathit{empty} \vee_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w))$

by (*rule WnextEqvEmptyOrNext*)

have *5*: $\vdash \mathit{fin}\ (\mathit{init}\ w) \equiv_i (\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i (\mathit{empty} \vee_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w)))$

using *3 4* **by** (*simp add*: *fin-d-def*)

have *6*: $\vdash (\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i (\mathit{empty} \vee_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w))) \equiv_i$

$((\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i ((\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w)))$

by *auto*

have *7*: $\vdash (\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i (\mathit{init}\ w) \wedge_i \mathit{empty}$

by *auto*

have *8*: $\vdash (\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w)) \equiv_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w))$

by *auto*

have *9*: $\vdash ((\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i ((\mathit{empty} \supset_i \mathit{init}\ w) \wedge_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w))) \equiv_i$

$((\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i \bigcirc(\mathit{fin}\ (\mathit{init}\ w))$

using *7 8* **by** *auto*

from *5 6 9* **show** *?thesis* **using** *prop03* **by** *blast*

qed

lemma *FinChopEqvOr*:

$\vdash (\mathit{fin}\ (\mathit{init}\ w)); f \equiv_i ((\mathit{init}\ w) \wedge_i f) \vee_i \bigcirc((\mathit{fin}\ (\mathit{init}\ w)); f)$

proof −

have *1*: $\vdash \mathit{fin}\ (\mathit{init}\ w) \equiv_i ((\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i \bigcirc(\mathit{fin}\ (\mathit{init}\ w))$

by (*rule FinStateEqvStateAndEmptyOrNextFinState*)

hence *2*: $\vdash (\mathit{fin}\ (\mathit{init}\ w)); f \equiv_i (((\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i \bigcirc(\mathit{fin}\ (\mathit{init}\ w))); f$

by (*rule LeftChopEqvChop*)

have *3*: $\vdash (((\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i \bigcirc\ (\mathit{fin}\ (\mathit{init}\ w))); f$

$\equiv_i ((\mathit{init}\ w) \wedge_i \mathit{empty}); f \vee_i (\bigcirc\ (\mathit{fin}\ (\mathit{init}\ w))); f$

by (*rule OrChopEqv*)
have *4*: $\vdash ((init\ w) \wedge_i\ empty\ );\ f \equiv_i (init\ w) \wedge_i f$
by (*rule StateAndEmptyChop*)
have *5*: $\vdash (\bigcirc\ (fin\ \ (init\ w)));\ f \equiv_i \bigcirc((\ fin\ \ (init\ w));\ f)$
by (*rule NextChop*)
from *2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *FinChopEqvDiamond*:
$\vdash\ \ (\ fin\ \ (init\ w));\ f \equiv_i \Diamond\ ((init\ w) \wedge_i f)$
proof –
have *1*: $\vdash (\ fin\ \ (init\ w)) \equiv_i (true_i;((init\ w) \wedge_i empty))$
by (*rule FinEqvTrueChopAndEmpty*)
hence *2*: $\vdash (\ fin\ \ (init\ w));f \equiv_i (true_i;((init\ w) \wedge_i empty));f$
by (*rule LeftChopEqvChop*)
have *3*: $\vdash true_i;((\ (init\ w) \wedge_i empty);f)\ \equiv_i (true_i;((init\ w) \wedge_i empty));f$
by (*rule ChopAssoc*)
have *4*: $\vdash true_i;((\ (init\ w) \wedge_i empty);f) \equiv_i \Diamond\ (\ (\ (init\ w) \wedge_i empty);f)$
by (*simp add*: *sometimes-d-def*)
have *5*: $\vdash (\ (init\ w) \wedge_i empty);f \equiv_i (init\ w) \wedge_i f$
using *StateAndEmptyChop* **by** *blast*
hence *6*: $\vdash \Diamond\ (\ (\ (init\ w) \wedge_i empty);f) \equiv_i \Diamond\ (\ (init\ w) \wedge_i f)$
by (*rule DiamondEqvDiamond*)
from *2 3 4 6* **show** *?thesis* **by** *simp*
qed

lemma *NotDiamondAndNot*:
$\vdash \neg_i(\ \Diamond\ (\ f \wedge_i \neg_i f))$
by *simp*

lemma *FinYields*:
$\vdash\ \ (\ fin\ \ (init\ w))\ \ yields\ \ (init\ w)$
proof –
have *1*: $\vdash (fin\ (init\ w));\ \neg_i(init\ w) \equiv_i \Diamond((init\ w) \wedge_i \neg_i(init\ w))$ **by** (*rule FinChopEqvDiamond*)
have *2*: $\vdash\ \ \neg_i(\ \Diamond((init\ w) \wedge_i \neg_i\ \ (init\ w)))$ **by** (*rule NotDiamondAndNot*)
have *3*: $\vdash \neg_i\ ((\ fin\ \ (init\ w));\ \neg_i\ \ (init\ w))$ **using** *1 2* **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *ImpAndFinStateOrFinNotState*:
$\vdash f \supset_i (f \wedge_i fin\ (init\ w)) \vee_i fin\ \neg_i\ (init\ w)$
by (*simp*)

lemma *AndFinChopEqvStateAndChop*:
$\vdash\ \ (f \wedge_i\ \ fin\ \ (init\ w));\ g \equiv_i f;\ ((init\ w) \wedge_i g)$
proof –
have *1*: $\vdash (\ fin\ \ (init\ w))\ yields\ \ (init\ w)$
by (*rule FinYields*)
have *2*: $\vdash f \wedge_i\ \ fin\ \ (init\ w) \supset_i\ \ fin\ \ (init\ w)$
by *auto*

hence *3*: $\vdash (\mathit{fin}\ (\mathit{init}\ w))\ \mathit{yields}\ (\mathit{init}\ w) \supset_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w))\ \mathit{yields}\ (\mathit{init}\ w)$
by (*rule LeftYieldsImpYields*)
have *4*: $\vdash (f \wedge_i \mathit{fin}\ (\mathit{init}\ w))\ \mathit{yields}\ (\mathit{init}\ w)$
using *1 3 MP* **by** *auto*
have *5*: $\vdash (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); g \wedge_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w))\ \mathit{yields}\ (\mathit{init}\ w)$
$\supset_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); (g \wedge_i (\mathit{init}\ w))$
by (*rule ChopAndYieldsImp*)
have *6*: $\vdash (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); g \supset_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); (g \wedge_i (\mathit{init}\ w))$
using *4 5* **by** *auto*
have *7*: $\vdash (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); (g \wedge_i (\mathit{init}\ w)) \supset_i f; (g \wedge_i (\mathit{init}\ w))$
by (*rule AndChopA*)
have *8*: $\vdash g \wedge_i (\mathit{init}\ w) \supset_i (\mathit{init}\ w) \wedge_i g$
by *auto*
hence *9*: $\vdash f; (g \wedge_i (\mathit{init}\ w)) \supset_i f; ((\mathit{init}\ w) \wedge_i g)$
by (*rule RightChopImpChop*)
have *10*: $\vdash (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); g \supset_i f; ((\mathit{init}\ w) \wedge_i g)$
using *6 7 9* **by** *auto*
have *11*: $\vdash f \supset_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)) \vee_i \mathit{fin} \neg_i (\mathit{init}\ w)$
by (*rule ImpAndFinStateOrFinNotState*)
hence *12*: $\vdash f; ((\mathit{init}\ w) \wedge_i g) \supset_i$
$((f \wedge_i \mathit{fin}\ (\mathit{init}\ w)) \vee_i \mathit{fin} \neg_i (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g)$
by (*rule LeftChopImpChop*)
have *13*: $\vdash ((f \wedge_i \mathit{fin}\ (\mathit{init}\ w)) \vee_i \mathit{fin} \neg_i (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g)$
$\equiv_i$
$(f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g) \vee_i (\mathit{fin} \neg_i (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g)$
by (*rule OrChopEqv*)
have *14*: $\vdash (\mathit{fin}\ (\mathit{init}\ (\neg_i w))); ((\mathit{init}\ w) \wedge_i g) \supset_i \diamond((\mathit{init}\ (\neg_i w)) \wedge_i ((\mathit{init}\ w) \wedge_i g))$
using *FinChopEqvDiamond itl-prop*(*31*) **by** *blast*
have *141*: $\vdash \neg_i(\diamond((\mathit{init}\ (\neg_i w)) \wedge_i ((\mathit{init}\ w) \wedge_i g))) \supset_i$
$\neg_i ((\mathit{fin}\ (\mathit{init}\ (\neg_i w))); ((\mathit{init}\ w) \wedge_i g))$
using *14* **by** *auto*
have *15*: $\vdash \neg_i(\diamond((\mathit{init}\ (\neg_i w)) \wedge_i ((\mathit{init}\ w) \wedge_i g)))$
using *NotDiamondAndNot Initprop* **by** *simp*
have *151*: $\vdash \neg_i ((\mathit{fin}\ (\mathit{init}\ (\neg_i w))); ((\mathit{init}\ w) \wedge_i g))$
using *15 141* **by** *auto*
have *152*: $\vdash (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g) \vee_i (\mathit{fin} \neg_i (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g) \supset_i$
$(f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g)$
using *151* **by** *auto*
have *16*: $\vdash f; ((\mathit{init}\ w) \wedge_i g) \supset_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g)$
using *12 13 152* **by** *fastforce*
have *17*: $\vdash (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); ((\mathit{init}\ w) \wedge_i g) \supset_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); g$
by (*rule ChopAndB*)
have *18*: $\vdash f; ((\mathit{init}\ w) \wedge_i g) \supset_i (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)); g$
using *16 17* **by** *auto*
from *10 18* **show** *?thesis* **by** *auto*
qed

lemma *DiAndFinEqvChopState*:
$\vdash \mathit{di}\ (f \wedge_i \mathit{fin}\ (\mathit{init}\ w)) \equiv_i f; (\mathit{init}\ w)$
proof –

have *1*: $\vdash (f \wedge_i fin(init\ w));\ true_i \equiv_i f;((init\ w) \wedge_i true_i)$ **by** (*rule AndFinChopEqvStateAndChop*)
have *2*: $\vdash (init\ w) \wedge_i true_i \equiv_i (init\ w)$ **by** *auto*
hence *3*: $\vdash f;\ ((init\ w) \wedge_i true_i) \equiv_i f;\ (init\ w)$ **by** (*rule RightChopEqvChop*)
have *4*: $\vdash (f \wedge_i\ fin\ \ (init\ w));\ true_i \equiv_i f;\ (init\ w)$ **using** *1 3* **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *FinNotStateEqvNotFinState*:
$\vdash fin\ \ (init\ \neg_i\ w) \equiv_i \neg_i(\ fin\ \ (init\ w))$
by (*simp*)

lemma *BiImpFinEqvYieldsState*:
$\vdash\ \ bi\ (f \supset_i\ fin\ \ (init\ w)) \equiv_i f\ yields\ \ (init\ w)$
proof −
have *1*: $\vdash\ \ di\ (f \wedge_i fin\ (init\ \neg_i\ w)) \equiv_i f;\ \ \ (init\ \neg_i\ w)$ **by** (*rule DiAndFinEqvChopState*)
have *2*: $\vdash f \wedge_i fin(init\ \neg_i\ w)\ \equiv_i f \wedge_i \neg_i(fin(init\ w))$ **using** *FinNotStateEqvNotFinState* **by** *auto*
have *3*: $\vdash f \wedge_i \neg_i\ (fin(init\ w)) \equiv_i \neg_i\ (f \supset_i\ fin\ \ (init\ w))$ **by** *auto*
have *4*: $\vdash f \wedge_i fin(init\ \ \neg_i\ w) \equiv_i \neg_i\ (f \supset_i\ fin(init\ w))$ **using** *2 3* **by**(*simp add*: *fin-d-def*)
hence *5*: $\vdash\ \ di\ (f \wedge_i\ fin\ \ \ (init\ \neg_i\ w)) \equiv_i\ di\ \neg_i\ (f \supset_i fin(init\ w))$ **by** (*rule DiEqvDi*)
have *6*: $\vdash\ \ di\ \neg_i\ (f \supset_i\ fin\ \ (init\ w)) \equiv_i \neg_i(\ bi\ (f \supset_i fin(init\ w)))$ **by** (*rule DiNotEqvNotBi*)
have *7*: $\vdash \neg_i\ (bi\ (f \supset_i\ fin\ \ (init\ w))) \equiv_i f;(init\ \neg_i\ w)$ **using** *1 5 6 Initprop* **by** *auto*
hence *8*: $\vdash bi\ (f \supset_i\ fin\ \ (init\ w)) \equiv_i \neg_i\ (f;\ \neg_i\ \ (init\ w))$ **by** *auto*
from *8* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *StateImpYields*:
assumes $\vdash\ \ (init\ w) \wedge_i f \supset_i\ fin\ \ (init\ w1)$
shows $\vdash (init\ w) \supset_i (f\ yields\ \ (init\ w1))$
proof −
have *1*: $\vdash (init\ w) \wedge_i f \supset_i\ fin\ \ (init\ w1)$ **using** *assms* **by** *auto*
hence *2*: $\vdash (init\ w) \supset_i (f \supset_i\ fin\ \ (init\ w1))$ **by** *auto*
hence *3*: $\vdash (init\ w) \supset_i bi\ (f \supset_i\ fin\ \ (init\ w1))$ **by** (*rule StateImpBiGen*)
have *4*: $\vdash bi\ (f \supset_i\ fin\ \ (init\ w1)) \equiv_i f\ yields\ \ (init\ w1)$ **by** (*rule BiImpFinEqvYieldsState*)
from *3 4* **show** *?thesis* **by** *auto*
qed

lemma *StateAndYieldsImpYields*:
assumes $\vdash\ \ (init\ w) \wedge_i f \supset_i f1$
shows $\vdash (init\ w) \wedge_i (f1\ yields\ \ g) \supset_i (f\ yields\ \ g)$
proof −
have *1*: $\vdash (init\ w) \wedge_i f \supset_i f1$ **using** *assms* **by** *auto*
hence *2*: $\vdash (init\ w) \wedge_i (f;\ \neg_i\ \ g) \supset_i f1;\ \neg_i\ \ g$ **by** (*rule StateAndChopImpChopRule*)
hence *3*: $\vdash (init\ w) \wedge_i \neg_i\ (f1;\ \neg_i\ \ g) \supset_i \neg_i\ (f;\ \neg_i\ \ g)$ **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *AndYieldsA*:
$\vdash\ \ f\ yields\ \ g \supset_i (f \wedge_i f1)\ yields\ \ g$
proof −
have *1*: $\vdash f \wedge_i f1 \supset_i f$ **by** *auto*

```
 from 1 show ?thesis by (rule LeftYieldsImpYields)
qed

lemma AndYieldsB:
 ⊢  f1 yields  g ⊃_i (f ∧_i f1) yields  g
proof –
 have 1: ⊢ f ∧_i f1 ⊃_i f1  by auto
 from 1 show ?thesis by (rule LeftYieldsImpYields)
qed

lemma RightYieldsImpYields:
 assumes ⊢   g⊃_i g1
 shows   ⊢ (f yields  g) ⊃_i (f yields  g1)
proof –
 have  1: ⊢ g ⊃_i g1  using assms by auto
 hence 2:  ⊢ ¬_i  g1 ⊃_i ¬_i  g by auto
 hence 3: ⊢ f; ¬_i  g1 ⊃_i f; ¬_i  g  by (rule RightChopImpChop)
 hence 4: ⊢ ¬_i (f; ¬_i  g) ⊃_i ¬_i (f; ¬_i  g1)  by auto
 from 4 show ?thesis by (simp add: yields-d-def)
qed

lemma RightYieldsEqvYields:
 assumes ⊢   g≡_i g1
 shows   ⊢ (f yields  g) ≡_i (f yields  g1)
proof –
 have  1: ⊢ g ≡_i g1  using assms by auto
 hence 2: ⊢ ¬_i  g ≡_i ¬_i  g1  by auto
 hence 3: ⊢ f; ¬_i  g ≡_i f; ¬_i  g1  by (rule RightChopEqvChop)
 hence 4: ⊢ ¬_i (f; ¬_i  g) ≡_i ¬_i (f; ¬_i  g1)  by auto
 from 4 show ?thesis by (simp add: yields-d-def)
qed

lemma BoxImpYields:
⊢   □ g ⊃_i f yields  g
proof –
 have  1: ⊢ f; ¬_i  g ⊃_i ◇¬_i  g  by (rule ChopImpDiamond)
 hence 2: ⊢ ¬_i (◇¬_i  g) ⊃_i ¬_i (f; ¬_i  g)  by auto
 from 2 show ?thesis by (simp add: yields-d-def)
qed

lemma BoxEqvTrueYields:
⊢   □ f ≡_i true_i yields  f
proof –
 have  1: ⊢ true_i; ¬_i  f ≡_i ◇ ¬_i  f  by (rule TrueChopEqvDiamond)
 hence 2: ⊢ ¬_i (true_i; ¬_i  f) ≡_i ¬_i( ◇ ¬_i  f)  by auto
 have  3: ⊢ □ f ≡_i ¬_i ( ◇ ¬_i  f)  by (simp add: always-d-def)
 have  4: ⊢ □ f ≡_i ¬_i (true_i; ¬_i  f)  using 2 3 by auto
 from 4 show ?thesis by (simp add: yields-d-def)
qed
```

lemma *YieldsGen*:
assumes $\vdash\ g$
shows $\vdash f\ \mathit{yields}\ g$
proof −
have *1*: $\vdash g$ **using** *assms* **by** *auto*
hence *2*: $\vdash \Box\ g$ **by** (*rule BoxGen*)
have *3*: $\vdash \Box\ g \supset_i f\ \mathit{yields}\ g$ **by** (*rule BoxImpYields*)
from *2 3* **show** *?thesis* **using** *MP* **by** *auto*
qed

lemma *YieldsAndYieldsEqvYieldsAnd*:
$\vdash (f\ \mathit{yields}\ g) \wedge_i (f\ \mathit{yields}\ g1) \equiv_i f\ \mathit{yields}\ (g \wedge_i g1)$
proof −
have *1*: $\vdash f; (\neg_i\ g \vee_i \neg_i\ g1) \equiv_i (f; \neg_i\ g) \vee_i (f; \neg_i\ g1)$ **by** (*rule ChopOrEqv*)
hence *2*: $\vdash (f; \neg_i\ g) \vee_i (f; \neg_i\ g1) \equiv_i f; (\neg_i\ g \vee_i \neg_i\ g1)$ **by** *auto*
have *3*: $\vdash \neg_i\ g \vee_i \neg_i\ g1 \equiv_i \neg_i (g \wedge_i g1)$ **by** *auto*
hence *4*: $\vdash f; (\neg_i\ g \vee_i \neg_i\ g1) \equiv_i f; \neg_i (g \wedge_i g1)$ **by** (*rule RightChopEqvChop*)
have *5*: $\vdash (f; \neg_i\ g) \vee_i (f; \neg_i\ g1) \equiv_i f; \neg_i (g \wedge_i g1)$ **using** *2 4* **by** *auto*
hence *6*: $\vdash \neg_i (f; \neg_i\ g) \wedge_i \neg_i (f; \neg_i\ g1) \equiv_i \neg_i (f; \neg_i (g \wedge_i g1))$ **by** *auto*
from *6* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *YieldsAndYieldsImpAndYieldsAnd*:
$\vdash (f\ \mathit{yields}\ g) \wedge_i (f1\ \mathit{yields}\ g1) \supset_i (f \wedge_i f1)\ \mathit{yields}\ (g \wedge_i g1)$
proof −
have *1*: $\vdash f\ \mathit{yields}\ g \supset_i (f \wedge_i f1)\ \mathit{yields}\ g$
by (*rule AndYieldsA*)
have *2*: $\vdash f1\ \mathit{yields}\ g1 \supset_i (f \wedge_i f1)\ \mathit{yields}\ g1$
by (*rule AndYieldsB*)
have *3*: $\vdash (f \wedge_i f1)\ \mathit{yields}\ g \wedge_i (f \wedge_i f1)\ \mathit{yields}\ g1 \equiv_i (f \wedge_i f1)\ \mathit{yields}\ (g \wedge_i g1)$
by (*rule YieldsAndYieldsEqvYieldsAnd*)
from *1 2 3* **show** *?thesis* **by** *auto*
qed

lemma *YieldsYieldsEqvChopYields*:
$\vdash f\ \mathit{yields}\ (g\ \mathit{yields}\ h) \equiv_i (f; g)\ \mathit{yields}\ h$
proof −
have *1*: $\vdash f; (g; \neg_i\ h) \equiv_i (f; g); \neg_i\ h$ **by** (*rule ChopAssoc*)
hence *2*: $\vdash f; (g; \neg_i\ h) \equiv_i (f; g); \neg_i\ h$ **by** *auto*
have *3*: $\vdash g; \neg_i\ h \equiv_i \neg_i \neg_i (g; \neg_i\ h)$ **by** *auto*
hence *4*: $\vdash f; (g; \neg_i\ h) \equiv_i f; \neg_i \neg_i (g; \neg_i\ h)$ **by** (*rule RightChopEqvChop*)
have *5*: $\vdash f; \neg_i \neg_i (g; \neg_i\ h) \equiv_i (f; g); \neg_i\ h$ **using** *2 4* **by** *auto*
hence *6*: $\vdash f; \neg_i (g\ \mathit{yields}\ h) \equiv_i (f; g); \neg_i\ h$ **by** (*simp add*: *yields-d-def*)
hence *7*: $\vdash \neg_i (f; \neg_i (g\ \mathit{yields}\ h)) \equiv_i \neg_i ((f; g); \neg_i\ h)$ **by** *auto*
from *7* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *EmptyYields*:
$\vdash \mathit{empty}\ \mathit{yields}\ f \equiv_i f$
proof −

have *1*: $\vdash$ *empty* ; $\neg_i$ $f \equiv_i \neg_i$ f **by** (*rule EmptyChop*)
hence *2*: $\vdash \neg_i$ (*empty* ; $\neg_i$ f) $\equiv_i f$ **by** *auto*
from *2* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *NextYields*:
$\vdash$ $(\bigcirc f)$ *yields* $g \equiv_i$ *wnext* (f *yields* g)
proof –
have *1*: $\vdash (\bigcirc f); \neg_i\ g \equiv_i \bigcirc(f; \neg_i\ g)$ **by** (*rule NextChop*)
hence *2*: $\vdash \neg_i\ ((\bigcirc f); \neg_i\ g) \equiv_i \neg_i\ (\bigcirc(f; \neg_i\ g))$ **by** *auto*
hence *3*: $\vdash (\bigcirc f)$ *yields* $g \equiv_i \neg_i\ (\bigcirc(f; \neg_i\ g))$ **by** (*simp add*: *yields-d-def*)
have *4*: $\vdash \neg_i(\bigcirc(f; \neg_i\ g)) \equiv_i$ *wnext* $\neg_i\ (f; \neg_i\ g)$ **by** *auto*
have *5*: $\vdash (\bigcirc f)$ *yields* $g \equiv_i$ *wnext* $\neg_i\ (f; \neg_i\ g)$ **using** *3 4* **by** *auto*
from *5* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *SkipChopEqvNext*:
$\vdash$ *skip* ; $f \equiv_i \bigcirc f$
by (*simp add*: *next-d-def*)

lemma *SkipYieldsEqvWeakNext*:
$\vdash$ *skip* *yields* $f \equiv_i$ *wnext* f
proof –
have *1*: $\vdash$ *skip* ; $\neg_i\ f \equiv_i \bigcirc\neg_i\ f$ **by** (*rule SkipChopEqvNext*)
hence *2*: $\vdash \neg_i$ (*skip* ; $\neg_i\ f$) $\equiv_i \neg_i(\bigcirc\neg_i\ f)$ **by** *auto*
have *3*: $\vdash \neg_i\ (\bigcirc\neg_i\ f) \equiv_i$ *wnext* f **by** *auto*
have *4*: $\vdash \neg_i$ (*skip* ; $\neg_i\ f$) $\equiv_i$ *wnext* f **using** *2 3* **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *NextImpSkipYields*:
$\vdash$ $\bigcirc f \supset_i$ *skip* *yields* f
proof –
have *1*: $\vdash \bigcirc f \supset_i$ *wnext* f **by** *auto*
have *2*: $\vdash$ *skip* *yields* $f \equiv_i$ *wnext* f **by** (*rule SkipYieldsEqvWeakNext*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *MoreEqvSkipChopTrue*:
$\vdash$ *more* $\equiv_i$ *skip* ; $true_i$
proof –
have *1*: $\vdash$ *skip* ; $true_i \equiv_i \bigcirc true_i$ **by** (*rule SkipChopEqvNext*)
hence *2*: $\vdash \bigcirc true_i \equiv_i$ *skip* ; $true_i$ **by** *auto*
from *2* **show** *?thesis* **by** (*simp add*: *more-d-def*)
qed

lemma *MoreChopImpMore*:
$\vdash$ *more* ; $f \supset_i$ *more*
proof –
have *1*: $\vdash (\bigcirc true_i); f \equiv_i \bigcirc(true_i; f)$ **by** (*rule NextChop*)

have *2*: $\vdash \bigcirc(true_i;\ f) \supset_i\ more$ **by** *auto*
have *3*: $\vdash (\bigcirc true_i;\ f) \supset_i\ more$ **using** *1 2* **by** *auto*
from *3* **show** *?thesis* **by** (*metis more-d-def*)
qed

lemma *ChopMoreImpMore*:
$\vdash\ f;\ more\ \supset_i\ more$
proof –
have *1*: $\vdash f;\ more\ \supset_i \Diamond\ more$ **by** (*rule ChopImpDiamond*)
have *2*: $\vdash \Diamond\ more\ \supset_i\ more$ **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *MoreChopEqvNextDiamond*:
$\vdash\ \ more\ ;\ f \equiv_i \bigcirc(\Diamond\ f)$
proof –
have *1*: $\vdash\ more\ ;\ f \equiv_i (\bigcirc\ true_i);\ f$ **by** (*simp add*: *more-d-def*)
have *2*: $\vdash (\bigcirc true_i);\ f \equiv_i \bigcirc(true_i;\ f)$ **by** (*rule NextChop*)
have *3*: $\vdash\ more\ ;\ f \equiv_i \bigcirc(true_i;\ f)$ **using** *1 2* **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *sometimes-d-def*)
qed

lemma *WeakNextBoxImpMoreYields*:
$\vdash\ \ more\ \ yields\ \ f \equiv_i wnext(\ \Box\ f)$
proof –
have *1*: $\vdash more\ ;\ \neg_i\ f \equiv_i \bigcirc(\Diamond\ \neg_i f)$ **by** (*rule MoreChopEqvNextDiamond*)
have *2*: $\vdash \bigcirc(\Diamond\ \neg_i f) \equiv_i \bigcirc(\neg_i(\Box f))$ **by** *auto*
have *3*: $\vdash \bigcirc(\neg_i(\Box f)) \equiv_i \neg_i\ (\ wnext(\ \Box\ f)\ )$ **by** *auto*
have *4*: $\vdash more\ ;\ \neg_i\ f\ \equiv_i \neg_i(more\ \ yields\ \ f)$ **by** (*metis itl-prop*(*30*) *itl-prop*(*4*) *yields-d-def*)
from *1 2 3 4* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *NotEqvYieldsMore*:
$\vdash\ \ \neg_i\ \ f \equiv_i f\ yields\ \ more$
proof –
have *1*: $\vdash f;\ \ empty\ \ \equiv_i f$ **by** (*rule ChopEmpty*)
hence *2*: $\vdash \neg_i\ (f;\ \ empty\ ) \equiv_i \neg_i\ \ f$ **by** *auto*
have *3*: $\vdash\ \ empty\ \ \equiv_i \neg_i\ \ more$ **by** *auto*
hence *4*: $\vdash f;\ \ empty\ \ \equiv_i f;\ \neg_i\ \ more$ **by** (*rule RightChopEqvChop*)
hence *5*: $\vdash \neg_i\ (f;\ \ empty\ ) \equiv_i \neg_i\ (f;\ \neg_i\ \ more\ )$ **by** *auto*
have *6*: $\vdash \neg_i\ \ f \equiv_i \neg_i\ (f;\ \neg_i\ \ more\ )$ **using** *2 5* **by** *auto*
from *6* **show** *?thesis* **by** (*metis yields-d-def*)
qed

lemma *LeftChopImpMoreRule*:
assumes $\vdash\ \ f \supset_i\ more$
shows $\vdash f;\ g \supset_i\ more$
proof –
have *1*: $\vdash f \supset_i\ more$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;\ g \supset_i\ more\ ;\ g$ **by** (*rule LeftChopImpChop*)

have *3*: $\vdash$ *more* ; $g \supset_i$ *more* **by** (*rule MoreChopImpMore*)
from *2 3* **show** *?thesis* **using** *prop02* **by** *blast*
qed

lemma *RightChopImpMoreRule*:
assumes $\vdash$ $g \supset_i$ *more*
shows $\vdash$ $f; g \supset_i$ *more*
proof –
have *1*: $\vdash g \supset_i$ *more* **using** *assms* **by** *auto*
hence *2*: $\vdash f; g \supset_i f;$ *more* **by** (*rule RightChopImpChop*)
have *3*: $\vdash f;$ *more* $\supset_i$ *more* **by** (*rule ChopMoreImpMore*)
from *2 3* **show** *?thesis* **using** *prop02* **by** *blast*
qed

lemma *NotDiEqvBiNot*:
$\vdash$ $\neg_i$ (*di* f) $\equiv_i$ *bi* ($\neg_i$ f)
proof –
have *1*: $\vdash f \equiv_i \neg_i \neg_i f$ **by** *auto*
hence *2*: $\vdash$ *di* $f \equiv_i$ *di* $\neg_i \neg_i f$ **by** (*rule DiEqvDi*)
hence *3*: $\vdash \neg_i$ (*di* f) $\equiv_i \neg_i$ (*di* $\neg_i \neg_i f$) **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *bi-d-def*)
qed

lemma *ChopImpDi*:
$\vdash$ $f; g \supset_i$ *di* f
proof –
have *1*: $\vdash g \supset_i$ *true*$_i$ **by** *auto*
hence *2*: $\vdash f; g \supset_i f;$ *true*$_i$ **by** (*rule RightChopImpChop*)
from *2* **show** *?thesis* **by** (*simp add*: *bi-d-def*)
qed

lemma *TrueEqvTrueChopTrue*:
$\vdash$ *true*$_i$ $\equiv_i$ *true*$_i$; *true*$_i$
proof –
have *1*: $\vdash$ *true*$_i$; *true*$_i$ $\supset_i$ *true*$_i$ **by** *auto*
have *2*: $\vdash$ *true*$_i$ $\supset_i$ *di true*$_i$ **by** (*rule DiIntro*)
hence *3*: $\vdash$ *true*$_i$ $\supset_i$ *true*$_i$; *true*$_i$ **by** (*simp add*: *di-d-def*)
from *1 3* **show** *?thesis* **by** *auto*
qed

lemma *DiEqvDiDi*:
$\vdash$ *di* $f \equiv_i$ *di* (*di* f)
proof –
have *1*: $\vdash$ *true*$_i$ $\equiv_i$ *true*$_i$; *true*$_i$ **by** (*rule TrueEqvTrueChopTrue*)
hence *2*: $\vdash f;$ *true*$_i$ $\equiv_i f;$ (*true*$_i$; *true*$_i$) **by** (*rule RightChopEqvChop*)
have *3*: $\vdash f;$ (*true*$_i$; *true*$_i$)$\equiv_i$ (f; *true*$_i$); *true*$_i$ **by** (*rule ChopAssoc*)
have *4*: $\vdash f;$ *true*$_i$ $\equiv_i$ (f; *true*$_i$); *true*$_i$ **using** *2 3* **using** *prop03* **by** *blast*
from *4* **show** *?thesis* **by** (*metis di-d-def*)
qed

lemma *BiEqvBiBi*:
$\vdash\ \ bi\ \ f \equiv_i bi(\ bi\ \ f)$
proof −
have *1*: $\vdash\ \ di\ \neg_i\ \ f \equiv_i\ \ di(\ \ di\ \neg_i\ \ f)$ **by** (*rule DiEqvDiDi*)
have *2*: $\vdash\ \ di\ \neg_i\ \ f \equiv_i \neg_i\ (\ bi\ \ f)$ **by** (*rule DiNotEqvNotBi*)
hence *3*: $\vdash\ \ di\ (\ di\ \neg_i\ \ f) \equiv_i\ \ di\ \neg_i\ (\ bi\ \ f)$ **by** (*rule DiEqvDi*)
have *4*: $\vdash\ \ di\ \neg_i\ \ f \equiv_i\ \ di\ \neg_i(\ \ bi\ \ f)$ **using** *1 3* **using** *prop03* **by** *blast*
hence *5*: $\vdash \neg_i\ (\ di\ \neg_i\ \ f) \equiv_i \neg_i\ (\ di\ \neg_i\ (\ bi\ \ f))$ **using** *itl-prop*(*33*) **by** *blast*
from *5* **show** *?thesis* **by** (*metis bi-d-def*)
qed

lemma *DiOrEqv*:
$\vdash\ \ \ di\ (f \vee_i\ \ g) \equiv_i\ \ di\ \ f \vee_i\ \ \ di\ \ g$
proof −
have *1*: $\vdash (f \vee_i\ \ g);\ true_i \equiv_i f;\ true_i \vee_i\ \ g;\ true_i$ **by** (*rule OrChopEqv*)
from *1* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiAndA*:
$\vdash\ \ \ di\ (f \wedge_i g) \supset_i\ \ di\ \ f$
proof −
have *1*: $\vdash (f \wedge_i g);\ true_i \supset_i f;\ true_i$ **by** (*rule AndChopA*)
from *1* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiAndB*:
$\vdash\ \ \ di\ (f \wedge_i g) \supset_i\ \ di\ \ g$
proof −
have *1*: $\vdash (f \wedge_i g);\ true_i \supset_i g;\ true_i$ **by** (*rule AndChopB*)
from *1* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiAndImpAnd*:
$\vdash\ \ \ di\ (f \wedge_i g) \supset_i\ \ di\ \ f \wedge_i\ \ di\ \ g$
proof −
have *1*: $\vdash\ \ di\ (f \wedge_i g) \supset_i\ \ di\ \ f$ **by** (*rule DiAndA*)
have *2*: $\vdash\ \ di\ (f \wedge_i g) \supset_i\ \ di\ \ g$ **by** (*rule DiAndB*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *DiSkipEqvMore*:
$\vdash\ \ \ di\ \ skip\ \ \equiv_i\ \ more$
proof −
have *1*: $\vdash\ \ skip\ ;\ true_i \equiv_i \bigcirc true_i$ **by** (*rule SkipChopEqvNext*)
have *2*: $\vdash \bigcirc true_i \equiv_i\ \ more$ **by** *auto*
have *3*: $\vdash\ \ skip\ ;\ true_i \equiv_i\ \ more$ **using** *1 2* **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiMoreEqvMore*:

$\vdash$ *di more* $\equiv_i$ *more*
proof –
have *1*: $\vdash$ *di* ($\bigcirc$ *true*$_i$) $\equiv_i$ $\bigcirc$(*di true*$_i$) **by** (*rule DiNext*)
have *2*: $\vdash$ $\bigcirc$(*di true*$_i$) $\supset_i$ *more* **by** *auto*
have *3*: $\vdash$ *di*($\bigcirc$ *true*$_i$) $\supset_i$ *more* **using** *1 2* **by** *auto*
hence *4*: $\vdash$ *di more* $\supset_i$ *more* **by** (*simp add*: *more-d-def*)
have *5*: $\vdash$ *more* $\supset_i$ *di more* **by** (*rule ImpDi*)
from *4 5* **show** *?thesis* **by** *auto*
qed

lemma *DiIfEqvRule*:
assumes $\vdash$ *f* $\equiv_i$ *if*$_i$ (*init w*) *then g else h*
shows $\vdash$ *di f* $\equiv_i$ *if*$_i$ (*init w*) *then* (*di g*) *else* (*di h*)
proof –
have *1*: $\vdash$ *f* $\equiv_i$ *if*$_i$ (*init w*) *then g else h* **using** *assms* **by** *auto*
hence *2*: $\vdash$ *f*; *true*$_i$ $\equiv_i$ *if*$_i$ (*init w*) *then* (*g*; *true*$_i$) *else* (*h*; *true*$_i$) **by** (*rule IfChopEqvRule*)
from *2* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DiEmpty*:
$\vdash$ *di empty*
proof –
have *1*: $\vdash$ *true*$_i$ **by** *auto*
have *2*: $\vdash$ *empty* ; *true*$_i$ $\equiv_i$ *true*$_i$ **by** (*rule EmptyChop*)
have *3*: $\vdash$ *empty* ; *true*$_i$ **using** *1 2* **by** *auto*
from *3* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *DaNotEqvNotBa*:
$\vdash$ *da* $\neg_i$ *f* $\equiv_i$ $\neg_i$ (*ba f*)
proof –
have *1*: $\vdash$ *ba f* $\equiv_i$ $\neg_i$ (*da* $\neg_i$ *f*) **by** (*simp add*: *ba-d-def*)
from *1* **show** *?thesis* **by** *simp*
qed

lemma *DaEqvDa*:
assumes $\vdash$ *f* $\equiv_i$ *g*
shows $\vdash$ *da f* $\equiv_i$ *da g*
using *assms* **by** *auto*

lemma *DaEqvNotBaNot*:
$\vdash$ *da f* $\equiv_i$ $\neg_i$ (*ba* $\neg_i$ *f*)
proof –
have *1*: $\vdash$ *ba* $\neg_i$ *f* $\equiv_i$ $\neg_i$ (*da* $\neg_i$ $\neg_i$ *f*) **by** (*simp add*: *ba-d-def*)
hence *2*: $\vdash$ *da* $\neg_i$ $\neg_i$ *f* $\equiv_i$ $\neg_i$(*ba* $\neg_i$ *f*) **by** *simp*
have *3*: $\vdash$ *f* $\equiv_i$ $\neg_i$ $\neg_i$ *f* **by** *simp*
hence *4*: $\vdash$ *da f* $\equiv_i$ *da* $\neg_i$ $\neg_i$ *f* **by** (*rule DaEqvDa*)
from *2 4* **show** *?thesis* **by** *simp*
qed

```
lemma BaElim:
⊢  ba f ⊃i f
proof –
 have 1: ⊢ ba f ≡i □(bi f) by (rule BaEqvBtBi)
 have 2: ⊢ bi f ⊃i f by (rule BiElim)
 hence 3: ⊢ □(bi f ⊃i f) by (rule BoxGen)
 have 4: ⊢ □(bi f ⊃i f) ⊃i □(bi f) ⊃i □ f by (rule BoxImpDist)
 have 5: ⊢ □(bi f) ⊃i □ f using 3 4 MP by simp
 have 6: ⊢ □ f ⊃i f by (rule BoxElim)
 from 1 5 6 show ?thesis using BaImpBt prop02 by blast
qed

lemma DaIntro:
⊢  f ⊃i da f
proof –
 have 1: ⊢ ba ¬i f ⊃i ¬i f by (rule BaElim)
 hence 2: ⊢ ¬i ¬i f ⊃i ¬i ( ba ¬i f) using prop27 by blast
 have 3: ⊢ f ≡i ¬i ¬i f by simp
 have 4: ⊢ da f ≡i ¬i ( ba ¬i f) by (rule DaEqvNotBaNot)
 from 2 3 4 show ?thesis by simp
qed

lemma BaGen:
 assumes ⊢  f
 shows ⊢ ba f
proof –
 have 1: ⊢  f using assms by auto
 hence 2: ⊢ □ f by (rule BoxGen)
 hence 3: ⊢ bi( □ f) by (rule BiGen)
 have 4: ⊢ ba f ≡i bi (□ f) by (rule BaEqvBiBt)
 from 3 4 show ?thesis by simp
qed

lemma BaImpDist:
⊢ ba (f ⊃i g) ⊃i ba f ⊃i ba g
proof –
 have 1: ⊢ bi (f ⊃i g) ⊃i (bi f ⊃i bi g) by (rule BiImpDist)
 hence 2: ⊢ □(bi (f ⊃i g) ⊃i (bi f ⊃i bi g)) by (rule BoxGen)
 have 3: ⊢ □(bi (f ⊃i g) ⊃i (bi f ⊃i bi g))
          ⊃i
          (□ (bi (f ⊃i g)) ⊃i (□(bi f) ⊃i □(bi g))) by simp
 have 4: ⊢ □(bi (f ⊃i g)) ⊃i (□(bi f) ⊃i □(bi g)) using 2 3 MP by simp
 have 5: ⊢ ba (f⊃i g) ≡i □(bi (f ⊃i g)) by (rule BaEqvBtBi)
 have 6: ⊢ ba f ≡i □(bi f) by (rule BaEqvBtBi)
 have 7: ⊢ ba g ≡i □(bi g) by (rule BaEqvBtBi)
 from 4 5 6 7 show ?thesis by simp
qed

lemma BaAndEqv:
```

$\vdash\ \mathit{ba}\ (f \wedge_i g) \equiv_i\ \mathit{ba}\ \ f \wedge_i\ \mathit{ba}\ \ g$
proof −
have *1*: $\vdash\ \mathit{ba}\ (f \wedge_i g) \equiv_i\ \Box(\mathit{bi}\ (f \wedge_i g))$ **by** (*rule BaEqvBtBi*)
have *2*: $\vdash\ \mathit{bi}\ (f \wedge_i g) \equiv_i \mathit{bi}\ f \wedge_i \mathit{bi}\ g$ **by** *auto*
hence *3*: $\vdash \Box(\mathit{bi}\ (f \wedge_i g)) \equiv_i \Box(\mathit{bi}\ f \wedge_i \mathit{bi}\ g)$ **by** *auto*
have *4*: $\vdash \Box(\mathit{bi}\ f \wedge_i \mathit{bi}\ g) \equiv_i \Box(\mathit{bi}\ f)\ \wedge_i \Box(\mathit{bi}\ g)$ **by** *auto*
have *5*: $\vdash \mathit{ba}\ \ f \equiv_i \Box(\mathit{bi}\ \ f)$ **by** (*rule BaEqvBtBi*)
have *6*: $\vdash\ \mathit{ba}\ \ g \equiv_i \Box(\mathit{bi}\ \ g)$ **by** (*rule BaEqvBtBi*)
from *1 3 4 5 6* **show** *?thesis* **by** *auto*
qed

lemma *BaImpBaEqvBa*:
$\vdash\ \mathit{ba}\ (f \equiv_i g) \supset_i\ (\mathit{ba}\ \ f \equiv_i\ \mathit{ba}\ \ g)$
proof −
have *1*: $\vdash\ \mathit{ba}\ (f \supset_i g) \supset_i\ \mathit{ba}\ \ f \supset_i\ \mathit{ba}\ \ g$ **by** (*rule BaImpDist*)
have *2*: $\vdash\ \mathit{ba}\ (g \supset_i f) \supset_i\ \mathit{ba}\ \ g \supset_i\ \mathit{ba}\ \ f$ **by** (*rule BaImpDist*)
have *3*: $\vdash \mathit{ba}\ (f \equiv_i g) \equiv_i \mathit{ba}\ ((f \supset_i g) \wedge_i (g \supset_i f))$ **by** *auto*
have *4*: $\vdash \mathit{ba}\ ((f \supset_i g) \wedge_i (g \supset_i f)) \equiv_i \mathit{ba}((f \supset_i g)) \wedge_i \mathit{ba}((g \supset_i f))$ **by** (*rule BaAndEqv*)
have *5*: $\vdash (\mathit{ba}\ \ f \supset_i\ \mathit{ba}\ \ g) \wedge_i (\mathit{ba}\ \ g \supset_i\ \mathit{ba}\ \ f) \equiv_i (\mathit{ba}\ \ f \equiv_i\ \mathit{ba}\ \ g)$ **by** *auto*
from *1 2 3 4 5* **show** *?thesis* **using** *itl-prop*(*31*) *itl-prop*(*32*) *prop02* **by** *smt*
qed

lemma *BaImpBa*:
assumes $\vdash\ \ f \supset_i g$
shows $\vdash \mathit{ba}\ \ f \supset_i\ \mathit{ba}\ \ g$
using *BaGen BaImpDist MP assms* **by** *blast*

lemma *BaEqvBa*:
assumes $\vdash\ \ f \equiv_i g$
shows $\vdash\ \mathit{ba}\ \ f \equiv_i\ \mathit{ba}\ \ g$
using *BaGen BaImpBaEqvBa MP assms* **by** *blast*

lemma *DaImpDa*:
assumes $\vdash\ \ f \supset_i g$
shows $\vdash\ \mathit{da}\ \ f \supset_i\ \mathit{da}\ \ g$
using *assms* **by** *fastforce*

lemma *DiamondEqvDiamondDiamond*:
$\vdash \Diamond\ f \equiv_i \Diamond\ (\Diamond\ f)$
proof −
have *1*: $\vdash \Diamond\ (\Diamond\ f) \equiv_i \mathit{true}_i;(\mathit{true}_i;f)$
by *simp*
have *2*: $\vdash \mathit{true}_i;(\mathit{true}_i;f)\ \equiv_i (\mathit{true}_i;\mathit{true}_i);f$
by (*rule ChopAssoc*)

have *3*: $\vdash (\mathit{true}_i;\mathit{true}_i);f \equiv_i \mathit{true}_i;f$
using *LeftChopEqvChop TrueEqvTrueChopTrue itl-prop*(*30*) **by** *blast*
have *4*: $\vdash \mathit{true}_i;f \equiv_i \Diamond f$
by (*simp add*: *sometimes-d-def*)
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *DaEqvDaDa*:
$\vdash \mathit{da}\ f \equiv_i \mathit{da}\ (\mathit{da}\ f)$
proof –
have *1*: $\vdash \mathit{da}\ f \equiv_i \Diamond(\mathit{di}\ f)$
by (*rule DaEqvDtDi*)
have *2*: $\vdash \mathit{di}\ f \equiv_i (\mathit{di}\ (\mathit{di}\ f))$
by (*rule DiEqvDiDi*)
hence *3*: $\vdash \Diamond\ (\mathit{di}\ f) \equiv_i \Diamond\ (\mathit{di}\ (\mathit{di}\ f))$
by (*rule DiamondEqvDiamond*)
have *4*: $\vdash \Diamond\ (\mathit{di}\ f) \equiv_i \Diamond(\Diamond\ (\mathit{di}\ (\mathit{di}\ f)))$
using *DiamondEqvDiamondDiamond DiEqvDiDi* **using** *3 prop03* **by** *blast*
have *5*: $\vdash \Diamond\ (\mathit{di}\ (\mathit{di}\ f)) \equiv_i \mathit{di}\ (\Diamond\ (\mathit{di}\ f))$
by (*rule DtDiEqvDiDt*)
hence *6*: $\vdash \Diamond(\Diamond\ (\mathit{di}\ (\mathit{di}\ f))) \equiv_i \Diamond\ (\mathit{di}\ (\Diamond\ (\mathit{di}\ f)))$
by (*rule DiamondEqvDiamond*)
have *7*: $\vdash \mathit{da}\ f \equiv_i \Diamond\ (\mathit{di}(\ \Diamond\ (\mathit{di}\ f)))$
using *1 3 4 6* **using** *prop03* **by** *blast*
have *8*: $\vdash \mathit{da}\ (\Diamond\ (\mathit{di}\ f)) \equiv_i \Diamond(\mathit{di}\ (\Diamond\ (\mathit{di}\ f)))$
by (*rule DaEqvDtDi*)
have *9*: $\vdash \mathit{da}\ (\mathit{da}\ f) \equiv_i \mathit{da}\ (\Diamond\ (\mathit{di}\ f))$
using *1* **by** (*rule DaEqvDa*)
from *7 8 9* **show** *?thesis* **by** *auto*
qed

lemma *BaEqvBaBa*:
$\vdash \mathit{ba}\ f \equiv_i \mathit{ba}\ (\mathit{ba}\ f)$
proof –
have *1*: $\vdash \mathit{da}\ (\neg_i\ f) \equiv_i \mathit{da}\ (\mathit{da}\ (\neg_i\ f))$ **by** (*rule DaEqvDaDa*)
have *2*: $\vdash \mathit{da}\ (\mathit{da}\ (\neg_i\ f)) \equiv_i \neg_i\ (\mathit{ba}\ (\neg_i\ (\mathit{da}\ (\neg_i\ f))))$ **by** (*rule DaEqvNotBaNot*)
have *3*: $\vdash \neg_i\ (\mathit{da}\ (\mathit{da}\ (\neg_i\ f))) \equiv_i \mathit{ba}\ (\neg_i\ (\mathit{da}\ (\neg_i\ f)))$ **by** *auto*
have *4*: $\vdash \neg_i\ (\mathit{da}\ (\neg_i\ f)) \equiv_i \mathit{ba}\ (\neg_i\ (\mathit{da}\ (\neg_i\ f)))$ **using** *1 2 3 prop01 prop03* **by** *blast*
from *4* **show** *?thesis* **by** (*metis ba-d-def*)
qed

lemma *BaLeftChopImpChop*:
$\vdash \mathit{ba}\ (f \supset_i f1) \supset_i f;\ g \supset_i f1;\ g$
proof –
have *1*: $\vdash \mathit{ba}\ (f \supset_i f1) \supset_i \mathit{bi}\ (f \supset_i f1)$ **by** (*rule BaImpBi*)
have *2*: $\vdash \mathit{bi}\ (f \supset_i f1) \supset_i f;\ g \supset_i f1;\ g$ **by** (*rule BiChopImpChop*)
from *1 2* **show** *?thesis* **by** *auto*

qed

lemma *BaRightChopImpChop*:
$\vdash\ ba\ (g \supset_i g1) \supset_i f;\ g \supset_i f;\ g1$
proof −
have *1*: $\vdash\ ba\ (g \supset_i g1) \supset_i \Box(g \supset_i g1)$ **by** (*rule BaImpBt*)
have *2*: $\vdash \Box(g \supset_i g1) \supset_i f;\ g \supset_i f;\ g1$ **by** (*rule BoxChopImpChop*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *ChopAndBaImport*:
$\vdash\ (f;\ f1) \wedge_i\ ba\ g \supset_i (f \wedge_i g);\ (f1 \wedge_i g)$
proof −
have *1*: $\vdash\ ba\ g \wedge_i (f;\ f1) \supset_i (g \wedge_i f);\ (g \wedge_i f1)$ **by** (*rule BaAndChopImport*)
have *2*: $\vdash (g \wedge_i f);\ (g \wedge_i f1) \equiv_i (f \wedge_i g);\ (f1 \wedge_i g)$ **by** (*rule AndChopAndCommute*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *BaImpBaImpBaAnd*:
$\vdash ba\ h \supset_i ba(g \supset_i ba\ h \wedge_i g)$
proof −
have *1*: $\vdash ba\ h \supset_i (g \supset_i ba\ h \wedge_i g)$ **by** *simp*
hence *2*: $\vdash ba(ba\ h) \supset_i ba(g \supset_i ba\ h \wedge_i g)$ **by** (*rule BaImpBa*)
have *3*: $\vdash ba\ h \equiv_i ba(ba\ h)$ **by** (*rule BaEqvBaBa*)
from *2 3* **show** *?thesis* **using** *itl-prop*(*31*) *prop02* **by** *blast*
qed

lemma *BaChopImpChopBa*:
$\vdash\ ba\ f \supset_i g;\ g1 \supset_i g;\ ((ba\ f) \wedge_i g1)$
proof −
have *1*: $\vdash\ ba\ f \supset_i\ ba\ (g1 \supset_i (ba\ f) \wedge_i g1)$ **by** (*rule BaImpBaImpBaAnd*)
have *2*: $\vdash\ ba\ (g1 \supset_i\ ba\ f \wedge_i g1) \supset_i g;\ g1 \supset_i g;\ (ba\ f \wedge_i g1)$ **by** (*rule BaRightChopImpChop*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *DiNotBaImpNotBa*:
$\vdash\ di\ \neg_i\ (ba\ f) \supset_i \neg_i\ (ba\ f)$
proof −
have *1*: $\vdash\ ba\ f \equiv_i\ ba(\ ba\ f)$ **by** (*rule BaEqvBaBa*)
have *2*: $\vdash\ ba\ (ba\ f) \supset_i bi\ (ba\ f)$ **by** (*rule BaImpBi*)
have *3*: $\vdash\ ba\ f \supset_i bi\ (ba\ f)$ **using** *1 2* **using** *itl-prop*(*31*) *prop02* **by** *blast*
hence *4*: $\vdash\ ba\ f \supset_i \neg_i\ (di\ \neg_i\ (ba\ f))$ **by** (*simp add*: *bi-d-def*)
from *4* **show** *?thesis* **by** *fastforce*
qed

lemma *NotBaChopImpNotBa*:
$\vdash\ (\neg_i\ (ba\ f));\ g \supset_i \neg_i\ (ba\ f)$
proof −
have *1*: $\vdash (\neg_i\ (ba\ f));\ g \supset_i\ di\ \neg_i\ (ba\ f)$ **by** (*rule ChopImpDi*)

have *2*: $\vdash\ di\ \neg_i\ (\ ba\ \ f) \supset_i \neg_i\ (\ ba\ \ f)$ **by** (*rule DiNotBaImpNotBa*)
from *1 2* **show** *?thesis* **using** *prop02* **by** *blast*
qed

lemma *DiamondFinImpFin*:
$\vdash \Diamond\ (fin\ f) \supset_i fin\ f$
proof –
have *1*: $\vdash fin\ f \equiv_i true_i;(f \wedge_i empty)$
by (*rule FinEqvTrueChopAndEmpty*)
hence *2*: $\vdash \Diamond\ (fin\ f) \equiv_i true_i;(true_i;(f \wedge_i empty))$
by *fastforce*
have *3*: $\vdash true_i;(true_i;(f \wedge_i empty)) \equiv_i (true_i;true_i);(f \wedge_i empty)$
by (*rule ChopAssoc*)
have *4*: $\vdash (true_i;true_i);(f \wedge_i empty) \equiv_i true_i;(f \wedge_i empty)$
using *TrueEqvTrueChopTrue* **using** *LeftChopEqvChop itl-prop*(*30*) **by** *blast*
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *ChopFinImpFin*:
$\vdash\ f;\ fin\ (init\ w) \supset_i fin\ (init\ w)$
proof –
have *1*: $\vdash f;\ fin\ (init\ w) \supset_i \Diamond\ (\ fin\ (init\ w))$ **by** (*rule ChopImpDiamond*)
have *2*: $\vdash \Diamond\ (fin\ (init\ w)) \supset_i fin\ (init\ w)$ **by** (*rule DiamondFinImpFin*)
from *1 2* **show** *?thesis* **using** *prop02* **by** *blast*
qed

lemma *FinImpYieldsFin*:
$\vdash\ fin\ (init\ w) \supset_i f\ yields\ (\ fin\ (init\ w))$
proof –
have *1*: $\vdash f;\ fin\ (init\ \neg_i\ w) \supset_i fin\ (init\ \neg_i\ w)$
by (*rule ChopFinImpFin*)
have *2*: $\vdash\ fin\ (init\ \neg_i\ w) \equiv_i \neg_i\ (\ fin\ (init\ w))$
using *FinNotStateEqvNotFinState* **by** *blast*
hence *3*: $\vdash f;\ fin\ (init\ \neg_i\ w) \equiv_i f;\ \neg_i\ (\ fin\ (init\ w))$
by (*rule RightChopEqvChop*)
have *4*: $\vdash f;\ \neg_i\ (\ fin\ (init\ w)) \supset_i \neg_i\ (\ fin\ (init\ w))$
using *1 2 3 itl-prop*(*31*) *prop02* **by** *blast*
hence *5*: $\vdash\ fin\ (init\ w) \supset_i \neg_i\ (f;\ \neg_i\ (\ fin\ (init\ w)))$
using *itl-prop*(*35*) **by** *auto*
from *5* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *ChopAndFin*:
$\vdash\ (f;\ g) \wedge_i fin\ (init\ w) \equiv_i f;\ (g \wedge_i fin\ (init\ w))$
proof –
have *1*: $\vdash\ fin\ (init\ w) \supset_i f\ yields\ (\ fin\ (init\ w))$
by (*rule FinImpYieldsFin*)
hence *2*: $\vdash (f;\ g) \wedge_i fin\ (init\ w) \supset_i (f;\ g) \wedge_i f\ yields\ (\ fin\ (init\ w))$

```
      by auto
 have  3: ⊢ (f; g) ∧_i f yields ( fin (init w)) ⊃_i f; (g ∧_i  fin (init w))
      by (rule ChopAndYieldsImp)
 have  4: ⊢ (f; g) ∧_i  fin (init w) ⊃_i f; (g ∧_i  fin (init w))
      using 2 3 by auto
 have 11: ⊢ f; (g ∧_i  fin (init w)) ⊃_i f; g
     by (rule ChopAndA)
 have 12: ⊢ f; (g ∧_i  fin (init w)) ⊃_i f;  fin (init w)
     by (rule ChopAndB)
 have 13: ⊢ f;  fin (init w) ⊃_i ◇ ( fin (init w))
     by (rule ChopImpDiamond)
 have 14: ⊢ ◇( fin (init w)) ⊃_i  fin (init w)
     by (rule DiamondFinImpFin)
 have 15: ⊢ f; (g ∧_i  fin (init w)) ⊃_i (f; g) ∧_i  fin (init w)
     using 11 12 13 14 itl-prop(32) prop02 by smt
 from 4 15 show ?thesis using itl-prop(31) by blast
qed

lemma ChopAndNotFin:
  ⊢   f; g ∧_i ¬_i ( fin (init w)) ≡_i f; (g ∧_i ¬_i ( fin (init w)))
proof −
 have  1: ⊢ f; g ∧_i  fin  (init ¬_i w) ≡_i f; (g ∧_i  fin  (init ¬_i w))
      by (rule ChopAndFin)
 have  2: ⊢  fin  (init ¬_i w) ≡_i ¬_i  ( fin (init w) )
      using FinNotStateEqvNotFinState by blast
 hence 3: ⊢ g ∧_i  fin  (init ¬_i w) ≡_i g ∧_i ¬_i(  fin (init w))
      by auto
 hence 4: ⊢ f; (g ∧_i  fin   (init ¬_i w)) ≡_i f; (g ∧_i ¬_i ( fin (init w)))
      by (rule RightChopEqvChop)
 from 1 2 4 show ?thesis by auto
qed

lemma FinChopChain:
⊢  ((init w)⊃_i  fin (init w1)); ((init w1) ⊃_i  fin (init w2))
      ⊃_i ((init w) ⊃_i  fin (init w2))
proof −
 have  1: ⊢  (init w) ∧_i  ((init w)⊃_i  fin (init w1)); ((init w1) ⊃_i  fin (init w2))
          ⊃_i
          ( (init w) ∧_i ((init w)⊃_i fin (init w1))); ((init w1)⊃_i  fin (init w2))
      by (rule StateAndChopImport)
 have  2: ⊢ (init w) ∧_i ((init w) ⊃_i  fin (init w1)) ⊃_i  fin (init w1)
      by auto
 have  3: ⊢ ((init w) ∧_i ((init w)⊃_i  fin (init w1))); ((init w1) ⊃_i  fin (init w2))
          ⊃_i
          ( fin (init w1)); ((init w1) ⊃_i  fin (init w2))
      using 2 by (rule LeftChopImpChop)
 have  4: ⊢  ( fin (init w1)); ((init w1) ⊃_i  fin (init w2)) ≡_i
           ◇((init w1) ∧_i ((init w1) ⊃_i  fin (init w2)))
     by (rule FinChopEqvDiamond)
 have 41: ⊢ ((init w1) ∧_i ((init w1) ⊃_i  fin (init w2))) ⊃_i fin (init w2)
```

```
    by auto
 have 42: ⊢ ◇((init w1) ∧ᵢ ((init w1) ⊃ᵢ fin (init w2))) ⊃ᵢ ◇ ( fin (init w2))
    using 41 DiamondImpDiamond by blast
 have 5: ⊢ ◇( fin( init w2)) ⊃ᵢ fin (init w2)
    using DiamondFinImpFin by blast
 have 6: ⊢ (init w) ∧ᵢ ((init w)⊃ᵢ fin (init w1)); ((init w1) ⊃ᵢ fin (init w2))
          ⊃ᵢ fin (init w2)
    using 1 3 4 5 42 itl-prop(30) prop02 prop15 by smt
 from 6 show ?thesis using prop32 by blast
qed

lemma ChopRule:
 assumes ⊢ (init w) ∧ᵢ f ⊃ᵢ fin (init w1)
         ⊢ (init w1)∧ᵢ f1 ⊃ᵢ fin (init w2)
 shows ⊢ (init w) ∧ᵢ (f; f1) ⊃ᵢ fin (init w2)
proof −
 have 1: ⊢ (init w) ∧ᵢ (f; f1) ⊃ᵢ ((init w) ∧ᵢ f); f1 by (rule StateAndChopImport)
 have 2: ⊢ (init w) ∧ᵢ f ⊃ᵢ fin (init w1) using assms by auto
 hence 3: ⊢ ((init w) ∧ᵢ f); f1 ⊃ᵢ ( fin (init w1)); f1 by (rule LeftChopImpChop)
 have 4: ⊢ ( fin (init w1)); f1 ≡ᵢ ◇((init w1) ∧ᵢ f1) by (rule FinChopEqvDiamond)
 have 5: ⊢ (init w1) ∧ᵢ f1 ⊃ᵢ fin (init w2) using assms by auto
 hence 6: ⊢ ◇((init w1) ∧ᵢ f1) ⊃ᵢ ◇ (fin (init w2)) by (rule DiamondImpDiamond)
 have 7: ⊢ ◇( fin (init w2)) ⊃ᵢ fin (init w2) using DiamondFinImpFin by blast
 from 1 3 4 6 7 show ?thesis using itl-prop(30) prop02 prop15 by smt
qed


lemma ChopRep:
 assumes ⊢ (init w) ∧ᵢ f ⊃ᵢ f1 ∧ᵢ fin (init w1)
         ⊢ (init w1) ∧ᵢ g ⊃ᵢ g1
 shows ⊢ (init w) ∧ᵢ (f; g) ⊃ᵢ (f1; g1)
proof −
 have 1: ⊢ (init w) ∧ᵢ f ⊃ᵢ f1 ∧ᵢ fin (init w1) using assms by auto
 hence 2: ⊢ (init w) ∧ᵢ (f; g) ⊃ᵢ (f1 ∧ᵢ fin (init w1)); g by (rule StateAndChopImpChopRule)
 have 3: ⊢ (f1 ∧ᵢ fin (init w1)); g ≡ᵢ f1; ((init w1) ∧ᵢ g) by (rule AndFinChopEqvStateAndChop)
 have 4: ⊢ (init w1)∧ᵢ g ⊃ᵢ g1 using assms by auto
 hence 5: ⊢ f1; ((init w1) ∧ᵢ g) ⊃ᵢ f1; g1 by (rule RightChopImpChop)
 from 2 3 5 show ?thesis by simp
qed

lemma ChopRepAndFin:
 assumes ⊢ (init w) ∧ᵢ f ⊃ᵢ f1 ∧ᵢ fin (init w1)
         ⊢ (init w1) ∧ᵢ g ⊃ᵢ g1 ∧ᵢ fin (init w2)
 shows ⊢ (init w) ∧ᵢ (f; g) ⊃ᵢ (f1; g1) ∧ᵢ fin (init w2)
proof −
 have 1: ⊢ (init w) ∧ᵢ f ⊃ᵢ f1 ∧ᵢ fin (init w1) using assms by auto
 have 2: ⊢ (init w1) ∧ᵢ g ⊃ᵢ g1 ∧ᵢ fin (init w2) using assms by auto
 have 3: ⊢ (init w) ∧ᵢ (f; g) ⊃ᵢ f1; (g1 ∧ᵢ fin (init w2)) using 1 2 by (rule ChopRep)
 have 4: ⊢ f1; (g1 ∧ᵢ fin (init w2)) ⊃ᵢ f1; g1 by (rule ChopAndA)
 have 5: ⊢ f1; (g1 ∧ᵢ fin (init w2)) ⊃ᵢ f1; fin (init w2) by (rule ChopAndB)
```

have *6*: $\vdash f1;\ fin\ \ (init\ w2) \supset_i\ fin\ \ (init\ w2)$ **by** (*rule ChopFinImpFin*)
from *1 2 3 4 5 6* **show** *?thesis* **using** *ChopRep ChopRule itl-prop*(*32*) **by** *blast*
qed

lemma *TrueChopMoreEqvMore*:
$\vdash true_i\ ;\ more \equiv_i more$
by *auto*

lemma *MoreChopLoop*:
assumes $\vdash\ \ f \supset_i\ more\ ;\ f$
shows $\vdash \neg_i\ f$
proof −
have *1*: $\vdash f \supset_i\ more\ ;\ f$
using *assms* **by** *auto*
hence *11*: $\vdash \Diamond\ f \supset_i \Diamond\ (more;f)$
by (*rule DiamondImpDiamond*)
have *12*: $\vdash \Diamond\ (more;f) \equiv_i true_i;(more;f)$
by *simp*
have *13*: $\vdash true_i;(more;f) \equiv_i (true_i;more);f$
by (*rule ChopAssoc*)
have *14*: $\vdash\ \Diamond\ (more;f) \equiv_i more;f$
using *TrueChopMoreEqvMore 12 13 LeftChopChopImpChopRule NowImpDiamond*
itl-prop(*31*) *prop02* **by** *metis*
have *2*: $\vdash\ more\ ;\ f \equiv_i \bigcirc(\Diamond\ f)$
by (*rule MoreChopEqvNextDiamond*)
have *3*: $\vdash\ \Diamond\ f \supset_i \bigcirc(\Diamond\ f)$
using *11 14 2* **by** *auto*
hence *4*: $\vdash \neg_i\ (\Diamond\ f)$
by (*rule NextLoop*)
have *5*: $\vdash \neg_i\ (\Diamond\ f) \supset_i \neg_i\ f$
by *auto*
from *4 5* **show** *?thesis* **using** *MP* **by** *blast*
qed

lemma *MoreChopContra*:
assumes $\vdash\ \ f \wedge_i \neg_i\ g \supset_i (\ more\ ;\ (f \wedge_i \neg_i\ g))$
shows $\vdash f \supset_i g$
proof −
have *1*: $\vdash f \wedge_i \neg_i\ g \supset_i (\ more\ ;\ (f \wedge_i \neg_i\ g))$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i\ (f \wedge_i \neg_i\ g)$ **by** (*rule MoreChopLoop*)
from *2* **show** *?thesis* **by** *auto*
qed

lemma *ChopLoop*:
assumes $\vdash\ \ f \supset_i g;f$
$\vdash g \supset_i\ more$
shows $\vdash \neg_i\ f$
proof −

have *1*: $\vdash f \supset_i g; f$ **using** *assms* **by** *auto*
have *2*: $\vdash g \supset_i$ *more* **using** *assms* **by** *auto*
hence *3*: $\vdash g; f \supset_i$ *more* ; f **by** (*rule LeftChopImpChop*)
have *4*: $\vdash f \supset_i$ *more* ; f **using** *1 3* **by** *auto*
from *4* **show** *?thesis* **using** *MoreChopLoop* **by** *auto*
qed

lemma *ChopContra*:
assumes $\vdash f \wedge_i \neg_i g \supset_i h; f \wedge_i \neg_i (h; g)$
$\vdash h \supset_i$ *more*
shows $\vdash f \supset_i g$
proof –
have *1*: $\vdash f \wedge_i \neg_i g \supset_i h; f \wedge_i \neg_i (h; g)$ **using** *assms* **by** *auto*
have *2*: $\vdash h \supset_i$ *more* **using** *assms* **by** *auto*
have *3*: $\vdash h; f \wedge_i \neg_i (h; g) \supset_i h; (f \wedge_i \neg_i g)$ **by** (*rule ChopAndNotChopImp*)
have *4*: $\vdash h; (f \wedge_i \neg_i g) \supset_i$ *more* ; $(f \wedge_i \neg_i g)$ **using** *2* **by** (*rule LeftChopImpChop*)
have *5*: $\vdash f \wedge_i \neg_i g \supset_i$ *more* ; $(f \wedge_i \neg_i g)$ **using** *1 3 4* **by** *auto*
from *5* **show** *?thesis* **using** *MoreChopContra* **by** *auto*
qed

5.7 Properties of Chopstar and Chopplus

lemma *EmptyImpCS*:
$\vdash$ *empty* $\supset_i f^\star$
proof –
have *1*: $\vdash f^\star \equiv_i$ *empty* $\vee_i (f \wedge_i$ *more*$);f^\star$ **by** (*rule ChopstarEqv*)
have *2*: $\vdash$ *empty* $\supset_i$ *empty* $\vee_i (f \wedge_i$ *more*$);f^\star$ **by** *auto*
from *1 2* **show** *?thesis* **using** *itl-prop*(*31*) *prop02* **by** *blast*
qed

lemma *CSEqvOrChopCS*:
$\vdash f^\star \equiv_i$ *empty* $\vee_i (f; f^\star)$
proof –
have *1*: $\vdash f^\star \equiv_i$ *empty* $\vee_i (f \wedge_i$ *more*$);f^\star$ **by** (*rule ChopstarEqv*)
have *2*: $\vdash (f \wedge_i$ *more*$);f^\star \supset_i f;f^\star$ **by** (*rule AndChopA*)
have *3*: $\vdash f^\star \supset_i$ *empty* $\vee_i f; f^\star$ **using** *1 2* **using** *prop14* **by** *blast*
have *4*: $\vdash$ *empty* $\supset_i f^\star$ **by** (*rule EmptyImpCS*)
have *5*: $\vdash f \supset_i$ *empty* $\vee_i (f \wedge_i$ *more*$)$ **by** *auto*
have *6*: $\vdash f; f^\star \supset_i f^\star \vee_i (f \wedge_i$ *more* $); f^\star$ **using** *5* **by** (*rule EmptyOrChopImpRule*)
have *7*: $\vdash f^\star \supset_i$ *empty* $\vee_i (f \wedge_i$ *more*$);f^\star$ **using** *1* **using** *itl-prop*(*31*) **by** *blast*
have *8*: $\vdash f; f^\star \supset_i$ *empty* $\vee_i (f \wedge_i$ *more* $); f^\star$ **using** *6 7 prop16* **by** *blast*
hence *9*: $\vdash f; f^\star \supset_i f^\star$ **using** *1 prop15* **by** *blast*
have *10*: $\vdash$ *empty* $\vee_i f; f^\star \supset_i f^\star$ **using** *9 4 prop30* **by** *blast*
from *3 10* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *CSAndMoreEqvAndMoreChop*:
$\vdash f^\star \wedge_i$ *more* $\equiv_i (f \wedge_i$ *more* $); f^\star$
proof –
have *1*: $\vdash ($ *empty* $\vee_i (f \wedge_i$ *more* $); f^\star) \wedge_i$ *more* $\supset_i (f \wedge_i$ *more* $); f^\star$ **by** *auto*

have *2*: $\vdash f^{\star} \equiv_i \mathit{empty} \vee_i (f \wedge_i \mathit{more}); f^{\star}$ **by** (*rule ChopstarEqv*)
have *3*: $\vdash f^{\star} \wedge_i \mathit{more} \supset_i (f \wedge_i \mathit{more}); f^{\star}$ **using** *1 2* **using** *prop18* **by** *blast*
have *4*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \supset_i f^{\star}$ **using** *2 prop19* **by** *blast*
have *5*: $\vdash (f \wedge_i \mathit{more}) \supset_i \mathit{more}$ **by** *auto*
hence *6*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \supset_i \mathit{more}$ **by** (*rule LeftChopImpMoreRule*)
have *7*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \supset_i f^{\star} \wedge_i \mathit{more}$ **using** *4 6* **using** *itl-prop*(*32*) **by** *blast*
from *3 7* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *CSAndMoreImpChopCS*:
$\vdash f^{\star} \wedge_i \mathit{more} \supset_i f; f^{\star}$
proof −
have *1*: $\vdash f^{\star} \wedge_i \mathit{more} \equiv_i (f \wedge_i \mathit{more}); f^{\star}$ **by** (*rule CSAndMoreEqvAndMoreChop*)
have *2*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \supset_i f; f^{\star}$ **by** (*rule AndChopA*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *NotAndMoreEqvEmptyOr*:
$\vdash \neg_i (f \wedge_i \mathit{more}) \equiv_i (\mathit{empty} \vee_i \neg_i f)$
by *auto*

lemma *MoreAndEmptyOrEqvMoreAnd*:
$\vdash \mathit{more} \wedge_i (\mathit{empty} \vee_i \neg_i f) \equiv_i \mathit{more} \wedge_i \neg_i f$
by *auto*

lemma *CSMoreNotImpChopCSAndMore*:
$\vdash f^{\star} \wedge_i \mathit{more} \wedge_i \neg_i f \supset_i (f \wedge_i \mathit{more}); (f^{\star} \wedge_i \mathit{more})$
proof −
have *1*: $\vdash f^{\star} \wedge_i \mathit{more} \equiv_i (f \wedge_i \mathit{more}); f^{\star}$
by (*rule CSAndMoreEqvAndMoreChop*)
have *2*: $\vdash \mathit{empty} \vee_i \mathit{more}$
by *auto*
hence *3*: $\vdash f^{\star} \supset_i \mathit{empty} \vee_i (f^{\star} \wedge_i \mathit{more})$
by *auto*
hence *4*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \supset_i (f \wedge_i \mathit{more}) \vee_i ((f \wedge_i \mathit{more}); (f^{\star} \wedge_i \mathit{more}))$
by (*rule ChopEmptyOrImpRule*)
hence *5*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \wedge_i \neg_i (f \wedge_i \mathit{more}) \supset_i ((f \wedge_i \mathit{more}); (f^{\star} \wedge_i \mathit{more}))$
using *prop29* **by** *blast*
have *6*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \equiv_i (f \wedge_i \mathit{more}); f^{\star} \wedge_i \mathit{more}$ **using** *1*
by *auto*
have *7*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \wedge_i \neg_i (f \wedge_i \mathit{more}) \equiv_i (f \wedge_i \mathit{more}); f^{\star} \wedge_i \mathit{more} \wedge_i \neg_i (f \wedge_i \mathit{more})$
using *6* **by** *auto*
have *8*: $\vdash (f \wedge_i \mathit{more}); f^{\star} \wedge_i \mathit{more} \wedge_i \neg_i f \supset_i (f \wedge_i \mathit{more}); (f^{\star} \wedge_i \mathit{more})$
using *5 7* **by** *auto*
have *9*: $\vdash f^{\star} \wedge_i \mathit{more} \wedge_i \neg_i f \equiv_i (f^{\star} \wedge_i \mathit{more}) \wedge_i (\mathit{more} \wedge_i \neg_i f)$
by *auto*
have *10*: $\vdash (f^{\star} \wedge_i \mathit{more}) \wedge_i (\mathit{more} \wedge_i \neg_i f) \equiv_i (f \wedge_i \mathit{more}); f^{\star} \wedge_i (\mathit{more} \wedge_i \neg_i f)$
using *1 prop06* **by** *auto*
from *1 8 9 10* **show** *?thesis* **by** *auto*
qed

```
lemma CSAndMoreImpCSChop:
⊢  f⋆ ∧ᵢ  more  ⊃ᵢ f⋆; f
proof −
 have  1: ⊢ f⋆ ∧ᵢ  more  ≡ᵢ (f ∧ᵢ  more ); f⋆
      by (rule CSAndMoreEqvAndMoreChop)
 have  2: ⊢  empty  ∨ᵢ   more
      by auto
 hence 3: ⊢ f⋆ ⊃ᵢ  empty  ∨ᵢ  (f⋆ ∧ᵢ  more )
      by auto
 hence 4: ⊢ (f ∧ᵢ  more ); f⋆ ⊃ᵢ
           (f ∧ᵢ  more ) ∨ᵢ  ((f ∧ᵢ  more ); (f⋆∧ᵢ more ))
      by (rule ChopEmptyOrImpRule)
 have  5: ⊢ f⋆ ∧ᵢ  more  ∧ᵢ ¬ᵢ  f ⊃ᵢ (f ∧ᵢ  more ); (f⋆∧ᵢ  more )
      by (rule CSMoreNotImpChopCSAndMore)
 have  6: ⊢ f⋆ ≡ᵢ  empty  ∨ᵢ  (f ∧ᵢ  more ); f⋆
      by (rule ChopstarEqv)
 hence 7: ⊢ f⋆; f ≡ᵢ f ∨ᵢ  ((f ∧ᵢ  more ); f⋆); f
      by (rule EmptyOrChopEqvRule)
 have  8: ⊢ (f ∧ᵢ  more ); (f⋆; f)≡ᵢ ((f ∧ᵢ  more ); f⋆); f
      by (rule ChopAssoc)
 have  9: ⊢ (f⋆ ∧ᵢ  more ) ∧ᵢ ¬ᵢ (f⋆; f) ⊃ᵢ
           (f ∧ᵢ  more ); (f⋆∧ᵢ  more ) ∧ᵢ ¬ᵢ ((f ∧ᵢ  more ); (f⋆; f))
      using 5 7 8 by auto
 have 10: ⊢ f ∧ᵢ  more  ⊃ᵢ  more
      by auto
 from 9 10 show ?thesis by (rule ChopContra)
qed

lemma NotEmptyEqvMore:
⊢ ¬ᵢ empty ≡ᵢ more
by simp

lemma NotCSImpMore:
⊢   ¬ᵢ  (f⋆) ⊃ᵢ  more
proof −
 have  1: ⊢ empty ⊃ᵢ (f⋆)   using EmptyImpCS by blast
 hence 2: ⊢  ¬ᵢ empty ∨ᵢ (f⋆) using itl-prop(35) by metis
 from 2 show ?thesis  using 1 NotEmptyEqvMore itl-prop(31) prop02 prop27 by blast
qed


lemma  CSChopCSImpCS:
⊢  f⋆; f⋆ ⊃ᵢ f⋆
proof −
 have  1: ⊢ f⋆ ≡ᵢ  empty  ∨ᵢ  (f ∧ᵢ  more ); f⋆
      by (rule ChopstarEqv)
 hence 2: ⊢ f⋆; f⋆ ≡ᵢ f⋆ ∨ᵢ  ((f ∧ᵢ  more ); f⋆); f⋆
      by (rule EmptyOrChopEqvRule)
 have 21: ⊢ f⋆; f⋆ ∧ᵢ ¬ᵢ (f⋆) ⊃ᵢ ((f ∧ᵢ  more ); f⋆); f⋆
```

using *2* **by** (*simp add*: *or-d-def*)

have *22*: $\vdash \neg_i (f^\star) \equiv_i \neg_i empty \wedge_i \neg_i ((f \wedge_i \ more); f^\star)$

using *1 prop20* **by** *blast*

have *23*: $\vdash \ \neg_i (f^\star) \supset_i \neg_i ((f \wedge_i \ more); f^\star)$

using *2 22* **using** *itl-prop*(*31*) *itl-prop*(*32*) **by** *blast*

have *24*: $\vdash \ f^\star; f^\star \wedge_i \neg_i (f^\star) \supset_i \neg_i (f^\star)$

by *auto*

have *25*: $\vdash \ f^\star; f^\star \wedge_i \neg_i (f^\star) \supset_i \neg_i ((f \wedge_i \ more); f^\star)$

using *23 24 MP* **by** *auto*

have *3*: $\vdash f^\star; f^\star \wedge_i \neg_i (f^\star) \supset_i ((f \wedge_i \ more); f^\star); f^\star \wedge_i \neg_i ((f \wedge_i \ more); f^\star)$

using *21 25* **by** *auto*

have *4*: $\vdash (f \wedge_i \ more);(f^\star; f^\star) \equiv_i ((f \wedge_i \ more); f^\star); f^\star$

by (*rule ChopAssoc*)

have *5*: $\vdash f^\star; f^\star \wedge_i \neg_i (f^\star) \supset_i (f \wedge_i \ more); (f^\star; f^\star) \wedge_i \neg_i ((f \wedge_i \ more); f^\star)$

using *3 4* **by** *auto*

have *6*: $\vdash f \wedge_i \ more \supset_i \ more$

by *auto*

from *5 6* **show** *?thesis* **using** *ChopContra* **by** *blast*

qed

lemma *ImpChopPlus*:

$\vdash \ f \supset_i f;f^\star$

proof −

have *1*: $\vdash f^\star \equiv_i empty \vee_i \ f; f^\star$ **by** (*rule CSEqvOrChopCS*)

hence *2*: $\vdash f;f^\star \equiv_i f;empty \vee_i \ f; (f;f^\star)$ **using** *ChopOrEqvRule* **by** *blast*

have *3*: $\vdash f;empty \equiv_i f$ **using** *ChopEmpty* **by** *blast*

from *2 3* **show** *?thesis* **by** *simp*

qed

lemma *ImpCS*:

$\vdash \ f \supset_i f^\star$

proof −

have *1*: $\vdash f \supset_i f;f^\star$ **by** (*rule ImpChopPlus*)

hence *2*: $\vdash f \supset_i \ empty \ \vee_i \ f;f^\star$ **by** *auto*

from *2* **show** *?thesis* **using** *CSEqvOrChopCS* **using** *prop15* **by** *blast*

qed

lemma *CSChopImpCS*:

$\vdash \ f^\star; f \supset_i f^\star$

proof −

have *1*: $\vdash f \supset_i f^\star$ **by** (*rule ImpCS*)

hence *2*: $\vdash f^\star; f \supset_i f^\star; f^\star$ **by** (*rule RightChopImpChop*)

have *3*: $\vdash f^\star; f^\star \supset_i f^\star$ **by** (*rule CSChopCSImpCS*)

from *2 3* **show** *?thesis* **using** *prop02* **by** *blast*

qed

lemma *ChopPlusImpCS*:

$\vdash \ f;f^\star \ \supset_i f^\star$

proof −

have *1*: $\vdash\ f;f^\star \supset_i \mathit{empty} \lor_i f;f^\star$ **by** *auto*
from *1* **show** *?thesis* **using** *CSEqvOrChopCS* **using** *prop15* **by** *blast*
qed

lemma *CSChopEqvOrChopPlusChop*:
$\vdash\ f^\star;\, g \equiv_i g \lor_i (f;f^\star)\, ;\, g$
proof –
have *1*: $\vdash f^\star \equiv_i \mathit{empty} \lor_i f;f^\star$ **by** (*rule CSEqvOrChopCS*)
from *1* **show** *?thesis* **using** *EmptyOrChopEqvRule* **by** *blast*
qed

lemma *CSElim*:
assumes $\vdash\ \mathit{empty} \supset_i g$
$\vdash (f \land_i \mathit{more});\, g \supset_i g$
shows $\vdash f^\star \supset_i g$
proof –
have *1*: $\vdash f^\star \equiv_i \mathit{empty} \lor_i (f \land_i \mathit{more});\, f^\star$
by (*rule ChopstarEqv*)
have *2*: $\vdash\ \mathit{empty} \supset_i g$
using *assms* **by** *blast*
have *3*: $\vdash (f \land_i \mathit{more});\, g \supset_i g$
using *assms* **by** *blast*
have *31*: $\vdash\ \neg_i\, g \supset_i \mathit{more}$
using *2* **by** *auto*
have *32*: $\vdash \neg_i\, g \supset_i \neg_i ((f \land_i \mathit{more});\, g)$
using *3 prop27* **by** *blast*
have *33*: $\vdash f^\star \land_i \mathit{more} \supset_i (f \land_i \mathit{more});\, f^\star$
using *1* **using** *CSAndMoreEqvAndMoreChop itl-prop*(*31*) **by** *blast*
have *34*: $\vdash f^\star \land_i \neg_i\, g \supset_i f^\star \land_i \mathit{more}$
using *31* **by** *auto*
have *35*: $\vdash f^\star \land_i \neg_i\, g \supset_i (f \land_i \mathit{more});\, f^\star$
using *33 34* **by** *auto*
have *36*: $\vdash f^\star \land_i \neg_i\, g \supset_i \neg_i ((f \land_i \mathit{more});\, g)$
using *32* **by** *auto*
have *4*: $\vdash f^\star \land_i \neg_i\, g \supset_i (f \land_i \mathit{more});\, f^\star \land_i \neg_i ((f \land_i \mathit{more});\, g)$
using *35 36* **by** *auto*
have *5*: $\vdash f \land_i \mathit{more} \supset_i \mathit{more}$
by *auto*
from *4 5* **show** *?thesis* **using** *ChopContra* **by** *blast*
qed

lemma *CSCSImpCS*:
$\vdash\ (f^\star)^\star \supset_i f^\star$
proof –
have *1*: $\vdash\ \mathit{empty} \supset_i f^\star$ **by** (*rule EmptyImpCS*)
have *2*: $\vdash (f^\star \land_i \mathit{more});\, f^\star \supset_i f^\star;\, f^\star$ **by** (*rule AndChopA*)
have *3*: $\vdash f^\star;\, f^\star \supset_i f^\star$ **by** (*rule CSChopCSImpCS*)
have *4*: $\vdash (f^\star \land_i \mathit{more});\, f^\star \supset_i f^\star$ **using** *2 3 prop02* **by** *blast*
from *1 4* **show** *?thesis* **using** *CSElim* **by** *blast*

qed

lemma *RightEmptyOrChopEqv*:
$\vdash\ g;(\ \mathit{empty}\ \lor_i\ f) \equiv_i g \lor_i\ (g;\ f)$
proof −
have *1*: $\vdash g;(\ \mathit{empty}\ \lor_i\ f) \equiv_i g;\mathit{empty} \lor_i g;f$ **by** (*rule ChopOrEqv*)
have *2*: $\vdash g;\mathit{empty}\ \equiv_i g$ **by** (*rule ChopEmpty*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RightEmptyOrChopEqvRule*:
assumes $\vdash f \equiv_i \mathit{empty} \lor_i \mathit{f1}$
shows $\vdash g;f \equiv_i g \lor_i (g;\mathit{f1})$
proof −
have *1*: $\vdash f \equiv_i \mathit{empty} \lor_i \mathit{f1}$ **using** *assms* **by** *auto*
hence *2*: $\vdash g;f \equiv_i g;(\mathit{empty} \lor_i \mathit{f1})$ **by** (*rule RightChopEqvChop*)
have *3*: $\vdash g;(\mathit{empty} \lor_i \mathit{f1}) \equiv_i g \lor_i (g;\mathit{f1})$ **by** (*rule RightEmptyOrChopEqv*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *ChopPlusEqvOrChopChopPlus*:
$\vdash\ (f;f^\star)\ \equiv_i f \lor_i\ f;\ (f;f^\star)$
proof −
have *1*: $\vdash f^\star \equiv_i\ \mathit{empty}\ \lor_i\ f;f^\star$ **by** (*rule CSEqvOrChopCS*)
from *1* **show** *?thesis* **by** (*rule RightEmptyOrChopEqvRule*)
qed

lemma *CSAndEmptyEqvEmpty*:
$\vdash (f^\star) \land_i \mathit{empty} \equiv_i \mathit{empty}$
using *EmptyImpCS* **by** *auto*

lemma *NotAndMoreChopAndEmpty*:
$\vdash \neg_i(((f\ \land_i\ \mathit{more});g) \land_i \mathit{empty})$
by *auto*

lemma *NotChopAndMoreAndEmpty*:
$\vdash \neg_i((f;(g \land_i\ \mathit{more})) \land_i \mathit{empty})$
by *auto*

lemma *ChopCsAndEmptyEqvAndEmpty*:
$\vdash ((f;f^\star) \land_i \mathit{empty}) \equiv_i (f\ \land_i\ \mathit{empty})$
proof −
have *1*: $\vdash ((f;f^\star) \land_i \mathit{empty}) \equiv_i (f \land_i \mathit{empty});(f^\star \land_i \mathit{empty})$
using *ChopAndEmptyEqvEmptyChopEmpty* **by** *blast*
have *2*: $\vdash (f \land_i \mathit{empty});(f^\star \land_i \mathit{empty}) \equiv_i (f \land_i \mathit{empty});\mathit{empty}$
using *CSAndEmptyEqvEmpty* **using** *RightChopEqvChop* **by** *blast*
have *3*: $\vdash (f \land_i \mathit{empty});\mathit{empty} \equiv_i f \land_i \mathit{empty}$
by (*rule ChopEmpty*)
from *1 2 3* **show** *?thesis* **by** *auto*

qed

lemma *AndMoreChopAndMoreEqvAndMoreChop*:
$\vdash (f \wedge_i \mathit{more});g \wedge_i \mathit{more} \equiv_i (f \wedge_i \mathit{more});g$
apply *simp-all*
using *interval-prefix-length-good* **by** *auto*

lemma *ChopPlusEqv*:
$\vdash (f;f^\star) \equiv_i f \vee_i (f \wedge_i \mathit{more}); (f;f^\star)$
proof −
have *1*: $\vdash f^\star \equiv_i \mathit{empty} \vee_i (f \wedge_i \mathit{more}); f^\star$
by (*rule ChopstarEqv*)
have *2*: $\vdash f^\star \equiv_i \mathit{empty} \vee_i f;f^\star$
by (*rule CSEqvOrChopCS*)
hence *3*: $\vdash \mathit{empty} \vee_i f;f^\star \equiv_i \mathit{empty} \vee_i (f \wedge_i \mathit{more});f^\star$
using *1 2 prop21* **by** *blast*
have *4*: $\vdash (f \wedge_i \mathit{more});(f^\star) \equiv_i (f \wedge_i \mathit{more});(\mathit{empty} \vee_i f;f^\star)$
using *2* **using** *RightChopEqvChop* **by** *blast*
hence *5*: $\vdash \mathit{empty} \vee_i f;f^\star \equiv_i \mathit{empty} \vee_i (f \wedge_i \mathit{more});(\mathit{empty} \vee_i f;f^\star)$
using *3 4* **by** *auto*
have *6*: $\vdash (f \wedge_i \mathit{more}); (\mathit{empty} \vee_i f;f^\star) \equiv_i$
$(f \wedge_i \mathit{more}); \mathit{empty} \vee_i (f \wedge_i \mathit{more}); (f;f^\star)$
using *ChopOrEqv* **by** *blast*
have *7*: $\vdash (f \wedge_i \mathit{more}); \mathit{empty} \equiv_i f \wedge_i \mathit{more}$
using *ChopEmpty* **by** *blast*
have *8*: $\vdash \mathit{empty} \vee_i f;f^\star \equiv_i$
$\mathit{empty} \vee_i (f \wedge_i \mathit{more}) \vee_i (f \wedge_i \mathit{more}); (f;f^\star)$
using *5 6 7* **by** *auto*
have *9*: $\vdash (\mathit{empty} \vee_i f;f^\star) \wedge_i \mathit{more} \equiv_i f;f^\star \wedge_i \mathit{more}$
by *auto*
have *10*: $\vdash (\mathit{empty} \vee_i (f \wedge_i \mathit{more}) \vee_i (f \wedge_i \mathit{more}); (f;f^\star)) \wedge_i \mathit{more} \equiv_i$
$((f \wedge_i \mathit{more}) \vee_i (f \wedge_i \mathit{more}); (f;f^\star)) \wedge_i \mathit{more}$
by *auto*
have *11*: $\vdash ((f \wedge_i \mathit{more}) \vee_i (f \wedge_i \mathit{more}); (f;f^\star)) \wedge_i \mathit{more} \equiv_i$
$(f \wedge_i \mathit{more}) \vee_i (f \wedge_i \mathit{more}); (f;f^\star)$
using *AndMoreChopAndMoreEqvAndMoreChop*
by (*metis 10 RightEmptyOrChopEqv itl-prop*(*31*) *itl-prop*(*32*) *prop15*)
have *12*: $\vdash f;f^\star \wedge_i \mathit{more} \equiv_i (f \wedge_i \mathit{more}) \vee_i (f \wedge_i \mathit{more}); (f;f^\star)$
using *8 9 10 11* **by** *auto*
have *13*: $\vdash f;f^\star \wedge_i \mathit{empty} \equiv_i f \wedge_i \mathit{empty}$
by (*rule ChopCsAndEmptyEqvAndEmpty*)
have *14*: $\vdash (f \wedge_i \mathit{more}) \vee_i (f \wedge_i \mathit{more}); (f;f^\star) \vee_i (f \wedge_i \mathit{empty}) \equiv_i$
$f \vee_i (f \wedge_i \mathit{more});(f;f^\star)$
by *auto*
have *15*: $\vdash f;f^\star \equiv_i (f;f^\star \wedge_i \mathit{empty}) \vee_i (f;f^\star \wedge_i \mathit{more})$
by *auto*
from *11 12 13 14 15* **show** *?thesis* **by** *auto*
qed

lemma *ChopPlusImpChopPlus*:
assumes $\vdash\ f \supset_i g$
shows $\vdash\ f;f^\star \supset_i g;g^\star$
proof $-$
have *1*: $\vdash f \supset_i g$
  using *assms* **by** *auto*
have *2*: $\vdash f;f^\star\ \equiv_i f \vee_i\ (f \wedge_i\ more\ );\ (f;f^\star)$
  by (*rule ChopPlusEqv*)
have *3*: $\vdash g;g^\star\ \equiv_i g \vee_i\ (g \wedge_i\ more\ );(g;g^\star)$
  by (*rule ChopPlusEqv*)
have *4*: $\vdash\ f;f^\star\ \wedge_i \neg_i (g;g^\star\ ) \supset_i ((f \wedge_i\ more\ );\ (f;f^\star)\ ) \wedge_i \neg_i ((g \wedge_i\ more\ );\ (g;g^\star)\ )$
  using *1 2 3* **by** *auto*
have *5*: $\vdash f \wedge_i\ more\ \supset_i g \wedge_i\ more$ **using** *1*
  by *auto*
have *6*: $\vdash (f \wedge_i\ more\ );\ (f;f^\star)\ \supset_i (g \wedge_i\ more\ );\ (f;f^\star)$
  using *5* **by** (*rule LeftChopImpChop*)
have *7*: $\vdash f;f^\star\ \wedge_i \neg_i (g;g^\star\ ) \supset_i$
    $((g \wedge_i\ more\ );\ (f;f^\star)\ ) \wedge_i \neg_i ((g \wedge_i\ more\ );\ (g;g^\star)\ )$
  using *4 6* **by** *auto*
have *8*: $\vdash g \wedge_i\ more\ \supset_i\ more$
  by *auto*
from *7 8* **show** *?thesis* **using** *ChopContra* **by** *blast*
qed

lemma *ChopChopPlusImpChopPlus*:
$\vdash\ f;\ (f;f^\star)\ \supset_i f;f^\star$
proof $-$
have *1*: $\vdash\ empty\ \vee_i\ more$ **by** *auto*
hence *2*: $\vdash f \supset_i\ empty\ \vee_i\ (f \wedge_i\ more\ )$ **by** *auto*
hence *3*: $\vdash f;\ (f;f^\star)\ \supset_i (f;f^\star)\ \vee_i\ (f \wedge_i\ more\ );(f;f^\star)$ **by** (*rule EmptyOrChopImpRule*)
have *4*: $\vdash\ f;f^\star \equiv_i f \vee_i\ (f \wedge_i\ more\ );\ (f;f^\star)$ **by** (*rule ChopPlusEqv*)
hence *5*: $\vdash (f \wedge_i\ more\ );\ (f;f^\star)\ \supset_i f;f^\star$ **by** *auto*
from *3 5* **show** *?thesis* **using** *ChopPlusImpCS RightChopImpChop* **by** *blast*
qed

lemma *CSImpCS*:
assumes $\vdash\ f \supset_i g$
shows $\vdash f^\star \supset_i g^\star$
proof $-$
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;f^\star \supset_i g;g^\star$ **by** (*rule ChopPlusImpChopPlus*)
hence *3*: $\vdash\ empty\ \vee_i\ f;f^\star\ \supset_i\ empty\ \vee_i\ g;g^\star$ **by** *auto*
from *2 3* **show** *?thesis* **using** *CSEqvOrChopCS prop14 prop15* **by** *blast*
qed

lemma *ChopPlusIntro*:
assumes $\vdash\ f \wedge_i \neg_i\ g \supset_i (g \wedge_i\ more\ );\ f$
shows $\vdash f \supset_i g;g^\star$
proof $-$
have *1*: $\vdash f \wedge_i \neg_i\ g \supset_i (g \wedge_i\ more\ );\ f$ **using** *assms* **by** *auto*

have *2*: $\vdash g;g^\star \ \equiv_i g \vee_i \ (g \wedge_i \ more\ ); (g;g^\star)$ **by** (*rule ChopPlusEqv*)
have *3*: $\vdash f \wedge_i \neg_i (g;g^\star\ ) \supset_i$
$(g \wedge_i \ more\ ); f \wedge_i \neg_i ((g \wedge_i \ more\ ); (g;g^\star)\ )$ **using** *1 2* **by** *auto*
have *4*: $\vdash g \wedge_i \ more \ \supset_i \ more$ **by** *auto*
from *3 4* **show** *?thesis* **using** *ChopContra* **by** *blast*
qed

lemma *ChopPlusElim*:
assumes $\vdash \ f \supset_i g$
$\vdash (f \wedge_i \ more\ ); g \supset_i g$
shows $\vdash f;f^\star \ \supset_i g$
proof –
have *1*: $\vdash f;f^\star \ \equiv_i f \vee_i \ (f \wedge_i \ more\ ); (f;f^\star)$ **by** (*rule ChopPlusEqv*)
have *2*: $\vdash f \supset_i g$ **using** *assms* **by** *blast*
hence *21*: $\vdash \neg_i g \supset_i \neg_i f$ **by** *auto*
have *3*: $\vdash (f \wedge_i \ more\ ); g \supset_i g$ **using** *assms* **by** *blast*
hence *31*: $\vdash \neg_i \ g \supset_i \neg_i ((f \wedge_i more\ ); g)$ **using** *prop27* **by** *blast*
hence *32*: $\vdash f;f^\star \wedge_i \neg_i \ g \supset_i \neg_i ((f \wedge_i more\ ); g)$ **by** *auto*
have *33*: $\vdash f;f^\star \wedge_i \neg_i \ g \supset_i (f \wedge_i \ more\ ); (f;f^\star)$ **using** *1 21* **by** *auto*
have *4*: $\vdash \ f;f^\star \wedge_i \neg_i \ g \supset_i$
$(f \wedge_i \ more\ ); (f;f^\star) \ \wedge_i \neg_i ((f \wedge_i more\ ); g)$ **using** *31 33* **by** *auto*
have *5*: $\vdash f \wedge_i \ more \ \supset_i \ more$ **by** *auto*
from *4 5* **show** *?thesis* **using** *ChopContra* **by** *blast*
qed

lemma *ChopPlusElimWithoutMore*:
assumes $\vdash \ f \supset_i g$
$\vdash f; g \supset_i g$
shows $\vdash f;f^\star \ \supset_i g$
proof –
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *blast*
have *2*: $\vdash (f; g) \supset_i g$ **using** *assms* **by** *blast*
have *3*: $\vdash (f \wedge_i \ more\ ); g \supset_i f; g$ **by** (*rule AndChopA*)
have *4*: $\vdash (f \wedge_i \ more\ ); g \supset_i g$ **using** *2 3 prop02* **by** *blast*
from *1 4* **show** *?thesis* **using** *ChopPlusElim* **by** *blast*
qed

lemma *ChopPlusEqvChopPlus*:
assumes $\vdash \ f \equiv_i g$
shows $\vdash f;f^\star \ \equiv_i g;g^\star$
proof –
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash f \supset_i g$ **by** *auto*
hence *3*: $\vdash f;f^\star \ \supset_i g;g^\star$ **by** (*rule ChopPlusImpChopPlus*)
have *4*: $\vdash g \supset_i f$ **using** *1* **by** *auto*
hence *5*: $\vdash \ g;g^\star \supset_i f;f^\star$ **by** (*rule ChopPlusImpChopPlus*)
from *3 5* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *CSEqvCS*:

assumes $\vdash\ f \equiv_i g$
shows $\vdash f^\star \equiv_i g^\star$
proof −
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash f;f^\star \ \equiv_i g;g^\star$ **by** (*rule ChopPlusEqvChopPlus*)
hence *3*: $\vdash\ empty\ \lor_i f;f^\star\ \ \equiv_i\ empty\ \lor_i\ g;g^\star$ **by** *auto*
from *3* **show** *?thesis* **using** *CSEqvOrChopCS* **using** *assms* **by** *auto*
qed

lemma *AndCSA*:
$\vdash\ \ (f \land_i g)^\star \supset_i f^\star$
proof −
have *1*: $\vdash f \land_i g \supset_i f$ **by** *auto*
from *1* **show** *?thesis* **using** *CSImpCS* **by** *blast*
qed

lemma *AndCSB*:
$\vdash\ \ (f \land_i g)^\star \supset_i g^\star$
proof −
have *1*: $\vdash f \land_i g \supset_i g$ **by** *auto*
from *1* **show** *?thesis* **using** *CSImpCS* **by** *blast*
qed

lemma *CSIntro*:
assumes $\vdash\ f \land_i\ more\ \supset_i (g \land_i\ more\ );\ f$
shows $\vdash f \supset_i g^\star$
proof −
have *1*: $\vdash f \land_i\ more\ \supset_i (g \land_i\ more\ );\ f$
using *assms* **by** *auto*
have *2*: $\vdash\ more\ \equiv_i \neg_i\ empty$
by *auto*
have *3*: $\vdash f \land_i \neg_i\ empty\ \supset_i (g \land_i\ more\ );\ f$
using *1 2* **by** *auto*
have *4*: $\vdash g^\star \equiv_i\ empty\ \lor_i\ (g \land_i\ more\ );\ g^\star$
by (*rule ChopstarEqv*)
hence *41*: $\vdash \neg_i(empty\ \lor_i\ (g \land_i\ more\ );\ g^\star) \equiv_i \neg_i empty \land_i \neg_i((g \land_i\ more\ );\ g^\star)$
using *prop20 prop21* **by** *blast*
have *411*: $\vdash \neg_i empty \land_i \neg_i((g \land_i\ more\ );\ g^\star) \equiv_i more \land_i \neg_i((g \land_i\ more\ );\ g^\star)$
using *NotEmptyEqvMore* **using** *prop06* **by** *blast*
have *42*: $\vdash \neg_i(\ g^\star)\ \equiv_i more \land_i \neg_i((g \land_i\ more\ );\ g^\star)$
using *4 41 411 prop01 prop03* **by** *blast*
have *43*: $\vdash f \land_i \neg_i(\ g^\star) \supset_i f \land_i more \land_i \neg_i\ ((g \land_i\ more\ );\ g^\star)$
using *42* **using** *itl-prop*(*31*) *prop12* **by** *blast*
have *44*: $\vdash\ f \land_i more \land_i \neg_i\ ((g \land_i\ more\ );\ g^\star) \supset_i (g \land_i\ more\ );\ f \land_i \neg_i\ ((g \land_i\ more\ );\ g^\star)$
using *3 43* **by** *auto*
have *5*: $\vdash f \land_i \neg_i\ (\ g^\star) \supset_i$
$(g \land_i\ more\ );\ f \land_i \neg_i\ ((g \land_i\ more\ );\ g^\star)$

```
        using 43 44 prop02 by auto
 have  6: ⊢ g ∧_i  more  ⊃_i  more
        by auto
 from 5 6 show ?thesis using   ChopContra by blast
qed

lemma CSElimWithoutMore:
 assumes ⊢    empty  ⊃_i g
         ⊢ f; g ⊃_i g
 shows  ⊢ f^⋆ ⊃_i g
proof −
 have 1: ⊢  empty  ⊃_i g using assms by blast
 have 2: ⊢ f; g ⊃_i g  using assms by blast
 have 3: ⊢ (f ∧_i  more ); g ⊃_i f; g  by (rule AndChopA)
 have 4: ⊢ (f ∧_i  more ); g ⊃_i g  using 2 3 prop02 by blast
 from 1 4 show ?thesis using CSElim by blast
qed

lemma ChopAssocB:
⊢ (f;g);h ≡_i f;(g;h)
using ChopAssoc itl-prop(30) by blast

lemma CSChopEqvChopOrRule:
 assumes ⊢   f≡_i (g^⋆; h)
 shows  ⊢ f ≡_i (g; f) ∨_i  h
proof −
 have   1: ⊢ f ≡_i (g^⋆; h)  using assms by auto
 have   2: ⊢ g^⋆ ≡_i  empty  ∨_i  (g; g^⋆)  by (rule CSEqvOrChopCS)
 hence  3: ⊢ g^⋆; h ≡_i h ∨_i  ((g; g^⋆); h)  by (rule EmptyOrChopEqvRule)
 have   4: ⊢ (g; g^⋆); h ≡_i  g; (g^⋆; h)    by (rule ChopAssocB)
 hence 41: ⊢ g^⋆; h ≡_i h ∨_i g; (g^⋆; h) using 3 by auto
 have   5: ⊢ g; f ≡_i g; (g^⋆; h)  using 1 by (rule RightChopEqvChop)
 hence  6: ⊢ (g^⋆; h) ≡_i h ∨_i  g; f using 41 by auto
 hence 61: ⊢ (g^⋆; h) ≡_i (g; f) ∨_i h  by auto
 from 1 61 show ?thesis  using prop03 by blast
qed

lemma CSChopIntroRule:
 assumes ⊢   f ∧_i ¬_i  h ⊃_i g; f
         ⊢ g ⊃_i  more
 shows  ⊢ f ⊃_i g^⋆; h
proof −
 have  1: ⊢ f ∧_i ¬_i  h ⊃_i g; f  using assms by blast
 have  2: ⊢ g ⊃_i  more  using assms by blast
 hence 3: ⊢ g ⊃_i g ∧_i  more  by auto
 hence 4: ⊢ g; f ⊃_i (g ∧_i  more ); f  by (rule LeftChopImpChop)
 have  5: ⊢ f ⊃_i (g ∧_i  more ); f ∨_i  h  using 1 4 by auto
 have  6: ⊢ g^⋆ ≡_i  empty  ∨_i  (g ∧_i  more ); g^⋆  by (rule ChopstarEqv)
 hence 7: ⊢ (g^⋆); h ≡_i h ∨_i  ((g ∧_i  more ); g^⋆); h  by (rule EmptyOrChopEqvRule)
 have  8: ⊢ ((g ∧_i  more ); g^⋆); h ≡_i (g ∧_i  more ); (g^⋆; h)  by (rule ChopAssocB)
```

have *9*: $\vdash (g^{\star});\ h \equiv_i h \vee_i (g \wedge_i \mathit{more});\ (g^{\star};\ h)$ **using** *7 8* **by** *auto*
have *10*: $\vdash f \wedge_i \neg_i (g^{\star};\ h) \supset_i (g \wedge_i \mathit{more});\ f \wedge_i \neg_i ((g \wedge_i \mathit{more});\ (g^{\star};\ h))$ **using** *5 9* **by** *auto*
have *11*: $\vdash g \wedge_i \mathit{more} \supset_i \mathit{more}$ **by** *auto*
from *10 11* **show** *?thesis* **using** *ChopContra* **by** *blast*
qed

lemma *DiamondAndEmptyEqvAndEmpty*:
$\vdash \Diamond f \wedge_i \mathit{empty} \equiv_i f \wedge_i \mathit{empty}$
proof –
have *1*: $\vdash \Diamond f \wedge_i \mathit{empty} \supset_i f \wedge_i \mathit{empty}$ **by** *auto*
have *2*: $\vdash f \wedge_i \mathit{empty} \supset_i \Diamond f \wedge_i \mathit{empty}$ **by** *auto*
from *1 2* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *InitAndEmptyEqvAndEmpty*:
$\vdash (\mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i w \wedge_i \mathit{empty}$
proof –
have *1*: $\vdash (\mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i (w \wedge_i \mathit{empty});\mathit{true}_i \wedge_i \mathit{empty}$
by *auto*
have *2*: $\vdash (w \wedge_i \mathit{empty});\mathit{true}_i \wedge_i \mathit{empty} \equiv_i (w \wedge_i \mathit{empty});(\mathit{true}_i \wedge_i \mathit{empty})$
using *ChopAndEmptyEqvEmptyChopEmpty* **by** *auto*
have *3*: $\vdash (w \wedge_i \mathit{empty});(\mathit{true}_i \wedge_i \mathit{empty}) \equiv_i (w \wedge_i \mathit{empty});\mathit{empty}$
using *RightChopEqvChop itl-prop*(*17*) **by** *blast*
have *4*: $\vdash (w \wedge_i \mathit{empty});\mathit{empty} \equiv_i w \wedge_i \mathit{empty}$
using *ChopEmpty* **by** *blast*
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *InitAndNotBoxInitImpNotEmpty*:
$\vdash \mathit{init}\ w \wedge_i \neg_i(\Box (\mathit{init}\ w)) \supset_i \neg_i \mathit{empty}$
proof –
have *1*: $\vdash (\mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i w \wedge_i \mathit{empty}$ **by** (*rule InitAndEmptyEqvAndEmpty*)
have *2*: $\vdash \neg_i(\Box (\mathit{init}\ w)) \wedge_i \mathit{empty} \equiv_i \Diamond \neg_i (\mathit{init}\ w) \wedge_i \mathit{empty}$ **by** *auto*
have *3*: $\vdash \Diamond \neg_i (\mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i \neg_i (\mathit{init}\ w) \wedge_i \mathit{empty}$ **by** *auto*
have *4*: $\vdash \neg_i (\mathit{init}\ w) \equiv_i (\mathit{init}\ \neg_i w)$ **by** *auto*
have *5*: $\vdash \neg_i (\mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i \neg_i w \wedge_i \mathit{empty}$ **using** *4 InitAndEmptyEqvAndEmpty* **by** *auto*
have *6*: $\vdash \neg_i(\Box (\mathit{init}\ w)) \wedge_i \mathit{empty} \equiv_i \neg_i w \wedge_i \mathit{empty}$ **using** *2 3 5 prop03* **by** *blast*
have *7*: $\vdash \neg_i(\mathit{init}\ w \wedge_i \neg_i(\Box (\mathit{init}\ w)) \wedge_i \mathit{empty})$ **using** *1 6* **by** *auto*
from *7* **show** *?thesis* **by** *auto*
qed

lemma *BoxImpTrueChopAndEmpty*:
$\vdash \Box f \supset_i \mathit{true}_i;(f \wedge_i \mathit{empty})$
by *auto*

lemma *BoxInitAndMoreImpBoxInitAndMoreAndFinInit*:
$\vdash \Box(\mathit{init}\ w) \wedge_i \mathit{more} \supset_i (\Box (\mathit{init}\ w) \wedge_i \mathit{more}) \wedge_i \mathit{fin}\ (\mathit{init}\ w)$
proof –

have *1*: $\vdash$ *fin* (*init w*) $\equiv_i$ *true*$_i$; (*init w* $\wedge_i$ *empty*) **using** *FinEqvTrueChopAndEmpty* **by** *blast*
have *2*: $\vdash$ $\Box$(*init w*) $\supset_i$ *true*$_i$;(*init w* $\wedge_i$ *empty*) **by** (*rule BoxImpTrueChopAndEmpty*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *CSImpBox*:
assumes $\vdash$ *f* $\supset_i$ *empty* $\vee_i$ ($\Box$ (*init w*) $\wedge_i$ *more*); *f*
shows $\vdash$ *init w* $\wedge_i$ *f* $\supset_i$ $\Box$ (*init w*)
proof –
have *1*: $\vdash$ *f* $\supset_i$ *empty* $\vee_i$ ($\Box$(*init w*) $\wedge_i$ *more*); *f*
using *assms* **by** *auto*
have *2*: $\vdash$ *init w* $\wedge_i$ $\neg_i$($\Box$ (*init w*)) $\supset_i$ $\neg_i$ *empty*
by (*rule InitAndNotBoxInitImpNotEmpty*)
have *3*: $\vdash$ *init w* $\wedge_i$ *f* $\wedge_i$ $\neg_i$($\Box$ (*init w*)) $\supset_i$ ($\Box$ (*init w*) $\wedge_i$ *more*); *f*
using *1 2* **by** *auto*
have *4*: $\vdash$ $\Box$ (*init w*) $\wedge_i$ *more* $\supset_i$ ($\Box$ (*init w*) $\wedge_i$ *more*) $\wedge_i$ *fin* (*init w*)
by (*rule BoxInitAndMoreImpBoxInitAndMoreAndFinInit*)
hence *5*: $\vdash$ ($\Box$(*init w*) $\wedge_i$ *more*); *f* $\supset_i$ (($\Box$ (*init w*) $\wedge_i$ *more*) $\wedge_i$ *fin* (*init w*)); *f*
by (*rule LeftChopImpChop*)
have *6*: $\vdash$ (($\Box$ (*init w*) $\wedge_i$ *more*) $\wedge_i$ *fin* (*init w*)); *f* $\equiv_i$
($\Box$(*init w*) $\wedge_i$ *more*); (*init w* $\wedge_i$ *f*)
by (*rule AndFinChopEqvStateAndChop*)
have *7*: $\vdash$ $\neg_i$($\Box$(*init w*)) $\supset_i$ ($\Box$ (*init w*)) *yields* $\neg_i$($\Box$ (*init w*))
by (*rule NotBoxStateImpBoxYieldsNotBox*)
have *8*: $\vdash$ ($\Box$(*init w*)) *yields* $\neg_i$ ($\Box$ (*init w*)) $\supset_i$
($\Box$ (*init w*) $\wedge_i$ *more*) *yields* $\neg_i$($\Box$(*init w*))
by (*rule AndYieldsA*)
have *9*: $\vdash$ ($\Box$(*init w*) $\wedge_i$ *more*); (*init w* $\wedge_i$ *f*) $\wedge_i$ ($\Box$(*init w*) $\wedge_i$ *more*) *yields* $\neg_i$($\Box$ (*init w*))
$\supset_i$
($\Box$ (*init w*) $\wedge_i$ *more*); ((*init w* $\wedge_i$ *f*) $\wedge_i$ $\neg_i$ ($\Box$ (*init w*)))
by (*rule ChopAndYieldsImp*)
have *10*: $\vdash$ (*init w* $\wedge_i$ *f*) $\wedge_i$ $\neg_i$($\Box$ (*init w*)) $\supset_i$
($\Box$(*init w*) $\wedge_i$ *more*); ((*init w* $\wedge_i$ *f*) $\wedge_i$ $\neg_i$($\Box$ (*init w*)))
using *3 5 6 7 8 9* **by** *auto*
have *11*: $\vdash$ ($\Box$(*init w*) $\wedge_i$ *more*); ((*init w* $\wedge_i$ *f*) $\wedge_i$ $\neg_i$($\Box$ (*init w*))) $\supset_i$
more ; ((*init w* $\wedge_i$ *f*) $\wedge_i$ $\neg_i$($\Box$ (*init w*)))
by (*rule AndChopB*)
have *12*: $\vdash$ (*init w* $\wedge_i$ *f*) $\wedge_i$ $\neg_i$($\Box$ (*init w*)) $\supset_i$
more ; ((*init w* $\wedge_i$ *f*) $\wedge_i$ $\neg_i$($\Box$ (*init w*)))
using *10 11* **by** *auto*
from *12* **show** *?thesis* **using** *MoreChopContra* **by** *blast*
qed

lemma *BoxCSEqvBox*:
$\vdash$ *init w* $\wedge_i$ ($\Box$(*init w*))* $\equiv_i$ $\Box$ (*init w*)
proof –
have *1*: $\vdash$ ($\Box$ (*init w*))* $\equiv_i$ *empty* $\vee_i$ ($\Box$ (*init w*) $\wedge_i$ *more*); ($\Box$ (*init w*))*
by (*rule ChopstarEqv*)
hence *2*: $\vdash$ ($\Box$ (*init w*))* $\supset_i$ *empty* $\vee_i$ ($\Box$ (*init w*) $\wedge_i$ *more*); ($\Box$ (*init w*))*
using *itl-prop*(*31*) **by** *blast*

hence *3*: $\vdash init\ w \wedge_i (\Box\ (init\ w))^\star \supset_i \Box\ (init\ w)$
by (*rule CSImpBox*)
have *11*: $\vdash \Box\ (init\ w) \supset_i (init\ w)$
by *auto*
have *12*: $\vdash \Box(\ init\ w) \supset_i (\Box\ (init\ w))^\star$
by (*rule ImpCS*)
have *13*: $\vdash \Box\ (init\ w) \supset_i init\ w \wedge_i (\Box\ (init\ w))^\star$
using *11 12* **using** *itl-prop*(*32*) **by** *blast*
from *3 13* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *BoxStateAndCSEqvCS*:
$\vdash\ \ \Box(\ init\ w) \wedge_i f^\star \equiv_i init\ w \wedge_i (\Box(\ init\ w) \wedge_i f)^\star$
proof –
have *1*: $\vdash \Box\ (init\ w) \supset_i init\ w$ **by** *auto*
have *2*: $\vdash f^\star \wedge_i\ more\ \equiv_i (f \wedge_i\ more\ );\ f^\star$ **by** (*rule CSAndMoreEqvAndMoreChop*)
have *3*: $\vdash \Box(\ init\ w) \wedge_i ((f \wedge_i\ more\ );\ f^\star) \equiv_i$
$(\Box\ (init\ w) \wedge_i f \wedge_i\ more\ );\ (\Box\ (init\ w) \wedge_i f^\star)$ **by** (*rule BoxStateAndChopEqvChop*)
have *4*: $\vdash \Box\ (init\ w) \wedge_i f \wedge_i\ more\ \supset_i (\Box\ (init\ w) \wedge_i f) \wedge_i\ more$ **by** *auto*
hence *5*: $\vdash (\Box\ (init\ w) \wedge_i f \wedge_i\ more\ );\ (\Box\ (init\ w) \wedge_i f^\star) \supset_i$
$((\Box\ (init\ w) \wedge_i f) \wedge_i\ more\ );\ (\Box\ (init\ w) \wedge_i f^\star)$ **by** (*rule LeftChopImpChop*)
have *6*: $\vdash (\Box(\ init\ w) \wedge_i f^\star) \wedge_i\ more\ \supset_i$
$((\Box\ (init\ w) \wedge_i f) \wedge_i\ more\ );\ (\Box\ (init\ w) \wedge_i f^\star)$ **using** *2 3 5* **by** *auto*
hence *7*: $\vdash \Box\ (init\ w) \wedge_i f^\star \supset_i (\Box\ (init\ w) \wedge_i f)^\star$ **by** (*rule CSIntro*)
have *71*: $\vdash init\ w \wedge_i \Box\ (init\ w) \wedge_i f^\star \supset_i init\ w \wedge_i (\Box\ (init\ w) \wedge_i f)^\star$ **using** *7 prop12* **by** *blast*
have *8*: $\vdash \Box(\ init\ w) \wedge_i f^\star \supset_i init\ w \wedge_i (\Box\ (init\ w) \wedge_i f)^\star$ **using** *1 71 prop37* **by** *blast*
have *11*: $\vdash (\Box\ (init\ w) \wedge_i f)^\star \supset_i (\Box\ (init\ w))^\star$ **by** (*rule AndCSA*)
have *12*: $\vdash init\ w \wedge_i (\Box\ (init\ w))^\star \equiv_i \Box\ (init\ w)$ **by** (*rule BoxCSEqvBox*)
have *13*: $\vdash (\Box\ (init\ w) \wedge_i f)^\star \supset_i f^\star$ **by** (*rule AndCSB*)
have *14*: $\vdash init\ w \wedge_i (\Box\ (init\ w) \wedge_i f)^\star \supset_i init\ w \wedge_i (\Box\ (init\ w))^\star \wedge_i f^\star$ **using** *11 13* **by** *auto*
have *15*: $\vdash init\ w \wedge_i (\Box\ (init\ w))^\star \wedge_i f^\star \supset_i \Box\ (init\ w) \wedge_i f^\star$ **using** *12* **by** *auto*
have *16*: $\vdash init\ w \wedge_i (\Box\ (init\ w) \wedge_i f)^\star \supset_i \Box\ (init\ w) \wedge_i f^\star$ **using** *14 15 prop02* **by** *blast*
from *8 16* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *BaCSImpCS*:
$\vdash\ \ ba\ (f \supset_i g) \supset_i f^\star \supset_i g^\star$
proof –
have *1*: $\vdash f^\star \equiv_i\ empty\ \vee_i\ (f \wedge_i\ more\ );\ f^\star$
by (*rule ChopstarEqv*)
have *2*: $\vdash g^\star \equiv_i\ empty\ \vee_i\ (g \wedge_i\ more\ );\ g^\star$
by (*rule ChopstarEqv*)
have *21*: $\vdash \neg_i(g^\star) \equiv_i\ \neg_i empty\ \wedge_i \neg_i(\ (g \wedge_i\ more\ );\ g^\star)$
using *2 prop20* **by** *blast*
hence *22*: $\vdash \neg_i(g^\star) \equiv_i\ more\ \wedge_i \neg_i(\ (g \wedge_i\ more\ );\ g^\star)$
by (*meson NotCSImpMore itl-prop*(*31*) *itl-prop*(*32*) *prop18 NotEmptyEqvMore*)
have *3*: $\vdash f^\star \wedge_i \neg_i\ (g^\star) \supset_i$
$(empty\ \vee_i (f \wedge_i\ more\ );\ f^\star) \wedge_i more \wedge_i \neg_i\ ((g \wedge_i\ more\ );\ g^\star)$
using *1 22 prop22* **by** *blast*
have *31*: $\vdash (empty\ \vee_i (f \wedge_i\ more\ );\ f^\star) \wedge_i more \equiv_i (f \wedge_i\ more\ );\ f^\star \wedge_i more$

by *auto*
have *32*: $\vdash f^\star \land_i \neg_i (g^\star) \supset_i (f \land_i\ \mathit{more}\ ); f^\star \land_i \neg_i ((g \land_i\ \mathit{more}\ ); g^\star)$
using *3 31* **by** *auto*
have *4*: $\vdash (f \supset_i g) \supset_i (f \land_i\ \mathit{more}\ \supset_i g \land_i\ \mathit{more}\ )$
by *auto*
hence *5*: $\vdash\ \mathit{ba}\ (f \supset_i g) \supset_i\ \mathit{ba}\ (f \land_i\ \mathit{more}\ \supset_i g \land_i\ \mathit{more}\ )$
by (*rule BaImpBa*)
have *6*: $\vdash\ \mathit{ba}\ (f \land_i\ \mathit{more}\ \supset_i g \land_i\ \mathit{more}\ ) \supset_i$
$(f \land_i\ \mathit{more}\ ); f^\star \supset_i (g \land_i\ \mathit{more}\ ); f^\star$
by (*rule BaLeftChopImpChop*)
have *7*: $\vdash\ \mathit{ba}\ (f \supset_i g) \land_i (f \land_i \mathit{more}\ ); f^\star \supset_i (g \land_i\ \mathit{more}\ ); f^\star$
using *5 6* **by** *auto*
have *8*: $\vdash (g \land_i\ \mathit{more}\ ); f^\star \land_i \neg_i ((g \land_i\ \mathit{more}\ ); g^\star)$
$\supset_i\ (g \land_i\ \mathit{more}\ ); (f^\star \land_i \neg_i (g^\star))$
by (*rule ChopAndNotChopImp*)
have *9*: $\vdash (g \land_i\ \mathit{more}\ ); (f^\star \land_i \neg_i (g^\star)) \supset_i\ \mathit{more}\ ; (f^\star \land_i \neg_i (g^\star))$
by (*rule AndChopB*)
have *10*: $\vdash\ \mathit{ba}\ (f \supset_i g) \supset_i \mathit{more}\ ; (f^\star \land_i \neg_i (g^\star)) \supset_i$
$\mathit{more}\ ; (\ \mathit{ba}\ (f \supset_i g) \land_i f^\star \land_i \neg_i (g^\star))$
by (*rule BaChopImpChopBa*)
have *11*: $\vdash\ \mathit{ba}\ (f \supset_i g) \land_i f^\star \land_i \neg_i (g^\star) \supset_i$
$\mathit{more}\ ; (\ \mathit{ba}\ (f \supset_i g) \land_i f^\star \land_i \neg_i (g^\star))$
using *32 7 8 9 10* **by** *auto*
hence *12*: $\vdash \neg_i\ (\ (\mathit{ba}\ (f \supset_i g)) \land_i (f^\star) \land_i (\neg_i (g^\star)))$
using *MoreChopLoop* **by** *blast*
from *12* **show** *?thesis* **using** *prop04 MP itl-prop*(*31*) **by** *blast*
qed

lemma *BaCSEqvCS*:
$\vdash\quad \mathit{ba}\ (f \equiv_i g) \supset_i (f^\star \equiv_i g^\star)$
proof −
have *1*: $\vdash \mathit{ba}\ (f \equiv_i g) \equiv_i \mathit{ba}\ (f \supset_i g) \land_i \mathit{ba}\ (g \supset_i f)$ **by** *auto*
have *2*: $\vdash \mathit{ba}\ (f \supset_i g) \supset_i (f^\star \supset_i g^\star)$ **by** (*rule BaCSImpCS*)
have *3*: $\vdash \mathit{ba}\ (g \supset_i f) \supset_i (g^\star \supset_i f^\star)$ **by** (*rule BaCSImpCS*)
have *4*: $\vdash \mathit{ba}\ (f \equiv_i g) \supset_i (f^\star \supset_i g^\star) \land_i\ (g^\star \supset_i f^\star)$ **using** *1 2 3* **by** *auto*
have *5*: $\vdash (f^\star \supset_i g^\star) \land_i (g^\star \supset_i f^\star) \equiv_i (f^\star \equiv_i g^\star)$ **by** *auto*
from *4 5* **show** *?thesis* **by** *auto*
qed

lemma *BaAndCSImport*:
$\vdash\quad \mathit{ba}\ \ f \land_i g^\star \supset_i (f \land_i g)^\star$
proof −
have *1*: $\vdash f \supset_i (g \supset_i f \land_i g)$ **by** *auto*
hence *2*: $\vdash\ \mathit{ba}\ \ f \supset_i\ \mathit{ba}\ \ (g \supset_i f \land_i g)$ **by** (*rule BaImpBa*)
have *3*: $\vdash\ \mathit{ba}\ \ (g \supset_i f \land_i g) \supset_i g^\star \supset_i (f \land_i g)^\star$ **by** (*rule BaCSImpCS*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *CSSkip*:
$\vdash \mathit{skip}^\star$

by (*metis ChopPlusImpCS EmptyImpCS EmptyNextInducta next-d-def*)

5.8 Properties of While

lemma *WhileEqvIf*:
$\vdash$ *while* (*init w*) *do* *f* $\equiv_i$ *if*$_i$ (*init w*) *then* (*f*; (*while* (*init w*) *do* *f*)) *else* *empty*
proof $-$
have *1*: $\vdash$ *while* (*init w*) *do* *f* $\equiv_i$ ((*init w*) $\wedge_i$ *f*)* $\wedge_i$ *fin* $\neg_i$ (*init w*)
by (*simp add*: *while-d-def*)
have *2*: $\vdash$ (*init w* $\wedge_i$ *f*)* $\equiv_i$ *empty* $\vee_i$ ((*init w* $\wedge_i$ *f*); (*init w* $\wedge_i$ *f*)*)
by (*rule CSEqvOrChopCS*)
have *21*: $\vdash$ ((*init w*) $\wedge_i$ *f*)* $\wedge_i$ *fin* $\neg_i$ (*init w*) $\equiv_i$
(*empty* $\vee_i$ ((*init w* $\wedge_i$ *f*); (*init w* $\wedge_i$ *f*)*)) $\wedge_i$ *fin* $\neg_i$ (*init w*)
using *2 prop06* **by** *blast*
have *22*: $\vdash$ (*empty* $\vee_i$ ((*init w* $\wedge_i$ *f*); (*init w* $\wedge_i$ *f*)*)) $\wedge_i$ *fin* $\neg_i$ (*init w*) $\equiv_i$
(*empty* $\wedge_i$ *fin* $\neg_i$(*init w*)) $\vee_i$ (((*init w* $\wedge_i$ *f*); (*init w* $\wedge_i$ *f*)*) $\wedge_i$ *fin* $\neg_i$ (*init w*))
by *auto*
have *3*: $\vdash$ *empty* $\wedge_i$ *fin* $\neg_i$ (*init w*) $\equiv_i$ $\neg_i$ (*init w*) $\wedge_i$ *empty*
using *AndFinEqvChopAndEmpty EmptyChop prop03* **by** *blast*
have *4*: $\vdash$ (*init w* $\wedge_i$ *f*); (*init w* $\wedge_i$ *f*)* $\equiv_i$ *init w* $\wedge_i$ (*f*; (*init w* $\wedge_i$ *f*)*)
by (*rule StateAndChop*)
have *41*: $\vdash$ ((*init w* $\wedge_i$ *f*); (*init w* $\wedge_i$ *f*)*) $\wedge_i$ *fin* $\neg_i$ (*init w*) $\equiv_i$
init w $\wedge_i$ (*f*; (*init w* $\wedge_i$ *f*)*) $\wedge_i$ *fin* $\neg_i$ (*init w*)
using *4* **by** *auto*
have *42*: $\vdash$ *init w* $\wedge_i$ (*f*; (*init w* $\wedge_i$ *f*)*) $\wedge_i$ *fin* $\neg_i$ (*init w*) $\equiv_i$
init w $\wedge_i$ (*f*; (*init w* $\wedge_i$ *f*)*) $\wedge_i$ *fin* (*init* $\neg_i$ *w*)
by (*simp*)
have *5*: $\vdash$ (*f*; ((*init w* $\wedge_i$ *f*)*)) $\wedge_i$ (*fin* (*init* $\neg_i$ *w*))
$\equiv_i$ (*f*; ((*init w* $\wedge_i$ *f*)* $\wedge_i$ (*fin* (*init* $\neg_i$ *w*))))
by (*rule ChopAndFin*)
have *51*: $\vdash$ (*f*; ((*init w* $\wedge_i$ *f*)* $\wedge_i$ (*fin* (*init* $\neg_i$ *w*)))) $\equiv_i$
(*f*; ((*init w* $\wedge_i$ *f*)* $\wedge_i$ (*fin* $\neg_i$ (*init* *w*))))
by (*simp*)
have *52*: $\vdash$ *init w* $\wedge_i$ (*f*; (*init w* $\wedge_i$ *f*)*) $\wedge_i$ *fin* $\neg_i$ (*init w*) $\equiv_i$
init w $\wedge_i$ (*f*; ((*init w* $\wedge_i$ *f*)* $\wedge_i$ *fin* $\neg_i$ (*init* *w*)))
using *42 5 51* **by** *auto*
have *6*: $\vdash$ *f*; ((*init w* $\wedge_i$ *f*)* $\wedge_i$ *fin* $\neg_i$ (*init w*)) $\equiv_i$ *f*; *while* (*init w*) *do* *f*
by (*simp add*: *while-d-def*)
have *61*: $\vdash$ *init w* $\wedge_i$ (*f*; ((*init w* $\wedge_i$ *f*)* $\wedge_i$ *fin* $\neg_i$ (*init* *w*))) $\equiv_i$
init w $\wedge_i$ (*f*; *while* (*init w*) *do* *f*) **using** *6*
by *auto*
have *62*: $\vdash$ (*empty* $\wedge_i$ *fin* $\neg_i$ (*init w*)) $\vee_i$ (((*init w* $\wedge_i$ *f*); (*init w* $\wedge_i$ *f*)*) $\wedge_i$ *fin* $\neg_i$ (*init w*))
$\equiv_i$ ($\neg_i$ (*init w*) $\wedge_i$ *empty*) $\vee_i$ (*init w* $\wedge_i$ (*f*; *while* (*init w*) *do* *f*))
using *21 22 3 4 52 61* **by** *auto*
have *7*: $\vdash$ *while* (*init w*) *do* *f*
$\equiv_i$ ($\neg_i$ (*init w*) $\wedge_i$ *empty*) $\vee_i$ (*init w* $\wedge_i$ (*f*; *while* (*init w*) *do* *f*))
using *1 21 22 62 prop03* **by** *blast*
have *71*: $\vdash$ *if*$_i$ (*init w*) *then* (*f*; (*while* (*init w*) *do* *f*)) *else* *empty* $\equiv_i$
($\neg_i$ (*init w*) $\wedge_i$ *empty*) $\vee_i$ (*init w* $\wedge_i$ (*f*; *while* (*init w*) *do* *f*))
by *auto*

```
 from 7 71 show ?thesis using itl-prop(30) prop03 by blast
qed

lemma WhileChopEqvIf:
⊢  ( while ( init w) do f); g ≡_i  if_i(init w) then (f; ((while ( init w) do f); g))  else  g
proof −
 have  1: ⊢  while  (init w)  do  f ≡_i
             if_i  (init w)  then  (f; ( while  (init w)  do  f))  else   empty
      by (rule WhileEqvIf)
 hence 2: ⊢ ( while  (init w)  do  f); g ≡_i
             if_i  (init w)  then  ((f;  while  (init w)  do  f); g)  else   (empty ; g)
      by (rule IfChopEqvRule)
 have  3: ⊢  empty ; g ≡_i g
      by (rule EmptyChop)
 have  4: ⊢ if_i  (init w)  then  ((f;  while  (init w)  do  f); g)  else   (empty ; g) ≡_i
           if_i  (init w)  then  ((f;  while  (init w)  do  f); g)  else   g
      using 3 by (simp add: ifthenelse-d-def)
 have  5: ⊢ ((f;  while  (init w)  do  f); g) ≡_i (f;  (while  (init w)  do  f; g))
      by (rule ChopAssocB)
 have  6: ⊢ if_i  (init w)  then  ((f;  while  (init w)  do  f); g)  else   g ≡_i
           if_i  (init w)  then  (f;  ((while  (init w)  do  f); g))  else   g
      using 5 by (simp add: ifthenelse-d-def)
 from 1 2 4 6 show ?thesis using prop02 prop03 by blast
qed

lemma WhileChopEqvIfRule:
 assumes ⊢  f ≡_i ( while  (init w)  do  g); h
 shows   ⊢ f ≡_i  if_i  (init w)  then  (g; f)  else  h
proof −
 have 1: ⊢ f ≡_i ( while  (init w)  do  g); h
     using assms by auto
 have 2: ⊢ ( while  (init w)  do  g); h ≡_i
           if_i  (init w)  then  (g; (( while  (init w)  do  g); h))  else  h
     by (rule WhileChopEqvIf)
 have 3: ⊢ (g; f) ≡_i (g; (( while  (init w)  do  g); h))
     using 1 by (rule RightChopEqvChop)
 have 4: ⊢ (g; (( while  (init w)  do  g); h)) ≡_i (g; f)
     using 3 by auto
 have 5: ⊢ if_i  (init w)  then  (g; (( while  (init w)  do  g); h))  else  h ≡_i
          if_i  (init w)  then  (g; f)  else  h
     using 4 by (simp add: ifthenelse-d-def)
 from 1 2 5 show ?thesis using prop03 by blast
qed

lemma WhileImpFin:
⊢   while  (init w)  do  f ⊃_i  fin ¬_i ( init w)
proof −
 have 1: ⊢ (init w ∧_i f)^⋆ ∧_i  fin ¬_i ( init w) ⊃_i  fin ¬_i ( init w) by auto
 from 1 show ?thesis by (simp add: while-d-def)
qed
```

lemma *WhileEqvEmptyOrChopWhile*:
$\vdash\ \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ f \equiv_i (\neg_i\ (\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i (\mathit{init}\ w \wedge_i (f \wedge_i \mathit{more}\ );\mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ f)$
proof –
have *1*: $\vdash (\mathit{init}\ w \wedge_i f)^\star \equiv_i \mathit{empty} \vee_i ((\mathit{init}\ w \wedge_i f) \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star$
by (*rule ChopstarEqv*)
have *2*: $\vdash (\mathit{init}\ w \wedge_i f) \wedge_i \mathit{more} \equiv_i \mathit{init}\ w \wedge_i (f \wedge_i \mathit{more}\ )$
by *auto*
hence *3*: $\vdash ((\mathit{init}\ w \wedge_i f) \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star \equiv_i (\mathit{init}\ w \wedge_i f \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star$
by (*rule LeftChopEqvChop*)
have *4*: $\vdash (\mathit{init}\ w \wedge_i f)^\star \equiv_i \mathit{empty} \vee_i (\mathit{init}\ w \wedge_i f \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star$
using *1 3* **by** (*meson EmptyImpCS itl-prop*(*31*) *prop30 prop19 prop02 prop14*)
have *5*: $\vdash (\mathit{init}\ w \wedge_i f)^\star \wedge_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w) \equiv_i$
$(\ \mathit{empty} \wedge_i \mathit{fin}\ \neg_i\ (\mathit{init}\ w)) \vee_i$
$((\mathit{init}\ w \wedge_i f \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star \wedge_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w))$
using *1 prop23* **using** *4* **by** *blast*
have *6*: $\vdash\ \mathit{empty} \wedge_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w) \equiv_i \neg_i\ (\ \mathit{init}\ w) \wedge_i \mathit{empty}$
using *AndFinEqvChopAndEmpty EmptyChop prop03* **by** *blast*
have *7*: $\vdash (\mathit{init}\ w \wedge_i f \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star \equiv_i \mathit{init}\ w \wedge_i (f \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star$
by (*rule StateAndChop*)
have *8*: $\vdash ((f \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star) \wedge_i \mathit{fin}\ (\ \mathit{init}\ \neg_i\ w) \equiv_i$
$(f \wedge_i \mathit{more}\ );\ ((\mathit{init}\ w \wedge_i f)^\star \wedge_i \mathit{fin}\ (\ \mathit{init}\ \neg_i\ w))$
by (*rule ChopAndFin*)
have *81*: $\vdash \mathit{fin}\ (\ \mathit{init}\ \neg_i\ w) \equiv_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w)$
using *FinEqvFin Initprop*(*2*) *itl-prop*(*30*) **by** *blast*
have *82*: $\vdash (f \wedge_i \mathit{more}\ );\ (\mathit{init}\ w \wedge_i f)^\star \wedge_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w) \equiv_i$
$(f \wedge_i \mathit{more}\ );\ ((\mathit{init}\ w \wedge_i f)^\star \wedge_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w))$
using *8 81* **by** *auto*
have *9*: $\vdash (\mathit{init}\ w \wedge_i f)^\star \wedge_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w) \equiv_i$
$(\neg_i\ (\ \mathit{init}\ w) \wedge_i \mathit{empty}\ ) \vee_i$
$(\mathit{init}\ w \wedge_i (f \wedge_i \mathit{more}\ );\ ((\mathit{init}\ w \wedge_i f)^\star \wedge_i \mathit{fin}\ \neg_i\ (\ \mathit{init}\ w)))$
using *5 6 7 82 prop24* **by** *blast*
from *9* **show** *?thesis* **by** (*metis while-d-def*)
qed

lemma *WhileIntro*:
assumes $\vdash\ \neg_i\ (\ \mathit{init}\ w) \wedge_i f \supset_i \mathit{empty}$
$\vdash \mathit{init}\ w \wedge_i f \supset_i (g \wedge_i \mathit{more}\ );\ f$
shows $\vdash f \supset_i \mathit{while}\ (\ \mathit{init}\ w)\ \mathit{do}\ g$
proof –
have *1*: $\vdash\ \neg_i\ (\ \mathit{init}\ w) \wedge_i f \supset_i \mathit{empty}$
using *assms* **by** *blast*
have *2*: $\vdash \mathit{init}\ w \wedge_i f \supset_i (g \wedge_i \mathit{more}\ );\ f$
using *assms* **by** *blast*
have *3*: $\vdash\ \mathit{while}\ (\ \mathit{init}\ w)\ \mathit{do}\ g \equiv_i$
$(\neg_i\ (\mathit{init}\ w) \wedge_i \mathit{empty}\ ) \vee_i (\mathit{init}\ w \wedge_i (g \wedge_i \mathit{more}\ );\ \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)$
by (*rule WhileEqvEmptyOrChopWhile*)
hence *31*: $\vdash \neg_i\ (\ \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g) \equiv_i$
$\neg_i(\ (\neg_i\ (\mathit{init}\ w) \wedge_i \mathit{empty}\ ) \vee_i (\mathit{init}\ w \wedge_i (g \wedge_i \mathit{more}\ );\ \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g))$
using *itl-prop*(*33*) **by** *blast*

hence *32*: $\vdash f \wedge_i \neg_i (\mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g) \equiv_i$

$f \wedge_i \neg_i((\neg_i(\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i (\mathit{init}\ w \wedge_i (g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g))$

using *prop05* **by** *blast*

have *33*: $\vdash f \wedge_i \neg_i((\neg_i(\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i (\mathit{init}\ w \wedge_i (g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \equiv_i$

$f \wedge_i \neg_i(\neg_i(\mathit{init}\ w) \wedge_i \mathit{empty}) \wedge_i \neg_i(\mathit{init}\ w \wedge_i (g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)$

by *auto*

have *34*: $\vdash f \wedge_i \neg_i(\neg_i(\mathit{init}\ w) \wedge_i \mathit{empty}) \wedge_i \neg_i(\mathit{init}\ w \wedge_i (g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g) \equiv_i$

$f \wedge_i ((\mathit{init}\ w) \vee_i \mathit{more}) \wedge_i (\neg_i(\mathit{init}\ w) \vee_i \neg_i((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g))$

by *auto*

have *35*: $\vdash f \wedge_i ((\mathit{init}\ w) \vee_i \mathit{more}) \wedge_i (\neg_i(\mathit{init}\ w) \vee_i \neg_i((g \wedge_i \mathit{more});\mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \equiv_i$

$(f \wedge_i (\mathit{init}\ w) \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \vee_i$

$(f \wedge_i (\mathit{init}\ w) \wedge_i \neg_i(\mathit{init}\ w)) \vee_i$

$(f \wedge_i \mathit{more} \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \vee_i$

$(f \wedge_i \mathit{more} \wedge_i \neg_i(\mathit{init}\ w))$

by *auto*

have *36*: $\vdash \neg_i(f \wedge_i (\mathit{init}\ w) \wedge_i \neg_i(\mathit{init}\ w))$

by *auto*

have *37*: $\vdash \neg_i(f \wedge_i \mathit{more} \wedge_i \neg_i(\mathit{init}\ w))$

using *1* **by** *auto*

have *38*: $\vdash (f \wedge_i \mathit{more} \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \supset_i$

$((g \wedge_i \mathit{more}); f \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g))$

using *1 2* **by** *auto*

have *39*: $\vdash (f \wedge_i (\mathit{init}\ w) \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \supset_i$

$((g \wedge_i \mathit{more}); f \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g))$

using *2* **by** *auto*

have *40*: $\vdash ((f \wedge_i (\mathit{init}\ w) \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \vee_i$

$(f \wedge_i (\mathit{init}\ w) \wedge_i \neg_i(\mathit{init}\ w)) \vee_i$

$(f \wedge_i \mathit{more} \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)) \vee_i$

$(f \wedge_i \mathit{more} \wedge_i \neg_i(\mathit{init}\ w))) \supset_i$

$(g \wedge_i \mathit{more}); f \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)$

using *39 38 37 38* **by** *auto*

have *4*: $\vdash f \wedge_i \neg_i (\mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g) \supset_i$

$(g \wedge_i \mathit{more}); f \wedge_i \neg_i ((g \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ g)$

using *32 33 34 35 40* **by** *auto*

have *5*: $\vdash g \wedge_i \mathit{more} \supset_i \mathit{more}$

by *auto*

from *4 5* **show** *?thesis* **using** *ChopContra* **by** *blast*

qed

lemma *WhileElim*:

assumes $\vdash \neg_i (\mathit{init}\ w) \wedge_i \mathit{empty} \supset_i g$

$\vdash \mathit{init}\ w \wedge_i (f \wedge_i \mathit{more}); g \supset_i g$

shows $\vdash \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ f \supset_i g$

proof −

have *1*: $\vdash \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ f \equiv_i$

$(\neg_i (\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i (\mathit{init}\ w \wedge_i (f \wedge_i \mathit{more}); \mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ f)$

by (*rule WhileEqvEmptyOrChopWhile*)

hence *11*: $\vdash (\mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ f) \wedge_i \neg_i g \equiv_i$

$((\neg_i(\mathit{init}\ w) \wedge_i \mathit{empty}) \vee_i (\mathit{init}\ w \wedge_i (f \wedge_i \mathit{more});\mathit{while}\ (\mathit{init}\ w)\ \mathit{do}\ f)) \wedge_i \neg_i g$

using *prop06* **by** *blast*

```
 have  2: ⊢ ¬ᵢ ( init w) ∧ᵢ  empty  ⊃ᵢ g
       using assms by blast
 hence 21: ⊢ ¬ᵢ g ⊃ᵢ ¬ᵢ(¬ᵢ ( init w) ∧ᵢ  empty)
       by auto
 have  22: ⊢  ((¬ᵢ (init w) ∧ᵢ empty) ∨ᵢ (init w ∧ᵢ (f ∧ᵢ more);while (init w) do f)) ∧ᵢ ¬ᵢ  g ⊃ᵢ
             (init w ∧ᵢ (f ∧ᵢ  more );  while ( init w)  do  f)
       using 21 by auto
 have  23: ⊢ (while ( init w)  do  f) ∧ᵢ ¬ᵢ g ⊃ᵢ
             (init w ∧ᵢ (f ∧ᵢ  more );  while ( init w)  do  f) ∧ᵢ ¬ᵢ  g
       using 11 21 by auto
 have   3: ⊢ (init w) ∧ᵢ ((f ∧ᵢ  more ); g) ⊃ᵢ g
       using assms by blast
 hence 31: ⊢ ¬ᵢ g ⊃ᵢ ¬ᵢ((init w) ∧ᵢ ((f ∧ᵢ  more ); g))
       using prop27 by blast
 have  32: ⊢ (init w ∧ᵢ (f ∧ᵢ  more );  while ( init w)  do  f) ∧ᵢ ¬ᵢ g ⊃ᵢ
            (((f ∧ᵢ  more );  (while  (init w)  do  f)) ∧ᵢ ¬ᵢ ((f ∧ᵢ  more ); g)) ∧ᵢ ¬ᵢg
       using 31 by auto
 have   4: ⊢  (while  (init w)  do  f) ∧ᵢ ¬ᵢ  g ⊃ᵢ
             ((f ∧ᵢ  more );  (while  (init w)  do  f)) ∧ᵢ ¬ᵢ ((f ∧ᵢ  more ); g)
       using 23 32 by auto
 have   5: ⊢ f ∧ᵢ  more ⊃ᵢ  more
       by auto
 from 4 5 show ?thesis using ChopContra by blast
qed

lemma BaWhileImpWhile:
⊢   ba (f ⊃ᵢ g) ⊃ᵢ ( while  (init w)  do  f) ⊃ᵢ ( while  (init w)  do  g)
proof −
 have  1: ⊢ (f ⊃ᵢ g) ⊃ᵢ ((init w ∧ᵢ f)  ⊃ᵢ (init w ∧ᵢ g))
      by auto
 hence 2: ⊢  ba (f ⊃ᵢ g) ⊃ᵢ  ba ((init w ∧ᵢ f)  ⊃ᵢ (init w∧ᵢ g))
      by (rule BaImpBa)
 have  3:  ⊢  ba ((init w ∧ᵢ f)  ⊃ᵢ (init w∧ᵢ g)) ⊃ᵢ ((init w∧ᵢ f)⋆ ⊃ᵢ (init w∧ᵢ g)⋆)
      by (rule BaCSImpCS)
 have  4: ⊢  ba (f ⊃ᵢ g) ⊃ᵢ ((init w∧ᵢ f)⋆∧ᵢ fin ¬ᵢ ( init w) ⊃ᵢ (init w∧ᵢ g)⋆∧ᵢ fin ¬ᵢ (init w))
      using 2 3 by auto
 from 4 show ?thesis by (simp add: while-d-def)
qed

lemma WhileImpWhile:
 assumes ⊢   f⊃ᵢ g
 shows ⊢ ( while  (init w)  do  f) ⊃ᵢ ( while  (init w)  do  g)
proof −
 have  1: ⊢ f ⊃ᵢ g
      using assms by auto
 hence 2: ⊢  ba (f ⊃ᵢ g)
      by (rule BaGen)
 have  3: ⊢  ba (f ⊃ᵢ g) ⊃ᵢ ( while  (init w)  do  f) ⊃ᵢ ( while  (init w)  do  g)
      by (rule BaWhileImpWhile)
 from 2 3 show ?thesis using MP by blast
```

qed

5.9 Properties of Halt

```
lemma WnextAndMoreEqvNext:
⊢ wnext f ∧ᵢ more ≡ᵢ ○ f
by auto

lemma BoxStateAndEmptyEqvStateAndEmpty:
⊢ □(empty ≡ᵢ (init w)) ∧ᵢ empty ≡ᵢ (init w) ∧ᵢ empty
apply simp-all
by auto

lemma BoxEmptyEqvIStateqvEmptyAndStateOrNotStateNext:
⊢ □(empty ≡ᵢ (init w)) ≡ᵢ (empty ∧ᵢ init w) ∨ᵢ (¬ᵢ(init w) ∧ᵢ ○(□(empty ≡ᵢ (init w))))
proof −
 have 1: ⊢ □(empty ≡ᵢ (init w)) ≡ᵢ
          (□(empty ≡ᵢ (init w)) ∧ᵢ empty) ∨ᵢ (□(empty ≡ᵢ (init w)) ∧ᵢ more)
      by auto
 have 2: ⊢ □(empty ≡ᵢ (init w)) ∧ᵢ empty ≡ᵢ (init w) ∧ᵢ empty
      using BoxStateAndEmptyEqvStateAndEmpty by blast
 have 3: ⊢ □(empty ≡ᵢ (init w))  ≡ᵢ (empty ≡ᵢ (init w)) ∧ᵢ wnext(□(empty ≡ᵢ (init w)))
      using BoxEqvAndWnextBox by blast
 hence 4: ⊢ □(empty ≡ᵢ (init w)) ∧ᵢ more  ≡ᵢ
             (empty ≡ᵢ (init w)) ∧ᵢ wnext(□(empty ≡ᵢ (init w))) ∧ᵢ more
      by auto
 have 5: ⊢ (empty ≡ᵢ (init w)) ∧ᵢ more ≡ᵢ ¬ᵢ(init w) ∧ᵢ more
      by auto
 have 6: ⊢ wnext(□(empty ≡ᵢ (init w))) ∧ᵢ more ≡ᵢ ○(□(empty ≡ᵢ (init w)))
      using WnextAndMoreEqvNext by auto
 from 1 2 4 5 6 show ?thesis by auto
qed

lemma HaltStateEqvIfStateThenEmptyElseNext:
⊢ halt( init w) ≡ᵢ ifᵢ (init w) then empty else ( ○( halt ( init w)))
proof −
 have 1: ⊢ halt( init w) ≡ᵢ □(empty ≡ᵢ (init w))
     by (simp add: halt-d-def)
 have 2: ⊢ □(empty ≡ᵢ (init w)) ≡ᵢ
          (empty ∧ᵢ init w) ∨ᵢ (¬ᵢ(init w) ∧ᵢ ○(□(empty ≡ᵢ (init w))))
     by (rule BoxEmptyEqvIStateqvEmptyAndStateOrNotStateNext)
 have 21: ⊢ (empty ∧ᵢ init w) ∨ᵢ (¬ᵢ(init w) ∧ᵢ ○(□(empty ≡ᵢ (init w)))) ≡ᵢ
          (init w ∧ᵢ empty) ∨ᵢ (¬ᵢ(init w) ∧ᵢ ○(□(empty ≡ᵢ (init w))))
     by auto
 have 22: ⊢ ○(halt ( init w)) ≡ᵢ ○(□(empty ≡ᵢ (init w)))
     using NextEqvNext using 1 by blast
 have 3: ⊢ ifᵢ (init w) then empty else ( ○( halt ( init w))) ≡ᵢ
          (init w ∧ᵢ empty ) ∨ᵢ (¬ᵢ(init w) ∧ᵢ ○( halt ( init w)))
     by (simp add: ifthenelse-d-def)
 from 1 2 21 22 3 show ?thesis by (simp add: halt-d-def)
```

```
qed

lemma HaltChopEqv:
⊢ ((halt ( init w)) ; f) ≡_i (if_i (init w) then ( f ) else (○( (halt ( init w)); f)))
proof −
 have 1: ⊢ halt(init w) ≡_i
          (if_i (init w) then empty else ( ○( halt ( init w))))
      by (rule HaltStateEqvIfStateThenEmptyElseNext)
 hence 2: ⊢ ((halt(init w));f) ≡_i
          (if_i (init w) then (empty;f) else ( ○( halt ( init w));f))
      by (rule IfChopEqvRule)
 have 3: ⊢ empty ; f ≡_i f
      by (rule EmptyChop)
 have 4: ⊢ (○ (halt ( init w))); f ≡_i ○( halt ( init w); f)
      by (rule NextChop)
 from 2 3 4 show ?thesis using prop07 prop03 by blast
qed

lemma AndHaltChopImp:
⊢ init w ∧_i ( halt ( init w); f) ⊃_i f
proof −
 have 1: ⊢ halt ( init w); f ≡_i if_i (init w) then f else ( ○( halt ( init w); f))
     by (rule HaltChopEqv)
 have 2: ⊢ init w ∧_i if_i (init w) then f else ( ○( halt ( init w); f)) ⊃_i f
     by (simp add: ifthenelse-d-def and-d-def)
 from 1 2 show ?thesis by auto
qed

lemma NotAndHaltChopImpNext:
⊢ ¬_i ( init w) ∧_i ( halt (init w); f) ⊃_i ○( halt ( init w); f)
proof −
 have 1: ⊢ halt ( init w); f ≡_i if_i (init w) then f else ( ○( halt ( init w); f))
     by (rule HaltChopEqv)
 have 2: ⊢ ¬_i ( init w) ∧_i if_i (init w) then f else ( ○( halt ( init w); f)) ⊃_i
         ○( halt ( init w); f)
     by auto
from 1 2 show ?thesis by auto
qed

lemma NotAndHaltChopImpSkipYields:
⊢ ¬_i ( init w) ∧_i ( halt ( init w); f) ⊃_i skip yields ( halt (init w); f)
proof −
 have 1: ⊢ ¬_i ( init w) ∧_i ( halt ( init w); f) ⊃_i ○( halt ( init w); f)
     by (rule NotAndHaltChopImpNext)
 have 2: ⊢ ○( halt ( init w); f) ⊃_i skip yields ( halt ( init w); f)
     by (rule NextImpSkipYields)
 from 1 2 show ?thesis by auto
qed

lemma TrueChopAndEmptyEqvChopAndEmpty:
```

$\vdash (\mathit{true}_i;(f \wedge_i \mathit{empty})) \wedge_i g \equiv_i (g;(f \wedge_i \mathit{empty}))$
by *force*

lemma *WprevEqvEmptyOrPrev*:
$\vdash \mathit{wprev}\ f \equiv_i \mathit{empty} \vee_i \mathit{prev}\ f$
by *auto*

lemma *NotChopSkipEqvMoreAndNotChopSkip*:
$\vdash (\neg_i\ f);\mathit{skip} \equiv_i \mathit{more} \wedge_i \neg_i(f;\mathit{skip})$
using *WprevEqvEmptyOrPrev*
by (*metis* (*full-types*) *and-d-def empty-d-def*
itl-prop(*30*) *itl-prop*(*33*) *itl-prop*(*4*) *prev-d-def prop03 prop28 wprev-d-def*)

lemma *HaltChopImpNotHaltChopNot*:
$\vdash \mathit{halt}\ (\mathit{init}\ w);\ f \supset_i \neg_i\ (\mathit{halt}\ (\mathit{init}\ w);\ \neg_i\ f)$
proof −
have *1*: $\vdash \mathit{halt}\ (\mathit{init}\ w);\ f \equiv_i \mathit{if}_i\ (\mathit{init}\ w)\ \mathit{then}\ f\ \mathit{else}\ (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f))$
by (*rule HaltChopEqv*)
have *2*: $\vdash \mathit{if}_i\ (\mathit{init}\ w)\ \mathit{then}\ f\ \mathit{else}\ (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f)) \supset_i$
$(((\mathit{init}\ w) \supset_i f) \wedge_i (\neg_i(\mathit{init}\ w) \supset_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f))))$
by (*rule prop11*)
have *3*: $\vdash \mathit{halt}\ (\mathit{init}\ w);\ \neg_i f \equiv_i$
$\mathit{if}_i\ (\mathit{init}\ w)\ \mathit{then}\ \neg_i f\ \mathit{else}\ (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f))$
by (*rule HaltChopEqv*)
have *4*: $\vdash \mathit{if}_i\ (\mathit{init}\ w)\ \mathit{then}\ \neg_i f\ \mathit{else}\ (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f)) \supset_i$
$(((\mathit{init}\ w) \supset_i \neg_i f) \wedge_i (\neg_i(\mathit{init}\ w) \supset_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f))))$
by (*rule prop11*)
have *5*: $\vdash \mathit{halt}\ (\mathit{init}\ w);\ f \wedge_i \mathit{halt}\ (\mathit{init}\ w);\ \neg_i f \supset_i$
$(((\mathit{init}\ w) \supset_i f) \wedge_i (\neg_i(\mathit{init}\ w) \supset_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f)))) \wedge_i$
$(((\mathit{init}\ w) \supset_i \neg_i f) \wedge_i (\neg_i(\mathit{init}\ w) \supset_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f))))$
using *1 2 3 4* **by** *auto*
have *6*: $\vdash (((\mathit{init}\ w) \supset_i f) \wedge_i (\neg_i(\mathit{init}\ w) \supset_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f)))) \wedge_i$
$(((\mathit{init}\ w) \supset_i \neg_i f) \wedge_i (\neg_i(\mathit{init}\ w) \supset_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f)))) \supset_i$
$(\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f)) \wedge_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f))$
by *auto*
have *7*: $\vdash \mathit{halt}\ (\mathit{init}\ w);\ f \wedge_i \mathit{halt}\ (\mathit{init}\ w);\ \neg_i f \supset_i$
$(\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f)) \wedge_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f))$
using *5 6 prop02* **by** *blast*
have *8*: $\vdash (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ f)) \wedge_i (\bigcirc(\mathit{halt}\ (\mathit{init}\ w);\ \neg_i f)) \equiv_i$
$\bigcirc\ (\mathit{halt}\ (\mathit{init}\ w);\ f \wedge_i \mathit{halt}\ (\mathit{init}\ w);\ \neg_i f)$
by *auto*
have *9*: $\vdash \mathit{halt}\ (\mathit{init}\ w);\ f \wedge_i \mathit{halt}\ (\mathit{init}\ w);\ \neg_i f \supset_i$
$\bigcirc\ (\mathit{halt}\ (\mathit{init}\ w);\ f \wedge_i \mathit{halt}\ (\mathit{init}\ w);\ \neg_i f)$
using *7 8 itl-prop*(*31*) *prop02* **by** *blast*
hence *10*: $\vdash \neg_i(\mathit{halt}\ (\mathit{init}\ w);\ f \wedge_i \mathit{halt}\ (\mathit{init}\ w);\ \neg_i f)$
using *NextLoop* **by** *blast*
from *10* **show** *?thesis* **by** *auto*
qed

lemma *HaltChopImpHaltYields*:
$\vdash\ \ \mathit{halt}\ (\ \mathit{init}\ w);\ f \supset_i (\ \mathit{halt}\ (\ \mathit{init}\ w))\ \mathit{yields}\ \ f$
proof –
have *1*: $\vdash\ \mathit{halt}\ (\ \mathit{init}\ w);\ f \supset_i \neg_i\ (\ \mathit{halt}\ (\ \mathit{init}\ w);\ \neg_i\ \ f)$ **by** (*rule HaltChopImpNotHaltChopNot*)
from *1* **show** *?thesis* **by** (*simp add*: *yields-d-def*)
qed

lemma *HaltChopAnd*:
$\vdash (\ \mathit{halt}\ (\mathit{init}\ w));\ f \wedge_i (\ \mathit{halt}\ \ (\mathit{init}\ w));\ g \supset_i (\ \mathit{halt}\ (\ \mathit{init}\ w));\ (f \wedge_i g)$
proof –
have *1*: $\vdash (\ \mathit{halt}\ (\mathit{init}\ w));\ g \supset_i (\ \mathit{halt}\ (\mathit{init}\ w))\ \ \mathit{yields}\ \ g$ **by** (*rule HaltChopImpHaltYields*)
hence *2*: $\vdash (\ \mathit{halt}\ \ (\mathit{init}\ w));\ f \wedge_i (\ \mathit{halt}\ \ (\mathit{init}\ w));\ g \supset_i$
$(\ \mathit{halt}\ \ (\mathit{init}\ w));\ f \wedge_i (\ \mathit{halt}\ \ (\mathit{init}\ w))\ \mathit{yields}\ \ g$ **by** *auto*
have *3*: $\vdash (\ \mathit{halt}\ \ (\mathit{init}\ w));\ f \wedge_i (\ \mathit{halt}\ \ (\mathit{init}\ w))\ \mathit{yields}\ \ g \supset_i$
$(\ \mathit{halt}\ \ (\mathit{init}\ w));\ (f \wedge_i g)$ **by** (*rule ChopAndYieldsImp*)
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *HaltAndChopAndHaltChopImpHaltAndChopAnd*:
$\vdash\ \ (\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ \mathit{f1} \wedge_i (\ \mathit{halt}\ \ (\mathit{init}\ w);\ g) \supset_i (\ \mathit{halt}\ (\ \mathit{init}\ w) \wedge_i f);\ (\mathit{f1} \wedge_i g)$
proof –
have *1*: $\vdash \mathit{f1} \supset_i \neg_i\ \ g \vee_i\ \ (\mathit{f1} \wedge_i g)$
by *auto*
hence *2*: $\vdash (\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ \mathit{f1} \supset_i$
$(\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ \neg_i\ \ g \vee_i\ \ ((\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ (\mathit{f1} \wedge_i g))$
by (*rule ChopOrImpRule*)
have *3*: $\vdash (\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ \neg_i\ \ g \supset_i\ \ \mathit{halt}\ \ (\mathit{init}\ w);\ \neg_i\ \ g$
by (*rule AndChopA*)
have *31*: $\vdash (\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ \mathit{f1} \supset_i$
$\mathit{halt}\ \ (\mathit{init}\ w);\ \neg_i\ \ g \vee_i\ \ ((\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ (\mathit{f1} \wedge_i g))$
using *2 3* **by** *auto*
have *4*: $\vdash\ \ \mathit{halt}\ \ (\mathit{init}\ w);\ g \supset_i \neg_i\ (\ \mathit{halt}\ \ (\mathit{init}\ w);\ \neg_i\ \ g)$
by (*rule HaltChopImpNotHaltChopNot*)
hence *41*: $\vdash (\ \mathit{halt}\ \ (\mathit{init}\ w);\ \neg_i\ \ g) \supset_i \neg_i(\mathit{halt}\ \ (\mathit{init}\ w);\ g)$
by *auto*
have *42*: $\vdash (\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ \mathit{f1} \supset_i$
$\neg_i(\ \mathit{halt}\ \ (\mathit{init}\ w);\ g) \vee_i\ \ ((\ \mathit{halt}\ \ (\mathit{init}\ w) \wedge_i f);\ (\mathit{f1} \wedge_i g))$
using *31 41* **by** *auto*
from *42* **show** *?thesis* **by** *auto*
qed

lemma *HaltImpBoxYields*:
$\vdash\ \ (\ \mathit{halt}\ \ (\mathit{init}\ w));\ f \supset_i (\Box\neg_i\ (\ \mathit{init}\ w))\ \ \mathit{yields}\ \ ((\ \mathit{halt}\ \ (\mathit{init}\ w));\ f)$
proof –
have *1*: $\vdash (\Box\ \neg_i\ \ (\mathit{init}\ w));\ \neg_i\ (\ \mathit{halt}\ \ (\mathit{init}\ w);\ f) \supset_i\ \ \mathit{di}\ \ (\Box\ \neg_i\ \ (\mathit{init}\ w))$
by (*rule ChopImpDi*)
have *2*: $\vdash \Box\ \neg_i\ \ (\mathit{init}\ w) \supset_i \neg_i\ \ (\mathit{init}\ w)$
by (*rule BoxElim*)
hence *3*: $\vdash\ \ \mathit{di}\ (\Box\ \neg_i\ \ (\mathit{init}\ w)) \supset_i\ \ \mathit{di}\ \neg_i\ \ (\mathit{init}\ w)$

```
     by (rule DiImpDi)
  have  4: ⊢ di (init ¬_i w) ≡_i (init ¬_i w)
     by (rule DiState)
  have 41: ⊢ (init ¬_i w) ≡_i ¬_i (init w)
     by auto
  have 42: ⊢ di ¬_i (init w) ≡_i ¬_i(init w)
     using 4 41 by auto
  have  5: ⊢ ((□¬_i (init w)); ¬_i ( halt (init w); f)) ⊃_i ¬_i ( init w)
     using 1 2 42 using 3 itl-prop(31) prop02 by blast
  hence 51: ⊢ ( halt (init w); f) ∧_i ((□¬_i (init w)); ¬_i ( halt (init w); f)) ⊃_i
         ( halt (init w); f) ∧_i ¬_i ( init w)
     using prop12 by blast
  have  6: ⊢ halt (init w); f ≡_i if_i (init w) then f else (○( halt (init w); f))
     by (rule HaltChopEqv)
  hence 61: ⊢ halt (init w); f ∧_i ¬_i ( init w) ≡_i
          (if_i (init w) then f else (○( halt (init w); f))) ∧_i ¬_i ( init w)
     using 6 by auto
  have 62: ⊢ (if_i (init w) then f else (○( halt (init w); f))) ∧_i
          ¬_i ( init w) ⊃_i (○( halt (init w); f))
     by auto
  have 63: ⊢ halt (init w); f ∧_i ¬_i ( init w) ⊃_i (○( halt (init w); f))
     using 61 62 by auto
  have  7: ⊢ ( halt (init w); f) ∧_i (□¬_i (init w)); ¬_i ( halt (init w); f) ⊃_i
         ○(( halt (init w)); f)
     using 51 63 using prop02 by blast
  have  8: ⊢ □ ¬_i (init w) ⊃_i empty ∨_i ○(□¬_i( init w))
     by auto
  hence  9: ⊢ ((□ ¬_i ( init w)); ¬_i ( halt (init w); f)) ⊃_i
         ¬_i ( halt (init w); f) ∨_i ○((□¬_i (init w)); ¬_i ( halt (init w); f))
     by (rule EmptyOrNextChopImpRule)
  hence 10: ⊢ (( halt (init w)); f) ∧_i (□ ¬_i (init w)); ¬_i ( halt (init w); f) ⊃_i
         ○((□¬_i (init w)); ¬_i ( halt (init w); f))
     using prop13 by blast
  have 11: ⊢ ( halt (init w)); f ∧_i (□ ¬_i (init w)); ¬_i ( halt (init w); f) ⊃_i
        ○(( halt (init w)); f) ∧_i ○((□¬_i (init w)); ¬_i ( halt (init w); f))
     using 7 10 by auto
  have 12: ⊢ ○(( halt (init w)); f) ∧_i ○((□¬_i (init w)); ¬_i ( halt (init w); f))
        ⊃_i ○((( halt (init w)); f) ∧_i ((□¬_i (init w)); ¬_i ( halt (init w); f)))
     by auto
  have 13: ⊢ ( halt (init w)); f ∧_i (□ ¬_i (init w)); ¬_i ( halt (init w); f) ⊃_i
         ○((( halt (init w)); f) ∧_i ((□¬_i (init w)); ¬_i ( halt (init w); f)))
     using 11 12 by auto
  hence 14: ⊢ ¬_i (( halt (init w)); f ∧_i (□ ¬_i (init w)); ¬_i ( halt (init w); f))
     using NextLoop by blast
  hence 15: ⊢ ( halt (init w)); f ⊃_i ¬_i ((□ ¬_i (init w)); ¬_i ( halt (init w); f))
     by auto
  from 15 show ?thesis by (simp add: yields-d-def)
qed
```

5.10 Properties of Groups of chops

lemma *NestedChopImpChop*:
assumes $\vdash\ \mathit{init}\ w \wedge_i f \supset_i g; (\mathit{init}\ w1 \wedge_i f1)$
$\vdash \mathit{init}\ w1 \wedge_i f1 \supset_i g1; (\mathit{init}\ w2 \wedge_i f2)$
shows $\vdash \mathit{init}\ w \wedge_i f \supset_i g; (g1; (\mathit{init}\ w2 \wedge_i f2))$
proof –
have *1*: $\vdash \mathit{init}\ w \wedge_i f \supset_i g; (\mathit{init}\ w1 \wedge_i f1)$ **using** *assms*(*1*) **by** *auto*
have *2*: $\vdash \mathit{init}\ w1 \wedge_i f1 \supset_i g1; (\mathit{init}\ w2 \wedge_i f2)$ **using** *assms*(*2*) **by** *auto*
hence *3*: $\vdash g; (\mathit{init}\ w1 \wedge_i f1) \supset_i g; (g1; (\mathit{init}\ w2 \wedge_i f2))$ **by** (*rule RightChopImpChop*)
from *1 3* **show** *?thesis* **by** *auto*
qed

5.11 Properties of Time Reversal

lemma *RNot*:
$\vdash (\neg_i f)^r \equiv_i \neg_i f^r$
by *simp*

lemma *RRNot*:
$\vdash (\neg_i (f^r))^r \equiv_i \neg_i f$
by (*metis EqvReverseReverse not-d-def rev-d.simps*(*1*) *rev-d.simps*(*3*))

lemma *RTrue*:
$\vdash (\mathit{true}_i)^r \equiv_i \mathit{true}_i$
by (*simp add*: *not-d-def true-d-def*)

lemma *ROr*:
$\vdash (f \vee_i g)^r \equiv_i f^r \vee_i g^r$
by (*simp add*: *not-d-def or-d-def*)

lemma *RROr*:
$\vdash (f^r \vee_i g^r)^r \equiv_i f \vee_i g$
proof –
have *1*: $\vdash (f^r \vee_i g^r)^r \equiv_i (f^r)^r \vee_i (g^r)^r$ **using** *ROr* **by** *blast*
have *2*: $\vdash (f^r)^r \vee_i (g^r)^r \equiv_i f \vee_i g$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RAnd*:
$\vdash (f \wedge_i g)^r \equiv_i f^r \wedge_i g^r$
by (*simp add*: *and-d-def not-d-def or-d-def*)

lemma *RRAnd*:
$\vdash (f^r \wedge_i g^r)^r \equiv_i f \wedge_i g$
proof –
have *1*: $\vdash (f^r \wedge_i g^r)^r \equiv_i (f^r)^r \wedge_i (g^r)^r$ **using** *RAnd* **by** *blast*
have *2*: $\vdash (f^r)^r \wedge_i (g^r)^r \equiv_i f \wedge_i g$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *REqvRule*:
assumes $\vdash f \equiv_i g$
shows $\vdash f^r \equiv_i g^r$
using *assms*
by (*metis ReverseEqv itl-prop*(*31*) *rev-d.simps*(*3*))

lemma *RImpRule*:
assumes $\vdash f \supset_i g$
shows $\vdash f^r \supset_i g^r$
using *assms*
by (*metis ReverseEqv rev-d.simps*(*3*))

lemma *RNextEqvPrev*:
$\vdash (\bigcirc\ f)^r \equiv_i prev\ (f^r)$
by (*simp add*: *next-d-def prev-d-def*)

lemma *RRNextEqvPrev*:
$\vdash (\bigcirc\ (f^r))^r \equiv_i prev\ (f)$
proof −
have *1*: $\vdash (\bigcirc\ (f^r))^r \equiv_i prev\ ((f^r)^r)$ **using** *RNextEqvPrev* **by** *blast*
have *2*: $\vdash prev\ ((f^r)^r) \equiv_i prev\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RWNextEqvWPrev*:
$\vdash (wnext\ f)^r \equiv_i wprev(f^r)$
by (*simp add*: *next-d-def not-d-def prev-d-def wnext-d-def wprev-d-def*)

lemma *RRWNextEqvWPrev*:
$\vdash (wnext\ (f^r))^r \equiv_i wprev(f)$
proof −
have *1*: $\vdash (wnext\ (f^r))^r \equiv_i wprev\ ((f^r)^r)$ **using** *RWNextEqvWPrev* **by** *blast*
have *2*: $\vdash wprev\ ((f^r)^r) \equiv_i wprev\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RPrevEqvNext*:
$\vdash (prev\ f)^r \equiv_i \bigcirc\ (f^r)$
by (*simp add*: *next-d-def prev-d-def*)

lemma *RRPrevEqvNext*:
$\vdash (prev\ (f^r))^r \equiv_i \bigcirc\ (f)$
proof −
have *1*: $\vdash (prev\ (f^r))^r \equiv_i \bigcirc\ ((f^r)^r)$ **using** *RPrevEqvNext* **by** *blast*
have *2*: $\vdash \bigcirc\ ((f^r)^r) \equiv_i \bigcirc\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RWPrevEqvWNext*:
$\vdash (wprev\ f)^r \equiv_i wnext(f^r)$

by (*simp add*: *next-d-def not-d-def prev-d-def wnext-d-def wprev-d-def*)

lemma *RRWPrevEqvWNext*:
$\vdash (wprev\ (f^r))^r \equiv_i wnext(f)$
proof −
have *1*: $\vdash (wprev\ (f^r))^r \equiv_i wnext\ ((f^r)^r)$ **using** *RWPrevEqvWNext* **by** *blast*
have *2*: $\vdash wnext\ ((f^r)^r) \equiv_i wnext\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RDiamondEqvDi*:
$\vdash (\Diamond f)^r \equiv_i di\ (f^r)$
by (*metis RTrue RightChopEqvChop di-d-def rev-d.simps*(*5*) *sometimes-d-def*)

lemma *RRDiamondEqvDi*:
$\vdash (\Diamond(f^r))^r \equiv_i di\ (f)$
proof −
have *1*: $\vdash (\Diamond\ (f^r))^r \equiv_i di\ ((f^r)^r)$ **using** *RDiamondEqvDi* **by** *blast*
have *2*: $\vdash di\ ((f^r)^r) \equiv_i di\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RBoxEqvBi*:
$\vdash (\Box\ f)^r \equiv_i bi\ (f^r)$
using *RDiamondEqvDi*
by (*simp add*: *always-d-def not-d-def sometimes-d-def true-d-def*)

lemma *RRBoxEqvBi*:
$\vdash (\Box\ (f^r))^r \equiv_i bi\ (f)$
proof −
have *1*: $\vdash (\Box\ (f^r))^r \equiv_i bi\ ((f^r)^r)$ **using** *RBoxEqvBi* **by** *blast*
have *2*: $\vdash bi\ ((f^r)^r) \equiv_i bi\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RDiEqvDiamond*:
$\vdash (di\ f)^r \equiv_i \Diamond\ (f^r)$
by (*metis RTrue LeftChopEqvChop di-d-def rev-d.simps*(*5*) *sometimes-d-def*)

lemma *RRDiEqvDiamond*:
$\vdash (di\ (f^r))^r \equiv_i \Diamond\ (f)$
proof −
have *1*: $\vdash (di\ (f^r))^r \equiv_i \Diamond\ ((f^r)^r)$ **using** *RDiEqvDiamond* **by** *blast*
have *2*: $\vdash \Diamond\ ((f^r)^r) \equiv_i \Diamond\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RBiEqvBox*:
$\vdash (bi\ f)^r \equiv_i \Box\ (f^r)$
using *RDiEqvDiamond*

by (*simp add*: *bi-d-def di-d-def not-d-def true-d-def*)

lemma *RRBiEqvBox*:
$\vdash (bi\ (f^r))^r \equiv_i \Box\ (f)$
proof –
have *1*: $\vdash (bi\ (f^r))^r \equiv_i \Box\ ((f^r)^r)$ **using** *RBiEqvBox* **by** *blast*
have *2*: $\vdash \Box\ ((f^r)^r) \equiv_i \Box\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RDaEqvDa*:
$\vdash (da\ f)^r \equiv_i da(f^r)$
by (*metis ChopAssoc da-d-def itl-prop*(*30*) *not-d-def rev-d.simps*(*1*)
rev-d.simps(*3*) *rev-d.simps*(*5*) *true-d-def*)

lemma *RRDaEqvDa*:
$\vdash (da\ (f^r))^r \equiv_i da(f)$
proof –
have *1*: $\vdash (da\ (f^r))^r \equiv_i da\ ((f^r)^r)$ **using** *RDaEqvDa* **by** *blast*
have *2*: $\vdash da\ ((f^r)^r) \equiv_i da\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RBaEqvBa*:
$\vdash (ba\ f)^r \equiv_i ba(f^r)$
using *RDaEqvDa*
by (*metis ba-d-def itl-prop*(*33*) *not-d-def rev-d.simps*(*1*) *rev-d.simps*(*3*))

lemma *RRBaEqvBa*:
$\vdash (ba\ (f^r))^r \equiv_i ba(f)$
proof –
have *1*: $\vdash (ba\ (f^r))^r \equiv_i ba\ ((f^r)^r)$ **using** *RBaEqvBa* **by** *blast*
have *2*: $\vdash ba\ ((f^r)^r) \equiv_i ba\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *ChopCsImpCSChop*:
$\vdash f;f^\star \supset_i f^\star;f$
by (*meson CSChopEqvChopOrRule CSChopEqvOrChopPlusChop ChopAssocB*
ChopPlusElimWithoutMore EmptyYields prop19 prop21 prop28)

lemma *CSChopImpChopCS*:
$\vdash f^\star;f \supset_i f;f^\star$
proof –
have *1*: $\vdash (f^r);(f^r)^\star \supset_i (f^r)^\star;(f^r)$ **using** *ChopCsImpCSChop* **by** *blast*
hence *2*: $\vdash ((f^r);(f^r)^\star \supset_i (f^r)^\star;(f^r)\ )^r$ **using** *ReverseEqv* **by** *blast*
have *3*: $\vdash ((f^r);(f^r)^\star \supset_i (f^r)^\star;(f^r)\ )^r \equiv_i (\ ((f^r);(f^r)^\star)^r \supset_i ((f^r)^\star;(f^r))^r)$ **by** *simp*
have *4*: $\vdash ((f^r);(f^r)^\star)^r \equiv_i ((f^r)^\star\ )^r;\ (f^r)^r$ **by** *simp*
have *5*: $\vdash ((f^r)^\star\ )^r;\ (f^r)^r \equiv_i ((f^r)^r)^\star;(f^r)^r$ **by** *simp*
have *6*: $\vdash (f^r)^r \equiv_i f$ **using** *EqvReverseReverse itl-prop*(*30*) **by** *blast*

have *7*: $\vdash ((f^r)^r)^\star;(f^r)^r \equiv_i f^\star;f$ **using** *6 CSEqvCS ChopEqvChop* **by** *blast*
have *8*: $\vdash ((f^r);(f^r)^\star)^r \equiv_i f^\star;f$ **using** *7 5* **by** *auto*
have *9*: $\vdash ((f^r)^\star;(f^r))^r \equiv_i (f^r)^r;((f^r)^\star)^r$ **by** *simp*
have *10*: $\vdash (f^r)^r;((f^r)^\star)^r \equiv_i (f^r)^r;\ ((f^r)^r)^\star$ **by** *simp*
have *11*: $\vdash (f^r)^r;\ ((f^r)^r)^\star \equiv_i f;f^\star$ **using** *6 ChopPlusEqvChopPlus* **by** *blast*
have *12*: $\vdash ((f^r);(f^r)^\star)^r \equiv_i f;f^\star$ **using** *9 10 11*
by (*metis 2 ChopCsImpCSChop itl-prop*(*31*) *prop03 rev-d.simps*(*3*) *rev-d.simps*(*5*) *rev-d.simps*(*6*))
from *2 3 8 12* **show** *?thesis* **by** *auto*
qed

lemma *CSChopEqvChopCS*:
$\vdash f;f^\star \equiv_i f^\star;f$
using *ChopCsImpCSChop CSChopImpChopCS* **using** *itl-prop*(*31*) **by** *blast*

lemma *TrueChopSkipEqvSkipChopTrue*:
$\vdash\ true_i;skip \equiv_i skip;true_i$
proof –
have *1*: $\vdash skip;skip^\star \equiv_i skip^\star;skip$ **using** *CSChopEqvChopCS* **by** *blast*
have *2*: $\vdash skip^\star \equiv_i true_i$ **using** *CSSkip* **by** *simp*
have *3*: $\vdash skip;skip^\star \equiv_i skip;true_i$ **using** *2* **using** *RightChopEqvChop* **by** *blast*
have *4*: $\vdash skip^\star;skip \equiv_i true_i;skip$ **using** *2* **using** *LeftChopEqvChop* **by** *blast*
from *1 3 4* **show** *?thesis* **using** *itl-prop*(*30*) *prop03* **by** *blast*
qed

lemma *RMoreEqvMore*:
$\vdash more^r \equiv_i more$
by (*metis TrueChopSkipEqvSkipChopTrue more-d-def next-d-def not-d-def rev-d.simps*(*1*)
rev-d.simps(*3*) *rev-d.simps*(*4*) *rev-d.simps*(*5*) *true-d-def*)

lemma *REmptyEqvEmpty*:
$\vdash empty^r \equiv_i empty$
by (*metis RMoreEqvMore empty-d-def not-d-def prop01 rev-d.simps*(*1*) *rev-d.simps*(*3*))

lemma *RInitEqvFin*:
$\vdash (init\ f)^r \equiv_i fin(f^r)$
proof –
have *1*: $\vdash (init\ f)^r \equiv_i ((f \wedge_i empty);true_i)^r$
by (*metis AndChopCommute REqvRule init-d-def*)
have *2*: $\vdash ((f \wedge_i empty);true_i)^r \equiv_i (true_i;(f \wedge_i empty)^r)$
using *RTrue* **by** *auto*
have *3*: $\vdash true_i;(f \wedge_i empty)^r \equiv_i true_i;(f^r \wedge_i empty)$
by (*meson ChopEqvChop RAnd REmptyEqvEmpty RTrue itl-prop*(*30*) *prop05 prop21*)
have *4*: $\vdash true_i;(f^r \wedge_i empty) \equiv_i fin(f^r)$
using *FinEqvTrueChopAndEmpty itl-prop*(*30*) **by** *blast*
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *RRInitEqvFin*:
$\vdash (init\ (f^r))^r \equiv_i fin(f)$
proof –

have *1*: $\vdash (\mathit{init}\ (f^r))^r \equiv_i \mathit{fin}\ ((f^r)^r)$ **using** *RInitEqvFin* **by** *blast*
have *2*: $\vdash \mathit{fin}\ ((f^r)^r) \equiv_i \mathit{fin}\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RFinEqvInit*:
$\vdash (\mathit{fin}\ f)^r \equiv_i \mathit{init}\ (f^r)$
proof −
have *1*: $\vdash \mathit{fin}\ f \equiv_i \mathit{true}_i;(f \wedge_i \mathit{empty})$
using *FinEqvTrueChopAndEmpty* **by** *auto*
have *2*: $\vdash (\mathit{fin}\ f)^r \equiv_i (\mathit{true}_i;(f \wedge_i \mathit{empty}))^r$
using *1 REqvRule* **by** *blast*
have *3*: $\vdash (\mathit{true}_i;(f \wedge_i \mathit{empty}))^r \equiv_i (f \wedge_i \mathit{empty})^r;\mathit{true}_i$
using *RTrue* **by** *auto*
have *4*: $\vdash (f \wedge_i \mathit{empty})^r;\mathit{true}_i \equiv_i (f^r \wedge_i \mathit{empty});\mathit{true}_i$
using *LeftChopEqvChop RAnd REmptyEqvEmpty prop03 prop05* **by** *blast*
have *5*: $\vdash (f^r \wedge_i \mathit{empty});\mathit{true}_i \equiv_i \mathit{init}(f^r)$
by *auto*
from *1 2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *RRFinEqvInit*:
$\vdash (\mathit{fin}\ (f^r))^r \equiv_i \mathit{init}\ (f)$
proof −
have *1*: $\vdash (\mathit{fin}\ (f^r))^r \equiv_i \mathit{init}\ ((f^r)^r)$ **using** *RFinEqvInit* **by** *blast*
have *2*: $\vdash \mathit{init}\ ((f^r)^r) \equiv_i \mathit{init}\ f$ **using** *EqvReverseReverse* **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *NextDiamondEqvDiamondNext*:
$\vdash \bigcirc(\diamond f) \equiv_i \diamond(\bigcirc f)$
proof −
have *1*: $\vdash \mathit{true}_i;\mathit{skip} \equiv_i \mathit{skip};\mathit{true}_i$ **by** (*rule TrueChopSkipEqvSkipChopTrue*)
hence *2*: $\vdash (\mathit{true}_i;\mathit{skip});f \equiv_i (\mathit{skip};\mathit{true}_i);f$ **using** *LeftChopEqvChop* **by** *blast*
have *3*: $\vdash (\mathit{true}_i;\mathit{skip});f \equiv_i \mathit{true}_i;(\mathit{skip};f)$ **using** *ChopAssoc itl-prop*(*30*) **by** *blast*
have *4*: $\vdash (\mathit{skip};\mathit{true}_i);f \equiv_i \mathit{skip};(\mathit{true}_i;f)$ **using** *ChopAssoc itl-prop*(*30*) **by** *blast*
from *2 3 4* **show** *?thesis* **by** (*simp add*: *next-d-def*)
qed

lemma *WeakNextBoxInduct*:
assumes $\vdash \mathit{wnext}\ (\Box f) \supset_i f$
shows $\vdash f$
proof −
have *1*: $\vdash \mathit{wnext}\ (\Box f) \supset_i f$ **using** *assms* **by** *blast*
hence *2*: $\vdash \neg_i f \supset_i \neg_i (\mathit{wnext}\ (\Box f))$ **using** *prop27* **by** *blast*
hence *3*: $\vdash \neg_i f \supset_i \bigcirc (\neg_i (\Box f))$ **by** *simp*
have *4*: $\vdash \neg_i (\Box f) \equiv_i (\diamond \neg_i f)$ **by** *auto*
hence *5*: $\vdash \bigcirc(\neg_i (\Box f)) \equiv_i \bigcirc (\diamond \neg_i f)$ **by** *auto*
have *6*: $\vdash \neg_i f \supset_i \bigcirc (\diamond \neg_i f)$ **using** *3 5* **by** *auto*
have *7*: $\vdash \bigcirc (\diamond \neg_i f) \equiv_i \diamond(\bigcirc \neg_i f)$ **using** *NextDiamondEqvDiamondNext* **by** *blast*

```
 have  8: ⊢ ¬ᵢ f ⊃ᵢ  ◇( ○ ¬ᵢ f)  using 6 7 by auto
 have  9: ⊢ ◇(¬ᵢ f) ⊃ᵢ  ◇(◇( ○ ¬ᵢ f)) using 8 DiamondImpDiamond by blast
 have 10: ⊢ ◇(◇( ○ ¬ᵢ f)) ≡ᵢ ◇( ○ ¬ᵢ f) using DiamondDiamondEqvDiamond by blast
 have 11: ⊢ ◇(¬ᵢ f) ⊃ᵢ ◇( ○ ¬ᵢ f) using 9 10 by auto
 have 12: ⊢ ◇(¬ᵢ f) ⊃ᵢ ○ (◇ ¬ᵢ f) using 7 11 by auto
 hence 13: ⊢ ¬ᵢ( ◇(¬ᵢ f)) using NextLoop by blast
 hence 14: ⊢ □f by (simp add: always-d-def)
 have 15: ⊢ □f ⊃ᵢ f using BoxElim by blast
 from 14 15 show ?thesis using MP by blast
qed

end
```

```
theory First
 imports
  Theorems
begin
```

6 The First Occurrence Operator in ITL

Runtime verification (RV) has gained significant interest in recent years. The behaviour of a program can be verified in real time by analysing its evolving trace. This approach has two significant benefits over static verification techniques such as model checking. Firstly, it is only necessary to verify actual execution paths rather than all possible paths. Secondly, it is possible to react at runtime should the program diverge from its specified behaviour. RV does not replace traditional verification techniques but it does provide an extra layer of security.

Linear Temporal Logic (LTL) is a popular formalism for writing specifications from which RV monitors can be derived automatically. By contrast, Interval Temporal Logic (ITL) has not been as widely represented in this field despite being more expressive and compositional. The principal issue is efficiency. ITL uses non-deterministic operators to construct sequential and iterative specifications (chop and chop-star, respectively) and these introduce combinatorial complexity. Approaches to mitigate this include using a deterministic subset of ITL or adapting the semantics to include a deterministic chop operator. This thesis proposes an alternative approach, wholly within existing ITL, and based upon a new, derived operator called "first occurrence".

A theory of first occurrence is developed and used to derive an algebra of RV monitors.

6.1 Definitions

6.1.1 Definitions Strict Initial and Final

```
definition bs-d ((bs -) [88] 87)
where
 bs f ≡ (empty ∨ᵢ ((bi f) ; skip))

definition bt-d ((bt -) [88] 87)
where
 bt f ≡ (empty ∨ᵢ (skip;(□ f)))
```

definition *ds-d* ((*ds* -) [*88*] *87*)
where
ds f $\equiv \neg_i$ (*bs* ($\neg_i$ *f*))

definition *dt-d* ((*dt* -) [*88*] *87*)
where
dt f $\equiv \neg_i$ (*bt* ($\neg_i$ *f*))

6.1.2 Definition First and Last Operators

definition *first-d* (($\triangleright$ -) [*88*] *87*)
where
$\triangleright$ *f* $\equiv$ (*f* $\wedge_i$ (*bs* ($\neg_i$ *f*)))

definition *last-d* (($\triangleleft$ -) [*88*] *87*)
where
$\triangleleft$ *f* $\equiv$ (*f* $\wedge_i$ (*bt* ($\neg_i$ *f*)))

6.2 First and Time Reversal

lemma *BsEqvRule*:
assumes $\vdash$ *f* $\equiv_i$ *g*
shows $\vdash$ *bs f* $\equiv_i$ *bs g*
proof –
have *1*: $\vdash$ *f* $\equiv_i$ *g* **using** *assms* **by** *auto*
hence *2*: $\vdash$ *bi*(*f*) $\equiv_i$ *bi*(*g*) **by** *simp*
hence *3*: $\vdash$ *bi*(*f*);*skip* $\equiv_i$ *bi*(*g*);*skip* **by** *auto*
hence *4*: $\vdash$ *empty* $\vee_i$ *bi*(*f*);*skip* $\equiv_i$ *empty* $\vee_i$ *bi*(*g*);*skip* **by** *auto*
hence *5*: $\vdash$ *bs*(*f*) $\equiv_i$ *bs*(*g*) **by** (*simp add*: *bs-d-def*)
from *1 2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *BtEqvRule*:
assumes $\vdash$ *f* $\equiv_i$ *g*
shows $\vdash$ *bt f* $\equiv_i$ *bt g*
proof –
have *1*: $\vdash$ *f* $\equiv_i$ *g* **using** *assms* **by** *auto*
hence *2*: $\vdash$ $\Box$(*f*) $\equiv_i$ $\Box$(*g*) **by** *simp*
hence *3*: $\vdash$ *skip*;$\Box$(*f*) $\equiv_i$ *skip*;$\Box$(*g*) **using** *RightChopEqvChop* **by** *blast*
hence *4*: $\vdash$ *empty* $\vee_i$ *skip*;$\Box$(*f*) $\equiv_i$ *empty* $\vee_i$ *skip*;$\Box$(*g*) **by** *auto*
hence *5*: $\vdash$ *bt*(*f*) $\equiv_i$ *bt*(*g*) **by** (*simp add*: *bt-d-def*)
from *1 2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *FstEqvRule*:
assumes $\vdash$ *f* $\equiv_i$ *g*
shows $\vdash$ $\triangleright$*f* $\equiv_i$ $\triangleright$*g*
proof –
have *1*: $\vdash$ *f* $\equiv_i$ *g* **using** *assms* **by** *auto*

hence *2*: $\vdash \neg_i\ f \equiv_i \neg_i\ g$ **by** *auto*
hence *3*: $\vdash bs(\neg_i\ f) \equiv_i bs(\neg_i\ g)$ **by** (*simp add*: *bs-d-def*)
hence *4*: $\vdash f \wedge_i bs(\neg_i\ f) \equiv_i g \wedge_i bs(\neg_i\ g)$ **using** *1* **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*:*first-d-def*)
qed

lemma *LstEqvRule*:
assumes $\vdash f \equiv_i g$
shows $\vdash \lhd f \equiv_i \lhd g$
proof –
have *1*: $\vdash f \equiv_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash \neg_i\ f \equiv_i \neg_i\ g$ **by** *auto*
hence *3*: $\vdash bt(\neg_i\ f) \equiv_i bt(\neg_i\ g)$ **by** (*simp add*: *bt-d-def*)
hence *4*: $\vdash f \wedge_i bt(\neg_i\ f) \equiv_i g \wedge_i bt(\neg_i\ g)$ **using** *1* **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*:*last-d-def*)
qed

lemma *RBsEqvBt*:
$\vdash (bs\ f)^r \equiv_i (bt\ (f^r))$
proof –
have *1*: $\vdash (bs\ f)^r \equiv_i (empty \vee_i ((bi\ f)\ ;\ skip))^r$
by (*simp add*: *bs-d-def*)
have *2*: $\vdash (empty \vee_i ((bi\ f)\ ;\ skip))^r \equiv_i (empty^r \vee_i ((bi\ f)\ ;\ skip)^r)$
using *ROr* **by** *blast*
have *3*: $\vdash (empty^r \vee_i ((bi\ f)\ ;\ skip)^r) \equiv_i (empty \vee_i (skip^r;(bi\ f)^r))$
using *REmptyEqvEmpty* **by** *auto*
have *4*: $\vdash (empty \vee_i (skip^r;(bi\ f)^r)) \equiv_i (empty \vee_i (skip;\Box\ (f^r)))$
by (*metis* (*no-types*, *lifting*) *NextEqvNext RBiEqvBox REmptyEqvEmpty next-d-def or-d-def prop01 prop21 prop39 rev-d*.*simps*(*4*))
have *5*: $\vdash (empty \vee_i (skip;\Box\ (f^r))) \equiv_i (bt\ (f^r))$
by (*simp add*: *bt-d-def*)
from *1 2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *RRBsEqvBt*:
$\vdash (bs\ (f^r))^r \equiv_i (bt\ (f))$
proof –
have *1*: $\vdash (bs\ (f^r))^r \equiv_i bt\ ((f^r)^r)$ **using** *RBsEqvBt* **by** *blast*
have *2*: $\vdash bt\ ((f^r)^r) \equiv_i bt\ f$ **using** *EqvReverseReverse* **using** *BtEqvRule* **by** *blast*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RBtEqvBs*:
$\vdash (bt\ f)^r \equiv_i (bs\ (f^r))$
proof –
have *1*: $\vdash (bt\ f)^r \equiv_i (empty \vee_i (skip;\Box\ f))^r$
by (*simp add*: *bt-d-def*)
have *2*: $\vdash (empty \vee_i (skip;\Box\ f))^r \equiv_i (empty^r \vee_i (skip;\Box\ f)^r)$
using *ROr* **by** *blast*
have *3*: $\vdash (empty^r \vee_i (skip;\Box\ f)^r) \equiv_i (empty \vee_i (\Box\ f)^r;skip^r)$

using *REmptyEqvEmpty* **by** *auto*
have *4*: $\vdash (empty \lor_i (\Box\ f)^r;skip^r) \equiv_i (empty \lor_i (bi\ (f^r));skip)$
by (*metis* (*no-types*, *lifting*) *LeftChopEqvChop RBoxEqvBi REmptyEqvEmpty or-d-def prop01 prop21 prop39 rev-d.simps*(*4*))
have *5*: $\vdash (empty \lor_i (bi\ (f^r));skip) \equiv_i (bs\ (f^r))$
by (*simp add*: *bs-d-def*)
from *1 2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *RRBtEqvBs*:
$\vdash (bt\ (f^r))^r \equiv_i (bs\ (f))$
proof –
have *1*: $\vdash (bt\ (f^r))^r \equiv_i bs\ ((f^r)^r)$ **using** *RBtEqvBs* **by** *blast*
have *2*: $\vdash bs\ ((f^r)^r) \equiv_i bs\ f$ **using** *EqvReverseReverse* **using** *BsEqvRule* **by** *blast*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RFirstEqvLast*:
$\vdash (\rhd\ f)^r \equiv_i (\lhd\ (f^r))$
proof –
have *1*: $\vdash (\rhd\ f)^r \equiv_i (f \land_i bs(\neg_i\ f))^r$ **by** (*simp add*: *first-d-def*)
have *2*: $\vdash (f \land_i bs(\neg_i\ f))^r \equiv_i (f^r \land_i (bs\ (\neg_i\ f))^r)$ **using** *RAnd* **by** *blast*
have *3*: $\vdash (f^r \land_i (bs\ (\neg_i\ f))^r) \equiv_i (f^r \land_i bt\ ((\neg_i\ f)^r))$ **using** *RBsEqvBt prop05* **by** *blast*
have *4*: $\vdash (f^r \land_i bt\ ((\neg_i\ f)^r)) \equiv_i (f^r \land_i bt\ (\neg_i(f^r)))$ **by** (*simp add*: *not-d-def*)
have *5*: $\vdash (f^r \land_i bt\ (\neg_i(f^r))) \equiv_i (\lhd\ (f^r))$ **by** (*simp add*: *last-d-def*)
from *1 2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *RRFirstEqvLast*:
$\vdash (\rhd\ (f^r))^r \equiv_i (\lhd\ (f))$
proof –
have *1*: $\vdash (\rhd\ (f^r))^r \equiv_i \lhd\ ((f^r)^r)$ **using** *RFirstEqvLast* **by** *blast*
have *2*: $\vdash \lhd\ ((f^r)^r) \equiv_i \lhd\ f$ **using** *EqvReverseReverse* **using** *LstEqvRule* **by** *blast*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *RLastEqvFirst*:
$\vdash (\lhd\ f)^r \equiv_i (\rhd\ (f^r))$
proof –
have *1*: $\vdash (\lhd\ f)^r \equiv_i (f \land_i bt(\neg_i\ f))^r$ **by** (*simp add*: *last-d-def*)
have *2*: $\vdash (f \land_i bt(\neg_i\ f))^r \equiv_i (f^r \land_i (bt\ (\neg_i f))^r)$ **using** *RAnd* **by** *blast*
have *3*: $\vdash (f^r \land_i (bt\ (\neg_i f))^r) \equiv_i (f^r \land_i bs\ (\ (\neg_i f)^r))$ **using** *RBtEqvBs prop05* **by** *blast*
have *4*: $\vdash (f^r \land_i bs\ (\ (\neg_i f)^r)) \equiv_i (f^r \land_i bs(\neg_i(f^r)))$ **by** (*simp add*: *not-d-def*)
have *5*: $\vdash (f^r \land_i bs(\neg_i(f^r))) \equiv_i (\rhd\ (f^r))$ **by** (*simp add*: *first-d-def*)
from *1 2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *RRLastEqvFirst*:
$\vdash (\lhd\ (f^r))^r \equiv_i (\rhd\ (f))$
proof –

have *1*: $\vdash (\lhd (f^r))^r \equiv_i \rhd ((f^r)^r)$ **using** *RLastEqvFirst* **by** *blast*
have *2*: $\vdash \rhd ((f^r)^r) \equiv_i \rhd f$ **using** *EqvReverseReverse* **using** *FstEqvRule* **by** *blast*
from *1 2* **show** *?thesis* **by** *auto*
qed

6.3 Semantic Theorems

6.3.1 Semantics First and Last Operators

lemma *FstAndBisem*:
$(\mathit{intlen}\ \sigma > 0 \wedge (\sigma \models f) \wedge (\ \sigma \models\ \ \mathit{bi}\ \neg_i f;\mathit{skip})) =$
$(\mathit{intlen}\ \sigma > 0 \wedge (\sigma \models f) \wedge (\forall\, \mathit{ia}{<}\mathit{intlen}\ (\sigma).\ (\mathit{prefix}\ \mathit{ia}\ \ \sigma \models \neg_i f))\ )$
apply *simp-all*
apply (*simp add*: *interval-prefix-length interval-suffix-length*)
proof −
have *1*: $(0 < \mathit{intlen}\ \sigma \wedge (\sigma \models f) \wedge$
$(\exists\, i.\ (i \leq \mathit{intlen}\ \sigma \longrightarrow (\forall\, \mathit{ia}{\leq}i.\ \neg\ (\mathit{prefix}\ \mathit{ia}\ (\mathit{prefix}\ i\ \sigma) \models f)) \wedge$
$\mathit{intlen}\ \sigma - i = \mathit{Suc}\ 0) \wedge i \leq \mathit{intlen}\ \sigma)$
$) =$
$(0 < \mathit{intlen}\ \sigma \wedge (\sigma \models f) \wedge$
$(\exists\, i.\ (i \leq \mathit{intlen}\ \sigma \longrightarrow (\forall\, \mathit{ia}{\leq}i.\ \neg\ (\mathit{prefix}\ \mathit{ia}\ (\mathit{prefix}\ i\ \sigma) \models f)) \wedge$
$i = \mathit{intlen}\ \sigma - \mathit{Suc}\ 0) \wedge i \leq \mathit{intlen}\ \sigma)$
$)$
by *auto*
also have ... =
$(0 < \mathit{intlen}\ \sigma \wedge (\sigma \models f) \wedge$
$(\forall\, \mathit{ia}{\leq}(\mathit{intlen}\ \sigma - \mathit{Suc}\ 0).\ \neg\ (\mathit{prefix}\ \mathit{ia}\ (\mathit{prefix}\ (\mathit{intlen}\ \sigma - \mathit{Suc}\ 0)\ \sigma) \models f)\ )$
$)$
using *diff-le-self* **by** *blast*
also have ... =
$(\mathit{intlen}\ \sigma > 0 \wedge (\sigma \models f) \wedge$
$(\forall\, \mathit{ia}{<}\mathit{intlen}\ (\ \sigma).\ \neg\ (\mathit{prefix}\ \mathit{ia}\ (\mathit{prefix}\ (\mathit{intlen}\ \sigma - \mathit{Suc}\ 0)\ \sigma) \models f))$
$)$ **by** (*metis Suc-pred less-Suc-eq-le*)
also have ... =
$(\mathit{intlen}\ \sigma > 0 \wedge (\sigma \models f) \wedge$
$(\forall\, \mathit{ia}{<}\mathit{intlen}\ (\ \sigma).\ (\mathit{prefix}\ \mathit{ia}\ (\mathit{prefix}\ (\mathit{intlen}\ \sigma - \mathit{Suc}\ 0)\ \sigma) \models \neg_i f))$
$)$
using *not-defs* **by** *blast*
also have ... =
$(\mathit{intlen}\ \sigma > 0 \wedge (\sigma \models f) \wedge (\forall\, \mathit{ia}{<}\mathit{intlen}\ (\ \sigma).\ \neg\ (\mathit{prefix}\ \mathit{ia}\ \ \sigma \models f)))$
by (*simp add*: *interval-pref-pref-help*)
finally show $(0 < \mathit{intlen}\ \sigma \wedge (\sigma \models f) \wedge$
$(\exists\, i.\ (i \leq \mathit{intlen}\ \sigma \longrightarrow (\forall\, \mathit{ia}{\leq}i.\ \neg\ (\mathit{prefix}\ \mathit{ia}\ (\mathit{prefix}\ i\ \sigma) \models f)) \wedge$
$\mathit{intlen}\ \sigma - i = \mathit{Suc}\ 0) \wedge i \leq \mathit{intlen}\ \sigma)$
$) =$
$(0 < \mathit{intlen}\ \sigma \wedge (\sigma \models f) \wedge (\forall\, \mathit{ia}{<}\mathit{intlen}\ \sigma.\ \neg\ (\mathit{prefix}\ \mathit{ia}\ \sigma \models f)))$ **.**
qed

lemma *Fstsem-0*:
$(\sigma \models \rhd f) =$
$($

```
  ( σ ⊨ f ∧_i empty) ∨ (intlen σ >0  ∧ (σ ⊨ f) ∧ ( σ ⊨  bi ¬_i f;skip))
 )
apply (simp add: first-d-def bs-d-def) using neq0-conv by blast

lemma Emptysem:
 (σ ⊨ f ∧_i empty) = ((σ ⊨ f) ∧ intlen σ = 0)
by simp

lemma Fstsem:
 (σ ⊨ ▷f) =
  (
   ( (σ ⊨ f) ∧ intlen σ = 0) ∨
   ( intlen σ>0 ∧ (σ ⊨ f) ∧ (∀ ia<intlen (σ). (prefix ia  σ ⊨ ¬_i f)))
  )
using Fstsem-0 Emptysem FstAndBisem by blast

lemma Lstsem:
 (σ ⊨ ◁ f) =
  ( ( (σ ⊨ f) ∧ intlen σ = 0) ∨
    ( intlen σ >0 ∧ (σ ⊨ f) ∧ (∀ ia < intlen σ. (suffix ((intlen σ) −ia)  σ ⊨ ¬_i f)) )
  )

proof −
 have (σ ⊨ ◁ f) = (σ ⊨ (▷ (f^r))^r)
     using RRFirstEqvLast iff-defs itl-valid by blast
 also have  ... = (intrev σ ⊨ ▷ (f^r))
     by (metis TimeReverseSem interval-rev-rev-ident)
 also have  ... =
  (
   ( (intrev σ ⊨ f^r) ∧ intlen (intrev σ) = 0) ∨
   ( intlen (intrev σ)>0 ∧ (intrev σ ⊨ f^r) ∧
     (∀ ia<intlen (intrev σ). (prefix ia  (intrev σ) ⊨ ¬_i(f^r))))
  )
     using Fstsem by blast
 also have  ... =
  (
   ( ( σ ⊨ f) ∧ intlen (σ) = 0) ∨
   ( intlen (σ)>0 ∧ ( σ ⊨ f) ∧
     (∀ ia<intlen (intrev σ). (prefix ia  (intrev σ) ⊨ (¬_i(f))^r)))
  )
     by (metis TimeReverseSem interval-intrev-intlen not-d-def rev-d.simps(1) rev-d.simps(3))
 also have  ... =
  (
   ( ( σ ⊨ f) ∧ intlen (σ) = 0) ∨
   ( intlen (σ)>0 ∧ ( σ ⊨ f) ∧
     (∀ ia<intlen (intrev σ). (intrev (prefix ia  (intrev σ)) ⊨ (¬_i(f)))))
  )
     using TimeReverseSem by (metis interval-rev-rev-ident)
 also have  ... =
  (
  ( ( σ ⊨ f ) ∧ intlen (σ) = 0) ∨
  ( intlen (σ)>0 ∧ ( σ ⊨ f ) ∧
    (∀ ia<intlen (σ). ( (suffix ((intlen σ) − ia)  (σ)) ⊨ (¬_i(f )))))
 )
     by (simp add: interval-intrev-prefix)
 finally show
 (σ ⊨ ◁ f ) =
 ( ( (σ ⊨ f ) ∧ intlen σ = 0) ∨
   ( intlen σ >0 ∧ (σ ⊨ f ) ∧
     (∀ ia < intlen σ. (suffix ((intlen σ) −ia)  σ ⊨ ¬_i f )) )
 )  .
qed
```

6.3.2 Various Semantic Lemmas

```
lemma DiLensem:
 (σ ⊨ di (f ∧_i len(i))) =
   ( (prefix i σ ⊨ f ) ∧ i≤intlen σ)
apply simp-all
using interval-prefix-length-good by auto

lemma PrefixFstsem:
 ( (prefix i σ ⊨ ▷f ) ∧ i≤intlen σ) =
  ( i≤intlen σ ∧
   (
    ( (prefix i σ ⊨ f ) ∧ i = 0) ∨
    ( i>0 ∧ (prefix i σ ⊨ f ) ∧ (∀ ia<i. (prefix ia σ ⊨ ¬_i f )))
   )
  )
proof −
 have 1: ( ((prefix i σ) ⊨ ▷f )) =
  (
   ( ((prefix i σ) ⊨ f ) ∧ intlen (prefix i σ) = 0) ∨
   ( intlen (prefix i σ)>0 ∧ ((prefix i σ) ⊨ f ) ∧
     (∀ ia<intlen (prefix i σ). (prefix ia  (prefix i σ) ⊨ ¬_i f )) )
  )
      using Fstsem by blast
 hence 2: ( ((prefix i σ) ⊨ ▷f ) ∧ i≤intlen σ) =
  (  i≤intlen σ ∧ (
   ( ((prefix i σ) ⊨ f ) ∧ intlen (prefix i σ) = 0) ∨
   ( intlen (prefix i σ)>0 ∧ ((prefix i σ) ⊨ f ) ∧
     (∀ ia<intlen (prefix i σ). (prefix ia  (prefix i σ) ⊨ ¬_i f )) )
    )
  )
      by auto
 hence 3: ( ((prefix i σ) ⊨ ▷f ) ∧ i≤intlen σ) =
  (  i≤intlen σ ∧ (
   ( ((prefix i σ) ⊨ f ) ∧ i = 0) ∨
   ( i>0 ∧ ((prefix i σ) ⊨ f ) ∧ (∀ ia<i. (prefix ia  (prefix i σ) ⊨ ¬_i f )))
    )
  )
     by auto
 hence 4: ( ((prefix i σ) ⊨ ▷f ) ∧ i≤intlen σ) =
  (  i≤intlen σ ∧ (
   ( ((prefix i σ) ⊨ f ) ∧ i = 0) ∨
   ( i>0 ∧ ((prefix i σ) ⊨ f ) ∧ (∀ ia<i. (prefix ia σ ⊨ ¬_i f )))
    )
  )
     using interval-pref-pref-3 using less-imp-add-positive by fastforce
 from 4 show ?thesis by auto
qed

lemma PrefixFstAndsem:
 ( (prefix i σ ⊨ ▷f ∧_i g) ∧ i≤intlen σ) =
  ( i≤intlen σ ∧
   (
    ( (prefix i σ ⊨ f ∧_i g ) ∧ i = 0) ∨
    ( i>0 ∧ (prefix i σ ⊨ f ∧_i g) ∧ (∀ ia<i. (prefix ia σ ⊨ ¬_i f )))
   )
 )
using PrefixFstsem by (metis and-defs)

lemma DiLenFstsem:
 (σ ⊨ di (▷f ∧_i len(i))) =
   ( i≤intlen σ ∧
   (
    ( (prefix i σ ⊨ f ) ∧ i = 0) ∨
    ( i>0 ∧ (prefix i σ ⊨ f ) ∧ (∀ ia<i. (prefix ia σ ⊨ ¬_i f )))
   )
 )
using DiLensem PrefixFstsem by blast

lemma DiLenFstAndsem:
 (σ ⊨ di (▷f ∧_i g ∧_i len(i))) =
   ( i≤intlen σ ∧
   (
    ( (prefix i σ ⊨ f ∧_i g) ∧ i = 0) ∨
    ( i>0 ∧ (prefix i σ ⊨ f ∧_i g) ∧ (∀ ia<i. (prefix ia σ ⊨ ¬_i f )))
   )
 )
proof –
 have 1: (σ ⊨ di ((▷f ∧_i g) ∧_i len(i))) =
         ( (prefix i σ ⊨ (▷f ∧_i g)) ∧ i≤intlen σ)
     using DiLensem by blast
 have 2: ((prefix i σ ⊨ (▷f ∧_i g)) ∧ i≤intlen σ)=
         ( i≤intlen σ ∧
          (
           ( (prefix i σ ⊨ f ∧_i g ) ∧ i = 0) ∨
           ( i>0 ∧ (prefix i σ ⊨ f∧_i g) ∧ (∀ ia<i. (prefix ia σ ⊨ ¬_i f )))
          )
```

```
      )
    using PrefixFstAndsem by blast
 from 1 2 show ?thesis by auto
qed

lemma FstLenSamesem:
 ( ( i≤intlen σ ∧
   (
    ( (prefix i σ ⊨ f) ∧ i = 0) ∨
    ( i>0 ∧ (prefix i σ ⊨ f) ∧ (∀ ia<i. (prefix ia σ ⊨ ¬_i f)))
   )
 ) ∧
  ( j≤intlen σ ∧
   (
    ( (prefix j σ ⊨ f) ∧ j = 0) ∨
    ( j>0 ∧ (prefix j σ ⊨ f) ∧ (∀ ia<j. (prefix ia σ ⊨ ¬_i f)))
   )
 )
) ⟶ (i=j)

using linorder-neqE-nat by (meson not-defs)
```

6.4 Theorems

6.4.1 Fixed length intervals

```
lemma LenZeroEqvEmpty:
 ⊢ len(0) ≡_i empty
by simp

lemma LenOneEqvSkip:
 ⊢ len(1) ≡_i skip
by (metis ChopEmpty One-nat-def len_d.simps(1) len_d.simps(2))

lemma LenNPlusOneA:
 ⊢  len(n+1) ≡_i skip;len(n)
by simp

lemma LenEqvLenChopLen:
 ⊢ len(i+j) ≡_i len(i);len(j)
proof
 (induct i)
 case 0
 then show ?case using add.left-neutral by auto
 next
 case (Suc i)
 then show ?case
 by (metis ChopAssoc NextEqvNext add-Suc len_d.simps(2) next-d-def prop03)
qed

lemma LenNPlusOneB:
```

```
⊢ len(n+1) ≡_i len(n);skip
proof −
 have  1: ⊢ len(n+1) ≡_i len(n);len(1) by (rule LenEqvLenChopLen)
 have  2: ⊢ len(1) ≡_i skip by (rule LenOneEqvSkip)
 hence 3: ⊢ len(n);len(1) ≡_i len(n);skip using RightChopEqvChop by blast
 from 1 3 show ?thesis using prop03 by blast
qed

lemma LenCommute:
⊢ (skip;(len n)) ≡_i (len n);skip
proof
  (induct n)
  case 0
  then show ?case using EmptyChop ChopEmpty
  using LenNPlusOneA LenNPlusOneB prop21 by blast
  next
  case (Suc n)
  then show ?case using ChopAssoc
  using LenNPlusOneA LenNPlusOneB prop21 by blast
qed

lemma SkipTrueEqvTrueSkip:
⊢ skip;true_i ≡_i true_i;skip
using TrueChopSkipEqvSkipChopTrue itl-prop(30) by blast

lemma PowerCommute:
⊢ (f;(power f n)) ≡_i ((power f n);f)
proof
  (induct n)
  case 0
  then show ?case using EmptyChop ChopEmpty pow-0 by fastforce
  next
  case (Suc n)
  then show ?case using ChopAssoc pow-Suc
  by (metis RightChopEqvChop prop03)
qed

lemma PowerRev:
⊢ (power skip n)^r ≡_i (power skip n)
proof
 (induct n)
 case 0
 then show ?case using REmptyEqvEmpty by auto
 next
 case (Suc n)
 then show ?case using PowerCommute rev-d.simps pow-Suc
 proof −
 have ∀p. (⊢ (power (skip::'a pitl) n) ^r ; p ≡_i power skip n ; p )
 using LeftChopEqvChop Suc.hyps by blast
 then show ?thesis
```

by (*metis* (*no-types*) *PowerCommute itl-prop*(*30*) *pow-Suc prop03 rev-d.simps*(*4*) *rev-d.simps*(*5*))
qed
qed

lemma *RLenEqvLen*:
$\vdash (len\ k)^r \equiv_i (len\ k)$
proof
(*induct k*)
case *0*
then show *?case* **using** *REmptyEqvEmpty* **by** *auto*
next
case (*Suc k*)
then show *?case* **using** *LenCommute*
by (*metis NextEqvNext REqvRule len_d.simps*(*2*) *next-d-def prop03 rev-d.simps*(*4*) *rev-d.simps*(*5*))
qed

lemma *PowerSkipEqvLen*:
$\vdash (power\ skip\ n) \equiv_i (len\ n)$
proof
(*induct n*)
case *0*
then show *?case* **by** *auto*
next
case (*Suc n*)
then show *?case* **by** *simp*
qed

lemma *ExistsLen*:
$(\forall \sigma.\ \exists k.\ (\sigma \models len(k)))$
using *len-defs* **by** *simp*

lemma *AndExistsLen*:
$(\forall \sigma.\ (\sigma \models f) = (\ (\sigma \models f) \wedge (\exists k.\ (\sigma \models len(k)))))$
using *ExistsLen* **by** *simp*

lemma *AndExistsLenChop*:
$(\forall \sigma.\ (\sigma \models f;g) = (\exists k.\ \ (\sigma \models (f \wedge_i len(k));g\ )))$
using *AndExistsLen* **by** *auto*

lemma *AndExistsLenChopR*:
$(\forall \sigma.\ (\sigma \models f;g) = (\exists k.\ \ (\sigma \models f;(g \wedge_i len(k)))))$
using *AndExistsLen* **by** *auto*

lemma *LFixedAndDistr*:
$\vdash (f0 \wedge_i len(k));g0 \wedge_i (f1 \wedge_i len(k));g1 \equiv_i (f0 \wedge_i f1 \wedge_i len(k));(g0 \wedge_i g1)$
apply *simp-all*
by (*metis interval-prefix-length-good*)

lemma *RFixedAndDistr*:
$\vdash f0;(g0 \wedge_i len(k)) \wedge_i f1;(g1 \wedge_i len(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i g1 \wedge_i len(k))$

```
apply simp-all
apply (simp add: interval-prefix-length interval-suffix-length)
by (metis diff-diff-cancel)

lemma LFixedAndDistrA:
⊢ (f0 ∧_i len(k));g0 ∧_i (f1 ∧_i len(k));g0 ≡_i (f0 ∧_i f1 ∧_i len(k));g0
proof −
 have 1: ⊢ (f0 ∧_i len(k));g0 ∧_i (f1 ∧_i len(k));g0 ≡_i (f0 ∧_i f1 ∧_i len(k));(g0 ∧_i g0)
     by (rule LFixedAndDistr)
 have 2: ⊢ (f0 ∧_i f1 ∧_i len(k));(g0 ∧_i g0) ≡_i (f0 ∧_i f1 ∧_i len(k));g0
     by auto
 from 1 2 show ?thesis using prop03 by blast
qed

lemma LFixedAndDistrB:
⊢ (f0 ∧_i len(k));g0 ∧_i (f0 ∧_i len(k));g1 ≡_i (f0 ∧_i len(k));(g0 ∧_i g1)
proof −
 have 1: ⊢ (f0 ∧_i len(k));g0 ∧_i (f0 ∧_i len(k));g1 ≡_i (f0 ∧_i f0 ∧_i len(k));(g0 ∧_i g1)
     by (rule LFixedAndDistr)
 have 2: ⊢ (f0 ∧_i f0 ∧_i len(k));(g0 ∧_i g1) ≡_i (f0 ∧_i len(k));(g0 ∧_i g1)
     by auto
 from 1 2 show ?thesis using prop03 by blast
qed

lemma LFixedAndDistrB1:
⊢ len(k);f ∧_i len(k);g ≡_i len(k);(f ∧_i g)
proof −
 have 1: ⊢ len(k);f ≡_i (true_i ∧_i len(k));f
     by auto
 have 2: ⊢ len(k);g ≡_i (true_i ∧_i len(k));g
     by auto
 have 3: ⊢ len(k);f ∧_i len(k);g ≡_i (true_i ∧_i len(k));f ∧_i (true_i ∧_i len(k));g
     using 1 2 by auto
 have 4: ⊢ (true_i ∧_i len(k));f ∧_i (true_i ∧_i len(k));g ≡_i (true_i ∧_i len(k));(f ∧_i g)
     using LFixedAndDistrB by blast
 have 5: ⊢ (true_i ∧_i len(k));(f ∧_i g) ≡_i (len(k));(f ∧_i g)
     by auto
 from 1 2 3 4 5 show ?thesis by auto
qed


lemma RFixedAndDistrA:
⊢ f0;(g0 ∧_i len(k)) ∧_i f0;(g1 ∧_i len(k)) ≡_i f0;(g0 ∧_i g1 ∧_i len(k))
proof −
 have 1: ⊢ f0;(g0 ∧_i len(k)) ∧_i f0;(g1 ∧_i len(k)) ≡_i (f0 ∧_i f0);(g0 ∧_i g1 ∧_i len(k))
     by (rule RFixedAndDistr)
 have 2: ⊢ (f0 ∧_i f0);(g0 ∧_i g1 ∧_i len(k)) ≡_i f0;(g0 ∧_i g1 ∧_i len(k))
     by auto
from 1 2 show ?thesis using prop03 by blast
qed
```

lemma *RFixedAndDistrB*:
$\vdash f0;(g0 \wedge_i len(k)) \wedge_i f1;(g0 \wedge_i len(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i len(k))$
proof –
have *1*: $\vdash f0;(g0 \wedge_i len(k)) \wedge_i f1;(g0 \wedge_i len(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i g0 \wedge_i len(k))$
by (*rule RFixedAndDistr*)
have *2*: $\vdash (f0 \wedge_i f1);(g0 \wedge_i g0 \wedge_i len(k)) \equiv_i (f0 \wedge_i f1);(g0 \wedge_i len(k))$
by *auto*
from *1 2* **show** *?thesis* **using** *prop03* **by** *blast*
qed

lemma *ChopSkipAndChopSkip*:
$\vdash f0;skip \wedge_i f1;skip \equiv_i (f0 \wedge_i f1);skip$
proof –
have *1*: $\vdash f0;(true_i \wedge_i len(1)) \wedge_i f1;(true_i \wedge_i len(1)) \equiv_i (f0 \wedge_i f1);(true_i \wedge_i len(1))$
by (*rule RFixedAndDistrB*)
have *2*: $\vdash (true_i \wedge_i len(1)) \equiv_i skip$
using *LenOneEqvSkip itl-prop*(*17*) *prop03* **by** *blast*
hence *3*: $\vdash f0;(true_i \wedge_i len(1)) \equiv_i f0;skip$
using *RightChopEqvChop* **by** *blast*
have *4*: $\vdash f1;(true_i \wedge_i len(1)) \equiv_i f1;skip$
using *2 RightChopEqvChop* **by** *blast*
have *5*: $\vdash f0;(true_i \wedge_i len(1)) \wedge_i f1;(true_i \wedge_i len(1)) \equiv_i f0;skip \wedge_i f1;skip$
using *3 4* **by** *auto*
have *6*: $\vdash (f0 \wedge_i f1);(true_i \wedge_i len(1)) \equiv_i (f0 \wedge_i f1);skip$
using *2 RightChopEqvChop* **by** *blast*
from *1 5 6* **show** *?thesis* **by** *auto*
qed

lemma *BiAndChopSkipEqv*:
$\vdash (bi\ (f \wedge_i g));skip \equiv_i (bi\ f);skip \wedge_i (bi\ g);skip$
proof –
have *1*: $\vdash bi\ (f \wedge_i g) \equiv_i (bi\ f) \wedge_i (bi\ g)$
by *auto*
hence *2*: $\vdash (bi\ (f \wedge_i g));skip \equiv_i (bi\ f \wedge_i bi\ g);skip$
by (*rule LeftChopEqvChop*)
have *3*: $\vdash (bi\ f \wedge_i bi\ g);skip \equiv_i (bi\ f);skip \wedge_i (bi\ g);skip$
using *ChopSkipAndChopSkip itl-prop*(*30*) **by** *blast*
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *DiAndChopSkipEqv*:
$\vdash (di\ (f \wedge_i g));skip \supset_i (di\ f);skip \wedge_i (di\ g);skip$
proof –
have *1*: $\vdash di\ (f \wedge_i g) \supset_i (di\ f) \wedge_i (di\ g)$
by *auto*
hence *2*: $\vdash (di\ (f \wedge_i g));skip \supset_i (di\ f \wedge_i di\ g);skip$
by (*rule LeftChopImpChop*)
have *3*: $\vdash (di\ f \wedge_i di\ g);skip \equiv_i (di\ f);skip \wedge_i (di\ g);skip$
using *ChopSkipAndChopSkip itl-prop*(*30*) **by** *blast*

from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *ChopEmptyAndEmpty*:
$\vdash f;g \wedge_i \mathit{empty} \equiv_i f \wedge_i g \wedge_i \mathit{empty}$
apply *simp-all*
by (*metis interval-prefix-intlen interval-suffix-zero le-zero-eq*)

lemma *ChopSkipImpMore*:
$\vdash f;\mathit{skip} \supset_i \mathit{more}$
proof −
have *1*: $\vdash \neg_i(f;\mathit{skip} \wedge_i \mathit{empty})$ **by** *auto*
hence *2*: $\vdash \neg_i(f;\mathit{skip}) \vee_i \mathit{more}$ **by** *auto*
from *2* **show** *?thesis* **by** *auto*
qed

lemma *MoreEqvMoreChopTrue*:
$\vdash \mathit{more} \equiv_i \mathit{more};\mathit{true}_i$
proof −
have *1*: $\vdash \mathit{more} \equiv_i \mathit{skip};\mathit{true}_i$
using *MoreEqvSkipChopTrue* **by** *blast*
have *2*: $\vdash \mathit{true}_i \equiv_i \mathit{true}_i;\mathit{true}_i$
by *auto*
hence *3*: $\vdash \mathit{skip};\mathit{true}_i \equiv_i \mathit{skip};(\mathit{true}_i;\mathit{true}_i)$
using *RightChopEqvChop* **by** *blast*
have *4*: $\vdash \mathit{skip};(\mathit{true}_i;\mathit{true}_i) \equiv_i (\mathit{skip};\mathit{true}_i);\mathit{true}_i$
using *ChopAssoc* **by** *blast*
have *5*: $\vdash (\mathit{skip};\mathit{true}_i);\mathit{true}_i \equiv_i \mathit{more};\mathit{true}_i$
using *MoreEqvSkipChopTrue* **by** (*simp add*: *more-d-def next-d-def*)
from *1 3 4 5* **show** *?thesis* **using** *prop03* **by** *blast*
qed

lemma *NotNotChopSkip*:
$\vdash \neg_i(\neg_i\ f\ ;\mathit{skip}) \equiv_i \mathit{empty} \vee_i (f;\mathit{skip})$
by (*metis WprevEqvEmptyOrPrev prev-d-def wprev-d-def*)

lemma *NotChopFixed*:
$\vdash \neg_i(f;(g \wedge_i \mathit{len}(k))) \equiv_i \neg_i(\Diamond(g \wedge_i \mathit{len}(k))) \vee_i (\neg_i f;(g \wedge_i \mathit{len}(k)))$
apply *simp* **by** (*smt diff-diff-cancel interval-suffix-length-good*)

lemma *NotFixedChop*:
$\vdash \neg_i((g \wedge_i \mathit{len}(k));f) \equiv_i \neg_i(\mathit{di}(g \wedge_i \mathit{len}(k))) \vee_i ((g \wedge_i \mathit{len}(k));\neg_i f)$
apply *simp* **by** *auto*

lemma *NotChopNotSkip*:
$\vdash \neg_i(f;\mathit{skip}) \equiv_i \mathit{empty} \vee_i ((\neg_i\ f);\mathit{skip})$
proof −

have *1*: $\vdash \neg_i(\neg_i(\neg_i f);\mathit{skip}) \equiv_i \mathit{empty} \vee_i ((\neg_i f);\mathit{skip})$ **using** *NotNotChopSkip* **by** *blast*
have *2*: $\vdash \neg_i(\neg_i(\neg_i f);\mathit{skip}) \equiv_i \neg_i(f;\mathit{skip})$ **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

6.4.2 Additional ITL theorems

lemma *BiOrBiImpBiOr*:
$\vdash \mathit{bi}\ f \vee_i \mathit{bi}\ g \supset_i \mathit{bi}(f \vee_i g)$
proof –
have *1*: $\vdash f \supset_i f \vee_i g$ **by** *auto*
hence *2*: $\vdash \mathit{bi}\ f \supset_i \mathit{bi}(f \vee_i g)$ **by** (*rule BiImpBiRule*)
have *3*: $\vdash g \supset_i f \vee_i g$ **by** *auto*
hence *4*: $\vdash \mathit{bi}\ g \supset_i \mathit{bi}(f \vee_i g)$ **by** (*rule BiImpBiRule*)
from *2 4* **show** *?thesis* **by** *auto*
qed

lemma *MoreAndBiImpBiChopSkip*:
$\vdash \mathit{more} \wedge_i \mathit{bi}\ f \supset_i (\mathit{bi}\ f);\mathit{skip}$
proof –
have *1*: $\vdash (\mathit{bi}\ f);\mathit{skip} \equiv_i \neg_i(\mathit{di}\ \neg_i f);\mathit{skip}$ **by** *auto*
have *2*: $\vdash \neg_i(\neg_i(\mathit{di}\ \neg_i f);\mathit{skip}) \equiv_i \mathit{empty} \vee_i (\mathit{di}\ \neg_i f);\mathit{skip}$ **by** (*rule NotNotChopSkip*)
have *3*: $\vdash \mathit{empty} \supset_i \mathit{empty} \vee_i \mathit{di}\ \neg_i f$ **by** *auto*
have *4*: $\vdash (\mathit{di}\ \neg_i f);\mathit{skip} \supset_i \mathit{di}\ \neg_i f$ **using** *ChopImpDi DiEqvDiDi itl-prop*(*31*) *prop02* **by** *blast*
hence *5*: $\vdash (\mathit{di}\ \neg_i f);\mathit{skip} \supset_i \mathit{empty} \vee_i \mathit{di}\ \neg_i f$ **by** (*rule prop26*)
have *6*: $\vdash \neg_i(\neg_i(\mathit{di}\ \neg_i f);\mathit{skip}) \supset_i \mathit{empty} \vee_i \mathit{di}\ \neg_i f$ **using** *2 3 5* **by** *auto*
hence *7*: $\vdash \neg_i(\mathit{empty} \vee_i \mathit{di}\ \neg_i f) \supset_i \neg_i(\neg_i(\neg_i(\mathit{di}\ \neg_i f);\mathit{skip}))$ **by** (*rule prop27*)
have *8*: $\vdash \neg_i(\neg_i(\neg_i(\mathit{di}\ \neg_i f);\mathit{skip})) \equiv_i \neg_i(\mathit{di}\ \neg_i f);\mathit{skip}$ **by** *auto*
have *9*: $\vdash \neg_i(\mathit{empty} \vee_i \mathit{di}\ \neg_i f) \equiv_i \mathit{more} \wedge_i \neg_i(\mathit{di}\ \neg_i f)$ **by** *auto*
have *10*: $\vdash \mathit{more} \wedge_i \neg_i(\mathit{di}\ \neg_i f) \equiv_i \mathit{more} \wedge_i \mathit{bi}\ f$ **by** *auto*
from *1 6 7 8 9 10* **show** *?thesis* **by** *auto*
qed

lemma *DiChopImpDiB*:
$\vdash \mathit{di}(f;g) \supset_i \mathit{di}\ f$
proof –
have *1*: $\vdash f\ ;\ (g;\mathit{true}_i) \supset_i \mathit{di}\ f$ **by** (*rule ChopImpDi*)
have *2*: $\vdash f\ ;\ (g;\mathit{true}_i) \equiv_i (f;g);\mathit{true}_i$ **by** (*rule ChopAssoc*)
from *1 2* **show** *?thesis* **by** (*simp add*: *di-d-def*)
qed

lemma *BiBiOrImpBi*:
$\vdash \mathit{bi}\ (\ \mathit{bi}\ f \vee_i \mathit{bi}\ g) \supset_i \mathit{bi}\ f \vee_i \mathit{bi}\ g$
using *BiElim* **by** *auto*

lemma *BiImpBiBiOr*:
$\vdash \mathit{bi}\ f \supset_i \mathit{bi}\ (\ \mathit{bi}\ f \vee_i \mathit{bi}\ g)$
proof –
have *1*: $\vdash \mathit{bi}\ f \supset_i \mathit{bi}\ f \vee_i \mathit{bi}\ g$ **by** *auto*
hence *2*: $\vdash \mathit{bi}\ (\mathit{bi}\ f) \supset_i \mathit{bi}(\mathit{bi}\ f \vee_i \mathit{bi}\ g)$ **using** *BiImpBiRule* **by** *blast*

have *3*: $\vdash bi\ (bi\ f) \equiv_i bi\ f$ **using** *BiEqvBiBi itl-prop*(*30*) **by** *blast*
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *BiImpBiBiOrB*:
$\vdash bi\ g\ \supset_i bi\ (\ bi\ f \vee_i bi\ g)$
proof –
have *1*: $\vdash bi\ g \supset_i bi\ f\ \vee_i bi\ g$ **by** *auto*
hence *2*: $\vdash bi\ (bi\ g) \supset_i bi(bi\ f\ \vee_i bi\ g)$ **using** *BiImpBiRule* **by** *blast*
have *3*: $\vdash bi\ (bi\ g) \equiv_i bi\ g$ **using** *BiEqvBiBi itl-prop*(*30*) **by** *blast*
from *2 3* **show** *?thesis* **by** *auto*
qed

lemma *BiBiOrEqvBi*:
$\vdash bi\ (\ bi\ f \vee_i bi\ g) \equiv_i bi\ f \vee_i bi\ g$
proof –
have *1*: $\vdash bi\ (\ bi\ f \vee_i bi\ g) \supset_i bi\ f \vee_i bi\ g$ **by** (*rule BiBiOrImpBi*)
have *2*: $\vdash bi\ f\ \supset_i bi\ (\ bi\ f \vee_i bi\ g)$ **by** (*rule BiImpBiBiOr*)
have *3*: $\vdash bi\ g\ \supset_i bi\ (\ bi\ f \vee_i bi\ g)$ **by** (*rule BiImpBiBiOrB*)
have *4*: $\vdash bi\ f\ \vee_i bi\ g \supset_i bi\ (\ bi\ f \vee_i bi\ g)$ **using** *2 3* **by** *auto*
from *1 4* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *DiEqvOrDiChopSkipA*:
$\vdash di\ f \equiv_i f\ \vee_i di(f;skip)$
proof –
have *1*: $\vdash di\ f \equiv_i f\ ;true_i$ **by** (*simp add*: *di-d-def*)
hence *2*: $\vdash di\ f \equiv_i f;\ (\ empty \vee_i more)$ **by** *auto*
hence *3*: $\vdash f;\ (\ empty \vee_i more) \equiv_i f;empty \vee_i f;more$ **using** *ChopOrEqv* **by** *blast*
have *4*: $\vdash f;empty \equiv_i f$ **by** (*rule ChopEmpty*)
have *5*: $\vdash more \equiv_i skip;true_i$ **using** *MoreEqvSkipChopTrue* **by** *blast*
hence *6*: $\vdash f;more \equiv_i f;(skip;true_i)$ **using** *RightChopEqvChop* **by** *blast*
have *7*: $\vdash f;(skip;true_i) \equiv_i (f;skip);true_i$ **by** (*rule ChopAssoc*)
from *2 3 4 6 7* **show** *?thesis* **by** *auto*
qed

lemma *DiEqvOrDiChopSkipB*:
$\vdash di\ f \equiv_i f\ \vee_i (di\ f);skip$
proof –
have *1*: $\vdash (di\ f) \equiv_i f\ \vee_i di(f;skip)$ **by** (*rule DiEqvOrDiChopSkipA*)
have *2*: $\vdash\ di(f;skip) \equiv_i (f;skip);true_i$ **by** (*simp add*: *di-d-def*)
have *3*: $\vdash (f;skip);true_i \equiv_i f;(skip;true_i)$ **by** (*rule ChopAssocB*)
have *4*: $\vdash di(f;skip) \equiv_i f;(skip;true_i)$ **using** *2 3* **by** *auto*
have *5*: $\vdash skip;true_i \equiv_i true_i;skip$ **by** (*rule SkipTrueEqvTrueSkip*)
hence *6*: $\vdash f;(skip;true_i) \equiv_i f;(true_i;skip)$ **using** *RightChopEqvChop* **by** *blast*
have *7*: $\vdash di(f;skip) \equiv_i f;(true_i;skip)$ **using** *4 6* **by** *auto*
have *8*: $\vdash f;(true_i;skip) \equiv_i (f;true_i);skip$ **by** (*rule ChopAssoc*)
have *9*: $\vdash (f;true_i);skip \equiv_i (di\ f);skip$ **by** (*simp add*: *di-d-def*)
have *10* : $\vdash di(f;skip) \equiv_i (di\ f);skip$ **using** *7 8 9* **by** *auto*
hence *11*: $\vdash f\ \vee_i di(f;skip) \equiv_i f\ \vee_i (di\ f);skip$ **by** *auto*

from *1 11* **show** *?thesis* **using** *prop03* **by** *blast*
qed

lemma *BiEqvAndEmptyOrBiChopSkip*:
$\vdash bi\ f \equiv_i f \wedge_i (empty \vee_i (bi\ f);skip)$
proof –
have *1*: $\vdash di\ \neg_i f \equiv_i \neg_i f \vee_i (di\ \neg_i f;skip)$ **by** (*rule DiEqvOrDiChopSkipB*)
have *2*: $\vdash di\ \neg_i f \equiv_i \neg_i(bi\ f)$ **by** (*rule DiNotEqvNotBi*)
have *3*: $\vdash \neg_i(bi\ f) \equiv_i \neg_i f \vee_i (di\ \neg_i f;skip)$ **using** *1 2* **using** *itl-prop*(*30*) *prop03* **by** *blast*
hence *4*: $\vdash bi\ f \equiv_i \neg_i(\neg_i f \vee_i (di\ \neg_i f;skip))$ **by** (*metis 1 bi-d-def prop01*)
have *5*: $\vdash \neg_i(\neg_i f \vee_i (di\ \neg_i f;skip)) \equiv_i f \wedge_i \neg_i(di\ \neg_i f;skip)$ **by** *auto*
have *6*: $\vdash di\ \neg_i f;skip \equiv_i \neg_i(bi\ f);skip$ **by** *auto*
hence *7*: $\vdash \neg_i(di\ \neg_i f;skip) \equiv_i \neg_i(\neg_i(bi\ f);skip)$ **by** *auto*
have *8*: $\vdash \neg_i(\neg_i(bi\ f);skip) \equiv_i (empty \vee_i (bi\ f);skip)$ **using** *NotNotChopSkip* **by** *blast*
hence *9*: $\vdash f \wedge_i \neg_i(di\ \neg_i f;skip) \equiv_i f \wedge_i (empty \vee_i (bi\ f);skip)$ **using** *7 8* **by** *auto*
from *4 5 9* **show** *?thesis* **using** *prop03* **by** *blast*
qed

lemma *DiDiAndEqvDi*:
$\vdash di\ (di\ f \wedge_i di\ g) \equiv_i di\ f \wedge_i di\ g$
proof –
have *1*: $\vdash bi\ (bi\ \neg_i f \vee_i bi\ \neg_i g) \equiv_i bi\ \neg_i f \vee_i bi\ \neg_i g$
by (*rule BiBiOrEqvBi*)
have *2*: $\vdash bi\ \neg_i f \equiv_i \neg_i (di\ f)$
by *auto*
have *3*: $\vdash bi\ \neg_i g \equiv_i \neg_i (di\ g)$
by *auto*
have *4*: $\vdash bi\ \neg_i f \vee_i bi\ \neg_i g \equiv_i \neg_i (di\ f) \vee_i \neg_i (di\ g)$
using *2 3* **by** *auto*
have *5*: $\vdash \neg_i (di\ f) \vee_i \neg_i (di\ g) \equiv_i \neg_i(di\ f \wedge_i di\ g)$
by *auto*
have *6*: $\vdash bi\ (bi\ \neg_i f \vee_i bi\ \neg_i g) \equiv_i \neg_i(di\ f \wedge_i di\ g)$
using *1 5* **by** *auto*
hence *7*: $\vdash \neg_i(bi\ (bi\ \neg_i f \vee_i bi\ \neg_i g))\equiv_i (di\ f \wedge_i di\ g)$
by *simp*
have *8* : $\vdash \neg_i(bi\ (bi\ \neg_i f \vee_i bi\ \neg_i g)) \equiv_i di\ (\neg_i(bi\ \neg_i f \vee_i bi\ \neg_i g))$
using *DiNotEqvNotBi itl-prop*(*30*) **by** *blast*
have *9* : $\vdash \neg_i(bi\ \neg_i f \vee_i bi\ \neg_i g) \equiv_i di\ f \wedge_i di\ g$
by *auto*
hence *10*: $\vdash di\ (\neg_i(bi\ \neg_i f \vee_i bi\ \neg_i g)) \equiv_i di\ (di\ f \wedge_i di\ g)$
using *DiEqvDi* **by** *blast*
from *7 8 10* **show** *?thesis* **using** *itl-prop*(*30*) *prop03* **by** *blast*
qed

lemma *BiInduct*:
$\vdash bi(f \supset_i wprev\ f) \wedge_i f \supset_i bi\ f$
proof –
have *1*: $\vdash \Box((f^r) \supset_i wnext(f^r)) \wedge_i f^r \supset_i \Box(f^r)$ **using** *BoxInduct* **by** *blast*
hence *2*: $\vdash (\Box((f^r) \supset_i wnext(f^r)) \wedge_i f^r \supset_i \Box(f^r))^r$ **using** *ReverseEqv* **by** *blast*
have *3*: $\vdash ((f^r)^r\ ) \equiv_i f$ **using** *EqvReverseReverse itl-prop*(*30*) **by** *blast*

have *4*: $\vdash (\Box\ (f^r))^r \equiv_i bi\ (f)$ **using** *RRBoxEqvBi* **by** *blast*
have *5*: $\vdash ((f^r) \supset_i wnext(f^r))^r \equiv_i (\ (f^r)^r \supset_i (wnext(f^r))^r\ )$ **by** *simp*
have *6*: $\vdash (wnext(f^r))^r \equiv_i wprev(f)$ **using** *RRWNextEqvWPrev* **by** *blast*
have *7*: $\vdash (\ (f^r)^r \supset_i (wnext(f^r))^r\ ) \equiv_i (\ f \supset_i wprev(f)\ )$ **using** *6 3 prop39* **by** *auto*
have *8*: $\vdash bi(\ (f^r)^r \supset_i (wnext(f^r))^r\ ) \equiv_i bi(\ f \supset_i wprev(f)\ )$ **using** *7 3 BiEqvBi* **by** *blast*
have *9*: $\vdash (\Box((f^r) \supset_i wnext(f^r)))^r \equiv_i bi\ (\ ((f^r) \supset_i wnext(f^r))^r\ )$ **using** *RBoxEqvBi* **by** *blast*
have *10*: $\vdash (\Box((f^r) \supset_i wnext(f^r)))^r \equiv_i bi(\ f \supset_i wprev(f)\ )$ **using** *8 9* **by** *auto*
have *11*: $\vdash (\Box((f^r) \supset_i wnext(f^r)) \wedge_i f^r \supset_i \Box(f^r))^r \equiv_i$
$((\Box((f^r) \supset_i wnext(f^r)))^r \wedge_i (f^r)^r \supset_i (\Box(f^r))^r)$ **using** *RAnd* **by** *auto*
have *12*: $\vdash ((\Box((f^r) \supset_i wnext(f^r)))^r \wedge_i (f^r)^r \supset_i (\Box(f^r))^r) \equiv_i$
$(bi(\ f \supset_i wprev(f)\ ) \wedge_i f \supset_i bi\ f\ )$ **using** *8 3 4 10* **by** *simp*
from *2 11 12* **show** *?thesis* **using** *MP itl-prop(31)* **by** *blast*
qed

lemma *PrevLoop*:
assumes $\vdash f \supset_i prev\ f$
shows $\vdash \neg_i f$
proof −
have *1*: $\vdash f \supset_i prev\ f$ **using** *assms* **by** *auto*
hence *2*: $\vdash f \supset_i (\ more \wedge_i wprev\ f)$ **by** *auto*
hence *3*: $\vdash f \supset_i wprev\ f$ **by** *auto*
hence *4*: $\vdash bi(f \supset_i wprev\ f)$ **by** (*rule BiGen*)
have *5*: $\vdash bi(f \supset_i wprev\ f) \wedge_i f \supset_i bi\ f$ **by** (*rule BiInduct*)
hence *6*: $\vdash bi(f \supset_i wprev\ f) \supset_i (f \supset_i bi\ f)$ **using** *prop36* **by** *blast*
have *7*: $\vdash (f \supset_i bi\ f)$ **using** *4 6 MP* **by** *blast*
have *8*: $\vdash bi\ f \supset_i f$ **by** (*rule BiElim*)
have *9*: $\vdash f \equiv_i bi\ f$ **using** *7 8 itl-prop(31)* **by** *blast*
have *10*: $\vdash f \supset_i more$ **using** *2* **by** *auto*
hence *11*: $\vdash bi\ f \supset_i\ bi\ more$ **using** *BiImpBiRule* **by** *blast*
have *12*: $\vdash \neg_i(bi\ more)$ **using** *DiEmpty* **by** *auto*
from *7 9 11 12* **show** *?thesis* **using** *MP prop27* **by** *blast*
qed

lemma *PrevImpNotPrevNot*:
$\vdash prev\ f \supset_i \neg_i(prev\ \neg_i\ f)$
by *auto*

lemma *BiEqvAndWprevBi*:
$\vdash bi\ f \equiv_i f \wedge_i wprev(bi\ f)$
proof −
have *1*: $\vdash \Box\ (f^r) \equiv_i f^r \wedge_i wnext(\Box\ (f^r))$
using *BoxEqvAndWnextBox* **by** *blast*
hence *2*: $\vdash (\Box\ (f^r) \equiv_i f^r \wedge_i wnext(\Box\ (f^r)))^r$
using *ReverseEqv* **by** *blast*
have *3*: $\vdash (\Box\ (f^r))^r \equiv_i bi\ (f)$
using *RRBoxEqvBi* **by** *blast*
have *4*: $\vdash (f^r)^r \equiv_i f$
using *EqvReverseReverse itl-prop(30)* **by** *blast*
have *5*: $\vdash (wnext(\Box\ (f^r)))^r \equiv_i wprev(\ (\Box(f^r))^r\ )$
using *RWNextEqvWPrev* **by** *blast*

have *6*: $\vdash wprev(\ (\Box(\ ((f^r))))^r)\ \equiv_i\ wprev\ (bi(f))$
  using *3 5* **by** *auto*
have *7*: $\vdash (wnext(\Box\ (f^r)))^r\ \equiv_i\ \ wprev\ (bi\ (f))$
  using *5 6* **by** *auto*
have *8*: $\vdash (\Box\ (f^r)\ \equiv_i\ f^r\ \wedge_i\ wnext(\Box\ (f^r)))^r\ \equiv_i$
    $(\ (\Box\ (f^r))^r\ \equiv_i\ ((f^r)^r)\ \wedge_i\ (wnext(\Box\ (f^r)))^r)$
  by (*meson 1 2 RAnd REqvRule iff-defs prop03 valid-def*)
have *9*: $\vdash (\ (\Box\ (f^r))^r\ \equiv_i\ ((f^r)^r)\ \wedge_i\ (wnext(\Box\ (f^r)))^r)\ \equiv_i$
    $(\ (bi\ f)\ \equiv_i\ f\ \wedge_i\ wprev(bi\ f))$
  using *7 3 4 prop40* **by** *auto*
from *9 8 2* **show** *?thesis* **by** *auto*
qed

lemma *DiIntroLoop*:
assumes $\vdash (f\ \wedge_i\ \neg_i\ g)\ \supset_i\ prev\ f$
shows $\vdash f\ \supset_i\ di\ g$
proof −
have *1*: $\vdash f\ \wedge_i\ \neg_i\ \ g\ \supset_i\ prev\ f$
  using *assms* **by** *auto*
hence *2*: $\vdash f\ \wedge_i\ \neg_i\ \ g\ \wedge_i\ (bi\ \neg_i\ g)\supset_i\ (prev\ f)\ \wedge_i\ (bi\ \neg_i\ g)$
  by *auto*
have *3*: $\vdash (bi\ \neg_i\ g)\ \supset_i\ \neg_i\ \ g$
  by (*rule BiElim*)
hence *4*: $\vdash bi\ \neg_i\ g\ \equiv_i\ (bi\ \neg_i\ g)\ \wedge_i\ \neg_i\ \ g$
  using *prop38* **by** *blast*
have *5*: $\vdash f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ prev\ f\ \wedge_i\ bi\ \neg_i\ g$
  using *2 4* **by** *auto*
have *6*: $\vdash bi\ \neg_i\ g\ \equiv_i\ (\neg_i\ g)\ \wedge_i\ wprev(bi\ \neg_i\ g)$
  by (*rule BiEqvAndWprevBi*)
have *7*: $\vdash prev\ f\ \wedge_i\ bi\ \neg_i\ g\ \supset_i\ prev\ f\ \wedge_i\ wprev(bi\ \neg_i\ g)$
  using *6* **using** *itl-prop*(*31*) *itl-prop*(*32*) *prop12* **by** *blast*
have *8*: $\vdash f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ prev\ f\ \wedge_i\ wprev(bi\ \neg_i\ g)$
  using *5 7* **by** *auto*
hence *9*: $\vdash f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ more\ \wedge_i\ wprev\ f\ \wedge_i\ wprev(bi\ \neg_i\ g)$
  by *auto*
hence *10*: $\vdash f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ wprev\ f\ \wedge_i\ wprev(bi\ \neg_i\ g)$
  by *auto*
hence *11*: $\vdash f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ wprev\ (f\ \wedge_i\ bi\ \neg_i\ g)$
  by *auto*
hence *12*: $\vdash bi(f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ wprev\ (f\ \wedge_i\ bi\ \neg_i\ g)\ )$
  by (*rule BiGen*)
have *13*: $\vdash bi(f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ wprev\ (f\ \wedge_i\ bi\ \neg_i\ g)\ )\ \wedge_i\ f\ \wedge_i\ (bi\ \neg_i\ g)$
    $\supset_i\ bi(f\ \wedge_i\ (bi\ \neg_i\ g))$
  by (*rule BiInduct*)
hence *14*: $\vdash bi(f\ \wedge_i\ (bi\ \neg_i\ g)\ \supset_i\ wprev\ (f\ \wedge_i\ bi\ \neg_i\ g)\ )\ \supset_i$
    $(\ (f\ \wedge_i\ (bi\ \neg_i\ g))\ \supset_i\ bi(f\ \wedge_i\ (bi\ \neg_i\ g)))$
  using *prop36* **by** *blast*
have *15*: $\vdash (\ (f\ \wedge_i\ (bi\ \neg_i\ g))\ \supset_i\ bi(f\ \wedge_i\ (bi\ \neg_i\ g)))$
  using *12 14 MP* **by** *blast*
have *16*: $\vdash bi(f\ \wedge_i\ (bi\ \neg_i\ g))\ \supset_i\ f\ \wedge_i\ (bi\ \neg_i\ g)$

```
     by (rule BiElim)
 have 17: ⊢ bi(f ∧_i (bi ¬_i g)) ≡_i (f ∧_i (bi ¬_i g))
     using 16 15 itl-prop(31) by blast
 have 18: ⊢ (f ∧_i (bi ¬_i g)) ⊃_i more
     using 9 by auto
 hence 19: ⊢ bi (f ∧_i (bi ¬_i g)) ⊃_i bi more
     using BiImpBiRule by blast
 have 20: ⊢ ¬_i(bi more)
     using DiEmpty by auto
 have 21: ⊢ ¬_i(f ∧_i (bi ¬_i g))
     using 17 19 20 by fastforce
 hence 22: ⊢ ¬_i f ∨_i ¬_i (bi ¬_i g)
     by auto
 have 23: ⊢ ¬_i (bi ¬_i g) ≡_i di g
     by auto
 from 22 23 show ?thesis by auto
qed

lemma DiEqvOrChopMore:
⊢ di f ≡_i (f ∨_i f;more)
proof –
 have 1: ⊢ di f ≡_i f;true_i by auto
 hence 2: ⊢ di f ≡_i f; (empty ∨_i more) by auto
 have 3: ⊢ f; (empty ∨_i more) ≡_i f;empty ∨_i f;more by auto
 have 4: ⊢ f;empty ≡_i f by (rule ChopEmpty)
 from 2 3 4 show ?thesis by auto
qed

lemma DiAndDiEqvDiAndDiOrDiAndDi:
⊢ di f ∧_i di g ≡_i di(f ∧_i di g) ∨_i di(g ∧_i di f)
proof –
 have 1: ⊢ di f ≡_i (f ∨_i f;more)
     using DiEqvOrChopMore by blast
 have 2: ⊢ di g ≡_i (g ∨_i g;more)
     using DiEqvOrChopMore by blast
 have 3: ⊢ di f ∧_i di g ≡_i (f ∨_i f;more) ∧_i (g ∨_i g;more)
     using 1 2 by auto
 have 4: ⊢ (f ∨_i f;more) ∧_i (g ∨_i g;more) ≡_i
          (f ∧_i g) ∨_i ( f ∧_i g;more) ∨_i (g ∧_i f;more) ∨_i (f;more ∧_i g;more )
     by auto
 have 5: ⊢ more ≡_i true_i;skip
     using MoreEqvSkipChopTrue SkipTrueEqvTrueSkip prop03 by blast
 hence 6: ⊢ f;more ≡_i f;(true_i;skip)
     using RightChopEqvChop by blast
 have 7: ⊢ f;(true_i;skip) ≡_i (f;true_i);skip
     by (rule ChopAssoc)
 have 8: ⊢ f;more ≡_i prev (di f)
     using 6 7 by (simp add: prev-d-def)
 have 9: ⊢ g;more ≡_i g;(true_i;skip)
     using 5 RightChopEqvChop by blast
```

have *10*: $\vdash g;(true_i;skip) \equiv_i (g;true_i);skip$
  by (*rule ChopAssoc*)
have *11*: $\vdash g;more \equiv_i prev\ (di\ g)$
  using *9 10* **by** (*simp add*: *prev-d-def*)
have *12*: $\vdash f;more \wedge_i g;more \equiv_i prev\ (di\ f) \wedge_i prev\ (di\ g)$
  using *8 11* **by** *auto*
hence *13*: $\vdash f;more \wedge_i g;more \equiv_i prev\ (di\ f \wedge_i di\ g)$
  by *auto*
have *14*: $\vdash (di\ f \wedge_i di\ g) \equiv_i$
    $((f \wedge_i g) \vee_i (f \wedge_i g;more) \vee_i (g \wedge_i f;more)) \vee_i (f;more \wedge_i g;more)$
  using *3 4* **by** *auto*
have *15*: $\vdash (di\ f \wedge_i di\ g) \equiv_i$
    $((f \wedge_i g) \vee_i (f \wedge_i g;more) \vee_i (g \wedge_i f;more)) \vee_i prev\ (di\ f \wedge_i di\ g)$
  using *13 14 prop28* **by** *blast*
hence *16*: $\vdash (di\ f \wedge_i di\ g) \supset_i$
    $((f \wedge_i g) \vee_i (f \wedge_i g;more) \vee_i (g \wedge_i f;more)) \vee_i prev\ (di\ f \wedge_i di\ g)$
  using *itl-prop*(*31*) **by** *blast*
hence *17*: $\vdash (di\ f \wedge_i di\ g) \wedge_i \neg_i((f \wedge_i g) \vee_i (f \wedge_i g;more) \vee_i (g \wedge_i f;more)) \supset_i$
    $prev\ (di\ f \wedge_i di\ g)$
  using *prop29* **by** *blast*
hence *18*: $\vdash (di\ f \wedge_i di\ g) \supset_i di((f \wedge_i g) \vee_i (f \wedge_i g;more) \vee_i (g \wedge_i f;more))$
  using *DiIntroLoop* **by** *blast*
have *19*: $\vdash di((f \wedge_i g) \vee_i (f \wedge_i g;more) \vee_i (g \wedge_i f;more)) \equiv_i$
    $di(f \wedge_i g) \vee_i di(f \wedge_i g;more) \vee_i di(g \wedge_i f;more)$
  by *auto*
have *20*: $\vdash f \supset_i di\ f$
  using *DiIntro* **by** *blast*
hence *21*: $\vdash f \wedge_i g \supset_i g \wedge_i di\ f$
  by *auto*
hence *22*: $\vdash di(f \wedge_i g) \supset_i di(g \wedge_i di\ f)$
  using *DiImpDi* **by** *blast*
hence *23*: $\vdash di(f \wedge_i g) \supset_i di(g \wedge_i di\ f) \vee_i di(f \wedge_i di\ g)$
  by *auto*
have *24*: $\vdash g;more \supset_i di\ g$
  by *auto*
hence *25*: $\vdash f \wedge_i g;more \supset_i f \wedge_i di\ g$
  by *auto*
hence *26*: $\vdash di(f \wedge_i g;more) \supset_i di(f \wedge_i di\ g)$
  using *DiImpDi* **by** *blast*
hence *27*: $\vdash di(f \wedge_i g;more) \supset_i di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f)$
  by *auto*
have *28*: $\vdash f;more \supset_i di\ f$
  by *auto*
hence *29*: $\vdash g \wedge_i f;more \supset_i g \wedge_i di\ f$
  by *auto*
hence *30*: $\vdash di(g \wedge_i f;more) \supset_i di(g \wedge_i di\ f)$
  using *DiImpDi* **by** *blast*
hence *31*: $\vdash di(g \wedge_i f;more) \supset_i di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f)$
  by *auto*
have *32*: $\vdash di(f \wedge_i g) \vee_i di(f \wedge_i g;more) \vee_i di(g \wedge_i f;more) \supset_i$

$di(f \wedge_i di\ g) \vee_i\ di(g \wedge_i di\ f)$
using *23 27 31* **by** *auto*
have *33*: $\vdash di((f \wedge_i g) \vee_i (f \wedge_i g;more) \vee_i (g \wedge_i f;more)) \supset_i$
$di(f \wedge_i di\ g) \vee_i\ di(g \wedge_i di\ f)$
using *19 32* **by** *auto*
have *34*: $\vdash (di\ f \wedge_i di\ g) \supset_i di(f \wedge_i di\ g) \vee_i\ di(g \wedge_i di\ f)$
using *18 33* **by** *auto*
have *35*: $\vdash\ f \supset_i di\ f$
using *DiIntro* **by** *blast*
hence *36*: $\vdash f \wedge_i di\ g \supset_i di\ f \wedge_i di\ g$
by *auto*
hence *37*: $\vdash di\ (f \wedge_i di\ g) \supset_i di\ (di\ f \wedge_i di\ g)$
using *DiImpDi* **by** *blast*
have *38*: $\vdash di\ (di\ f \wedge_i di\ g) \equiv_i di\ f \wedge_i di\ g$
using *DiDiAndEqvDi* **by** *blast*
have *39*: $\vdash di\ (f \wedge_i di\ g) \supset_i di\ f \wedge_i di\ g$
using *37 38* **using** *itl-prop(31) prop02* **by** *blast*
have *40*: $\vdash\ g \supset_i di\ g$
using *DiIntro* **by** *blast*
hence *41*: $\vdash\ g \wedge_i di\ f \supset_i di\ f \wedge_i di\ g$
by *auto*
hence *42*: $\vdash di\ (g \wedge_i di\ f) \supset_i di\ (di\ f \wedge_i di\ g)$
using *DiImpDi* **by** *blast*
have *43*: $\vdash di\ (di\ f \wedge_i di\ g) \equiv_i di\ f \wedge_i di\ g$
using *DiDiAndEqvDi* **by** *blast*
have *44*: $\vdash di\ (g \wedge_i di\ f) \supset_i di\ f \wedge_i di\ g$
using *42 43* **using** *itl-prop(31) prop02* **by** *blast*
have *45*: $\vdash di\ (f \wedge_i di\ g) \vee_i di\ (g \wedge_i di\ f) \supset_i di\ f \wedge_i di\ g$
using *39 44 prop30* **by** *blast*
from *34 45* **show** *?thesis* **using** *itl-prop(31)* **by** *blast*
qed

lemma *BoxStateEqvBiFinState*:
$\vdash \Box\ (init\ w) \equiv_i bi\ (fin\ (init\ w))$
proof –
have *1*: $\vdash \Diamond\ (\neg_i\ (init\ w)) \equiv_i true_i\ ;\ \neg_i(init\ w)$
by *simp*
have *2*: $\vdash \Diamond\ (init(\neg_i\ w)) \equiv_i true_i\ ;\ init\ (\neg_i\ w)$
by *simp*
have *3*: $\vdash di\ (true_i \wedge_i fin\ (init\ (\neg_i\ w))) \equiv_i true_i\ ;\ init\ (\neg_i\ w)$
using *DiAndFinEqvChopState* **by** *blast*
have *4*: $\vdash \Diamond\ (init(\neg_i\ w)) \equiv_i di\ (true_i \wedge_i fin\ (init\ (\neg_i\ w)))$
using *1 2 3* **by** *simp*
have *5*: $\vdash \neg_i\ (\Diamond\ (init(\neg_i\ w))) \equiv_i \neg_i\ (di\ (true_i \wedge_i fin\ (init\ (\neg_i\ w))))$
using *4* **by** *simp*
have *6*: $\vdash \Box\ (init\ w) \equiv_i \neg_i\ (di\ (true_i \wedge_i fin\ (init\ (\neg_i\ w))))$
using *5* **by** *auto*
have *7*: $\vdash \Box\ (init\ w) \equiv_i bi\ (\neg_i\ (fin\ (init\ (\neg_i\ w))))$
using *6* **by** *auto*
have *8*: $\vdash init\ (\neg_i\ w) \equiv_i \neg_i\ (init\ w)$

by *simp*
have *9*: $\vdash fin\ (init\ (\neg_i\ w)) \equiv_i fin\ (\neg_i\ (init\ w))$
using *8 FinEqvFin* **by** *blast*
have *10*: $\vdash fin\ (init\ (\neg_i\ w)) \equiv_i \neg_i\ (fin\ (init\ w))$
using *8 FinNotStateEqvNotFinState FinEqvFin* **by** *blast*
have *11*: $\vdash \neg_i\ (fin\ (init\ (\neg_i\ w))) \equiv_i (fin\ (init\ w))$
using *10* **by** *simp*
have *12*: $\vdash bi\ (\neg_i\ (fin\ (init\ (\neg_i\ w)))) \equiv_i bi\ (fin\ (init\ w))$
using *11* **by** *simp*
have *13*: $\vdash \Box\ (init\ w) \equiv_i bi\ (fin\ (init\ w))$
using *7 12* **by** *simp*
from *13* **show** *?thesis* **by** *simp*
qed

lemma *DiamondStateEqvDiFinState*:
$\vdash \Diamond\ (init\ w) \equiv_i di\ (fin\ (init\ w))$
proof –
have *1*: $\vdash \Box\ (init\ (\neg_i\ w)) \equiv_i bi\ (fin\ (init\ (\neg_i\ w)))$ **using** *BoxStateEqvBiFinState* **by** *blast*
have *2*: $\vdash \neg_i\ (\Box\ (init\ (\neg_i\ w))) \equiv_i \neg_i\ (bi\ (fin\ (init\ (\neg_i\ w))))$ **using** *1* **by** *auto*
have *3*: $\vdash \Diamond\ (\neg_i\ (init\ (\neg_i\ w))) \equiv_i di\ (\neg_i\ (fin\ (init\ (\neg_i\ w))))$ **using** *2* **by** *auto*
have *4*: $\vdash \Diamond\ (init\ \ w) \equiv_i di\ (\neg_i\ (fin\ (init\ (\neg_i\ w))))$ **using** *3* **by** *auto*
have *5*: $\vdash \Diamond\ (init\ \ w) \equiv_i di\ (fin\ (init\ w))$ **using** *4 FinNotStateEqvNotFinState* **by** *auto*
from *1 2 3 4 5* **show** *?thesis* **by** *simp*
qed

lemma *OrDiEqvDi*:
$\vdash f \lor_i di\ f \equiv_i di\ f$
proof –
have *1*: $\vdash f \supset_i di\ f$ **using** *DiIntro* **by** *blast*
from *1* **show** *?thesis* **by** *auto*
qed

lemma *AndDiEqv*:
$\vdash f \land_i di\ f \equiv_i f$
proof –
have *1*: $\vdash f \supset_i di\ f$ **using** *DiIntro* **by** *blast*
from *1* **show** *?thesis* **by** *auto*
qed

lemma *BiEmptyEqvEmpty*:
$\vdash bi\ empty \equiv_i empty$
proof –
have *1*: $\vdash bi\ empty \equiv_i \neg_i\ (di\ \neg_i\ empty)$ **by** (*simp add*: *bi-d-def*)
have *2*: $\vdash \neg_i\ (di\ \neg_i\ empty) \equiv_i \neg_i\ (\neg_i\ empty;true_i)$ **by** (*simp add*: *di-d-def*)
have *3*: $\vdash \neg_i\ (\neg_i\ empty;true_i) \equiv_i \neg_i\ (more;true_i)$ **by** *auto*
have *4*: $\vdash more;true_i \equiv_i\ \ more$ **using** *MoreEqvMoreChopTrue* **by** *auto*
hence *5*: $\vdash \neg_i(more;true_i) \equiv_i\ \ \neg_i\ more$ **using** *prop01* **by** *blast*
from *1 2 3 5* **show** *?thesis* **by** *auto*
qed

lemma *EmptyChopSkipInduct*:
assumes $\vdash \mathit{empty} \supset_i f$
$\vdash \mathit{prev}\ f \supset_i f$
shows $\vdash f$
proof −
have *1*: $\vdash \mathit{empty} \supset_i f$ **using** *assms*(*1*) **by** *auto*
have *2*: $\vdash \mathit{prev}\ f \supset_i f$ **using** *assms*(*2*) **by** *blast*
have *3*: $\vdash (\mathit{empty} \vee_i \mathit{prev}\ f) \supset_i f$ **using** *1 2 prop30* **by** *blast*
have *4*: $\vdash \mathit{wprev}\ f \equiv_i (\mathit{empty} \vee_i \mathit{prev}\ f)$ **by** *auto*
hence *5*: $\vdash \mathit{wprev}\ f \supset_i f$ **using** *3* **using** *itl-prop*(*31*) *prop02* **by** *blast*
hence *6*: $\vdash \neg_i f \supset_i \neg_i (\mathit{wprev}\ f)$ **using** *prop27* **by** *blast*
hence *7*: $\vdash \neg_i f \supset_i \mathit{prev}\ (\neg_i f)$ **by** *auto*
hence *8*: $\vdash \neg_i \neg_i f$ **by** (*rule PrevLoop*)
from *8* **show** *?thesis* **by** *auto*
qed

lemma *MoreImpImpChopSkipEqv*:
$\vdash \mathit{more} \supset_i ((f \supset_i g);\mathit{skip} \equiv_i ((f;\mathit{skip}) \supset_i (g;\mathit{skip})))$
proof −
have *1*: $\vdash \mathit{more} \wedge_i (f \supset_i g);\mathit{skip} \equiv_i \mathit{more} \wedge_i (\neg_i f \vee_i g);\mathit{skip}$
by *auto*
have *2*: $\vdash (\neg_i f \vee_i g);\mathit{skip} \equiv_i \neg_i f;\mathit{skip} \vee_i g;\mathit{skip}$
using *OrChopEqv* **by** *auto*
hence *3*: $\vdash \mathit{more} \wedge_i (\neg_i f \vee_i g);\mathit{skip} \equiv_i \mathit{more} \wedge_i (\neg_i f;\mathit{skip} \vee_i g;\mathit{skip})$
by *auto*
have *4*: $\vdash \neg_i(\neg_i f;\mathit{skip}) \equiv_i \mathit{empty} \vee_i (f;\mathit{skip})$
using *NotNotChopSkip* **by** *blast*
hence *5*: $\vdash (\neg_i f;\mathit{skip}) \equiv_i \neg_i(\mathit{empty} \vee_i (f;\mathit{skip}))$
using *itl-prop*(*30*) *itl-prop*(*33*) *itl-prop*(*4*) *prop03* **by** *blast*
have *6*: $\vdash \neg_i(\mathit{empty} \vee_i (f;\mathit{skip})) \equiv_i (\mathit{more} \wedge_i \neg_i(f;\mathit{skip}))$
by *auto*
have *7*: $\vdash (\neg_i f;\mathit{skip} \vee_i g;\mathit{skip}) \equiv_i ((\mathit{more} \wedge_i \neg_i(f;\mathit{skip})) \vee_i g;\mathit{skip})$
using *5 6* **by** *auto*
hence *8*: $\vdash \mathit{more} \wedge_i(\neg_i f;\mathit{skip} \vee_i g;\mathit{skip}) \equiv_i \mathit{more} \wedge_i ((\mathit{more} \wedge_i \neg_i(f;\mathit{skip})) \vee_i g;\mathit{skip})$
by *auto*
have *9*: $\vdash \mathit{more} \wedge_i ((\mathit{more} \wedge_i \neg_i(f;\mathit{skip})) \vee_i g;\mathit{skip}) \equiv_i \mathit{more} \wedge_i (\neg_i(f;\mathit{skip}) \vee_i g;\mathit{skip})$
by *auto*
have *10*: $\vdash \mathit{more} \wedge_i (\neg_i(f;\mathit{skip}) \vee_i g;\mathit{skip}) \equiv_i \mathit{more} \wedge_i ((f;\mathit{skip}) \supset_i (g;\mathit{skip}))$
by *auto*
have *11*: $\vdash \mathit{more} \wedge_i (f \supset_i g);\mathit{skip} \equiv_i \mathit{more} \wedge_i ((f;\mathit{skip}) \supset_i (g;\mathit{skip}))$
using *1 2 3 8 9 10* **by** *auto*
from *11* **show** *?thesis* **using** *prop31* **using** *MP itl-prop*(*31*) **by** *blast*
qed

lemma *MoreImpImpPrevEqv*:
$\vdash \mathit{more} \supset_i (\mathit{prev}(f \supset_i g) \equiv_i (\mathit{prev}\ f \supset_i \mathit{prev}\ g))$
using *MoreImpImpChopSkipEqv* **by** *auto*

lemma *BiBoxNotEqvNotTrueChopChopTrue*:
$\vdash \mathit{bi}(\Box \neg_i f) \equiv_i \neg_i((\mathit{true}_i;f);\mathit{true}_i)$

by *auto*

lemma *DiAndEmptyEqvAndEmpty*:
$\vdash \mathit{di}\ f \wedge_i \mathit{empty} \equiv_i f \wedge_i \mathit{empty}$
proof −
have *1* : $\vdash \mathit{di}\ f \equiv_i (f \vee_i \mathit{di}\ f;\mathit{skip})$ **using** *DiEqvOrDiChopSkipB* **by** *blast*
hence *2*: $\vdash \mathit{di}\ f \wedge_i \mathit{empty} \equiv_i (f \vee_i \mathit{di}\ f;\mathit{skip}) \wedge_i \mathit{empty}$ **using** *prop06* **by** *blast*
have *3* : $\vdash (f \vee_i \mathit{di}\ f;\mathit{skip}) \wedge_i \mathit{empty} \equiv_i (f \wedge_i \mathit{empty}) \vee_i (\mathit{di}\ f;\mathit{skip} \wedge_i \mathit{empty})$ **by** *auto*
have *4*: $\vdash \neg_i(\mathit{di}\ f;\mathit{skip} \wedge_i \mathit{empty})$ **by** *auto*
hence *5* : $\vdash (f \wedge_i \mathit{empty}) \vee_i (\mathit{di}\ f;\mathit{skip} \wedge_i \mathit{empty}) \equiv_i (f \wedge_i \mathit{empty})$ **by** *auto*
from *2 3 5* **show** *?thesis* **by** *auto*
qed

6.4.3 Strict initial intervals

lemma *DsMoreDi*:
$\vdash \mathit{ds}\ f \equiv_i \mathit{more} \wedge_i (\mathit{di}\ f);\mathit{skip}$
proof −
have *1*: $\vdash \mathit{ds}\ f \equiv_i \neg_i(\mathit{bs}\ \neg_i\ f)$
by (*simp add*: *ds-d-def*)
have *2*: $\vdash \neg_i(\mathit{bs}\ \neg_i\ f) \equiv_i \neg_i(\mathit{empty} \vee_i (\mathit{bi}\ \neg_i\ f);\mathit{skip})$
by (*simp add*: *bs-d-def*)
have *3*: $\vdash \neg_i(\mathit{empty} \vee_i (\mathit{bi}\ \neg_i\ f);\mathit{skip}) \equiv_i \neg_i \mathit{empty} \wedge_i \neg_i((\mathit{bi}\ \neg_i f);\mathit{skip})$
by *auto*
have *4*: $\vdash \neg_i \mathit{empty} \wedge_i \neg_i((\mathit{bi}\ \neg_i f);\mathit{skip}) \equiv_i \mathit{more} \wedge_i \neg_i((\mathit{bi}\ \neg_i f);\mathit{skip})$
by *auto*
have *5*: $\vdash \mathit{more} \wedge_i \neg_i((\mathit{bi}\ \neg_i f);\mathit{skip}) \equiv_i \mathit{more} \wedge_i \neg_i(\neg_i(\mathit{di}\ f);\mathit{skip})$
by *auto*
have *6*: $\vdash \mathit{more} \wedge_i \neg_i(\neg_i(\mathit{di}\ f);\mathit{skip}) \equiv_i \mathit{more} \wedge_i (\mathit{empty} \vee_i (\mathit{di}\ f);\mathit{skip})$
using *NotNotChopSkip* **using** *prop05* **by** *blast*
have *7*: $\vdash \mathit{more} \wedge_i (\mathit{empty} \vee_i (\mathit{di}\ f);\mathit{skip}) \equiv_i \mathit{more} \wedge_i (\mathit{di}\ f);\mathit{skip}$
by *auto*
from *1 2 3 4 5 6 7* **show** *?thesis* **by** *auto*
qed

lemma *DsDi*:
$\vdash \mathit{ds}\ f \equiv_i (\mathit{di}\ f);\mathit{skip}$
proof −
have *1*: $\vdash \mathit{ds}\ f \equiv_i \mathit{more} \wedge_i (\mathit{di}\ f);\mathit{skip}$ **by** (*rule DsMoreDi*)
have *2*: $\vdash (\mathit{di}\ f);\mathit{skip} \supset_i \mathit{more}$ **by** *auto*
hence *3*: $\vdash \mathit{more} \wedge_i (\mathit{di}\ f);\mathit{skip} \equiv_i (\mathit{di}\ f);\mathit{skip}$ **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *BsEqvNotDsNot*:
$\vdash \mathit{bs}\ f \equiv_i \neg_i(\mathit{ds}\ \neg_i\ f)$
proof −
have *1*: $\vdash \mathit{ds}\ \neg_i\ f \equiv_i \mathit{more} \wedge_i (\mathit{di}\ \neg_i\ f);\mathit{skip}$ **by** (*rule DsMoreDi*)
hence *2*: $\vdash \neg_i(\mathit{ds}\ \neg_i\ f) \equiv_i \neg_i(\mathit{more} \wedge_i (\mathit{di}\ \neg_i\ f);\mathit{skip})$ **by** *auto*

have *3*: $\vdash \neg_i(more \wedge_i (di\ \neg_i\ f);skip) \equiv_i empty \vee_i \neg_i((di\ \neg_i\ f);skip)$ **by** *auto*
have *4*: $\vdash empty \vee_i \neg_i((di\ \neg_i\ f);skip) \equiv_i empty \vee_i \neg_i(\neg_i(bi\ f);skip)$ **by** *auto*
have *5*: $\vdash \neg_i(\neg_i(bi\ f);skip) \equiv_i empty \vee_i (bi\ f);skip$ **by** (*rule NotNotChopSkip*)
hence *6*: $\vdash empty \vee_i \neg_i(\neg_i(bi\ f);skip) \equiv_i empty \vee_i (bi\ f);skip$ **by** *auto*
from *2 3 4 6* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *NotBsEqvDsNot*:
$\vdash \neg_i(bs\ f) \equiv_i ds\ \neg_i\ f$
proof −
have *1*: $\vdash bs\ f \equiv_i \neg_i(ds\ \neg_i\ f)$ **by** (*rule BsEqvNotDsNot*)
hence *2*: $\vdash \neg_i(\ bs\ f) \equiv_i \neg_i\neg_i(ds\ \neg_i\ f)$ **by** *auto*
from *2* **show** *?thesis* **by** *auto*
qed

lemma *NotDsEqvBsNot*:
$\vdash \neg_i\ (ds\ f) \equiv_i bs\ \neg_i\ f$
proof −
have *1*: $\vdash \neg_i(ds\ \ f) \equiv_i \neg_i\neg_i(bs\ \neg_i f)$ **by** (*simp add*: *ds-d-def*)
from *1* **show** *?thesis* **by** *auto*
qed

lemma *NotDsAndEmpty*:
$\vdash \neg_i(ds\ f \wedge_i empty)$
proof −
have *1*: $\vdash ds\ f \equiv_i more \wedge_i (di\ f);skip$ **by** (*rule DsMoreDi*)
have *2*: $\vdash more \wedge_i (di\ f);skip \wedge_i empty \supset_i false_i$ **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *BsMoreEqvEmpty*:
$\vdash bs\ more \equiv_i empty$
proof −
have *1*: $\vdash bs\ more \equiv_i empty \vee_i (bi\ more);skip$ **by** (*simp add*: *bs-d-def*)
have *2*: $\vdash bi\ more \supset_i false_i$ **using** *DiEmpty* **by** *auto*
hence *3*: $\vdash (bi\ more);skip \supset_i false_i$ **by** *auto*
hence *4*: $\vdash empty \vee_i ((bi\ more);skip) \equiv_i empty$ **using** *prop25* **by** *blast*
from *1 4* **show** *?thesis* **by** *auto*
qed

lemma *BsAndEqv*:
$\vdash bs\ f \wedge_i bs\ g \equiv_i bs(f \wedge_i g)$
proof −
have *1*: $\vdash bs\ f \equiv_i empty \vee_i (bi\ f)\ ;skip$
by (*simp add*: *bs-d-def*)
have *2*: $\vdash bs\ g \equiv_i empty \vee_i (bi\ g)\ ;skip$
by (*simp add*: *bs-d-def*)
have *3*: $\vdash bs\ f \wedge_i bs\ g \equiv_i (empty \vee_i (bi\ f)\ ;skip) \wedge_i (empty \vee_i (bi\ g)\ ;skip)$
using *1 2* **by** *auto*
have *4*: $\vdash (empty \vee_i (bi\ f)\ ;skip) \wedge_i (empty \vee_i (bi\ g)\ ;skip) \equiv_i$

$empty \vee_i ((bi\ f)\ ;skip \wedge_i (bi\ g)\ ;skip)$
by *auto*
have *5*: $\vdash ((bi\ f)\ ;skip \wedge_i (bi\ g)\ ;skip) \equiv_i bi(f \wedge_i g);skip$
using *BiAndChopSkipEqv itl-prop*(*30*) **by** *blast*
hence *6*: $\vdash empty \vee_i ((bi\ f)\ ;skip \wedge_i (bi\ g)\ ;skip) \equiv_i empty \vee_i bi(f \wedge_i g);skip$
by *auto*
from *3 4 6* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *DsEqvRule*:
assumes $\vdash f \equiv_i g$
shows $\vdash ds\ f \equiv_i ds\ g$
by (*meson DiEqvDi DsDi LeftChopEqvChop assms itl-prop*(*30*) *prop03*)

lemma *DsOrEqv*:
$\vdash ds\ f \vee_i ds\ g \equiv_i ds\ (f \vee_i g)$
proof −
have *1*: $\vdash ds\ f \equiv_i \neg_i(bs\ \neg_i\ f)$ **by** (*simp add*: *ds-d-def*)
have *2*: $\vdash ds\ g \equiv_i \neg_i(bs\ \neg_i\ g)$ **by** (*simp add*: *ds-d-def*)
have *3*: $\vdash ds\ f \vee_i ds\ g \equiv_i \neg_i(bs\ \neg_i\ f) \vee_i \neg_i(bs\ \neg_i\ g)$ **using** *1 2* **by** *auto*
have *4*: $\vdash \neg_i(bs\ \neg_i\ f) \vee_i \neg_i(bs\ \neg_i\ g) \equiv_i \neg_i(bs\ \neg_i\ f \wedge_i bs\ \neg_i\ g)$ **by** *auto*
have *5*: $\vdash bs\ \neg_i\ f \wedge_i bs\ \neg_i\ g \equiv_i bs(\ \neg_i\ f \wedge_i \neg_i g)$ **by** (*rule BsAndEqv*)
hence *6*: $\vdash \neg_i(bs\ \neg_i\ f \wedge_i bs\ \neg_i\ g) \equiv_i \neg_i(bs(\ \neg_i\ f \wedge_i \neg_i g))$ **by** *auto*
have *7*: $\vdash \neg_i(bs(\ \neg_i\ f \wedge_i \neg_i g)) \equiv_i ds\ (\neg_i(\neg_i\ f \wedge_i \neg_i g))$ **by** (*rule NotBsEqvDsNot*)
have *8*: $\vdash \neg_i(\neg_i\ f \wedge_i \neg_i g) \equiv_i (f \vee_i g)$ **by** *auto*
hence *9*: $\vdash ds(\neg_i(\neg_i\ f \wedge_i \neg_i g)) \equiv_i ds\ (f \vee_i g)$ **by** (*rule DsEqvRule*)
from *3 4 6 7 9* **show** *?thesis* **by** *auto*
qed

lemma *BsOrImp*:
$\vdash bs\ f \vee_i bs\ g \supset_i bs(f \vee_i g)$
proof −
have *1*: $\vdash bi\ f \vee_i bi\ g \supset_i bi(f \vee_i g)$
by (*rule BiOrBiImpBiOr*)
hence *2*: $\vdash (bi\ f \vee_i bi\ g);skip \supset_i (bi(f \vee_i g));skip$
by (*rule LeftChopImpChop*)
have *3*: $\vdash (bi\ f);skip \vee_i (bi\ g);skip \supset_i (bi(f \vee_i g));skip$
using *1 OrChopEqv 2 itl-prop*(*31*) *prop02* **by** *blast*
hence *4*: $\vdash empty \vee_i (bi\ f);skip \vee_i (bi\ g);skip \supset_i\ empty \vee_i (bi(f \vee_i g));skip$
by *auto*
hence *5*: $\vdash empty \vee_i (bi\ f);skip \vee_i empty \vee_i (bi\ g);skip \supset_i\ empty \vee_i (bi(f \vee_i g));skip$
by *auto*
from *5* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *DsAndImp*:
$\vdash ds\ (f \wedge_i g) \supset_i ds\ f \wedge_i ds\ g$
proof −
have *1*: $\vdash bs\ \neg_i f \vee_i bs\ \neg_i g \supset_i bs(\neg_i f \vee_i \neg_i g)$ **by** (*rule BsOrImp*)
have *2*: $\vdash \neg_i f \vee_i \neg_i g \equiv_i \neg_i(f \wedge_i g)$ **by** *auto*

hence *3*: $\vdash bs(\neg_i f \vee_i \neg_i g) \equiv_i bs\ \neg_i(f \wedge_i g)$ **by** (*rule BsEqvRule*)
have *4*: $\vdash bs\ \neg_i f \vee_i bs\ \neg_i g \supset_i bs\ \neg_i(f \wedge_i g)$ **using** *1 3* **by** *auto*
have *5*: $\vdash bs\ \neg_i f \equiv_i \neg_i(ds\ f)$ **using** *NotDsEqvBsNot* **by** *auto*
have *6*: $\vdash bs\ \neg_i g \equiv_i \neg_i(ds\ g)$ **using** *NotDsEqvBsNot* **by** *auto*
have *7*: $\vdash bs\ \neg_i(f \wedge_i g) \equiv_i \neg_i(ds\ (f \wedge_i g))$ **using** *NotDsEqvBsNot* **by** *auto*
have *8*: $\vdash \neg_i(ds\ f) \vee_i \neg_i(ds\ g) \supset_i \neg_i(ds\ (f \wedge_i g))$ **using** *4 5 6 7* **by** *auto*
hence *9*: $\vdash \neg_i(ds\ f \wedge_i ds\ g) \supset_i \neg_i(ds\ (f \wedge_i g))$ **by** *auto*
from *9* **show** *?thesis* **by** *auto*
qed

lemma *DsAndImpElimL*:
$\vdash ds\ (f \wedge_i g) \supset_i ds\ f$
using *DsAndImp* **by** *auto*

lemma *DsAndImpElimR*:
$\vdash ds\ (f \wedge_i g) \supset_i ds\ g$
using *DsAndImp* **by** *auto*

lemma *BiImpBs*:
$\vdash bi\ f \supset_i bs\ f$
proof –
have *1*: $\vdash empty \supset_i empty \vee_i (bi\ f);skip$ **by** *auto*
hence *2*: $\vdash empty \wedge_i bi\ f \supset_i empty \vee_i (bi\ f);skip$ **by** *auto*
have *2*: $\vdash more \wedge_i bi\ f \supset_i (bi\ f);skip$ **by** (*rule MoreAndBiImpBiChopSkip*)
hence *3*: $\vdash more \wedge_i bi\ f \supset_i empty \vee_i (bi\ f);skip$ **by** *auto*
have *4*: $\vdash bi\ f \equiv_i (bi\ f \wedge_i empty) \vee_i (bi\ f \wedge_i more)$ **by** *auto*
have *5*: $\vdash empty \vee_i (bi\ f);skip \equiv_i bs\ f$ **by** (*simp add*: *bs-d-def*)
from *2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *BsImpBsBs*:
$\vdash bs\ f \supset_i bs\ (\ bs\ f)$
proof –
have *1*: $\vdash bi\ f \supset_i bs\ f$ **by** (*rule BiImpBs*)
hence *2*: $\vdash bi\ (bi\ f) \supset_i bi(bs\ f)$ **by** (*rule BiImpBiRule*)
hence *3*: $\vdash (bi\ f) \supset_i bi(bs\ f)$ **using** *BiEqvBiBi itl-prop*(*31*) *prop02* **by** *blast*
hence *4*: $\vdash (bi\ f);skip \supset_i (bi(bs\ f));skip$ **by** (*rule LeftChopImpChop*)
hence *5*: $\vdash empty \vee_i (bi\ f);skip \supset_i empty \vee_i (bi(bs\ f));skip$ **by** *auto*
from *5* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *DsImpDi*:
$\vdash ds\ f \supset_i di\ f$
proof –
have *1*: $\vdash bi\ \neg_i\ f \supset_i bs\ \neg_i\ f$ **by** (*rule BiImpBs*)
hence *2*: $\vdash \neg_i(bs\ \neg_i\ f) \supset_i \neg_i(bi\ \neg_i\ f)$ **by** (*rule prop27*)
from *2* **show** *?thesis* **using** *NotBsEqvDsNot DiEqvNotBiNot* **by** (*simp add*: *ds-d-def*)
qed

lemma *BsImpBsRule*:

assumes $\vdash f \supset_i g$
shows $\vdash bs\ f \supset_i bs\ g$
proof –
have *1*: $\vdash f \supset_i g$ **using** *assms* **by** *auto*
hence *2*: $\vdash bi\ f \supset_i bi\ g$ **by** (*rule BiImpBiRule*)
hence *3*: $\vdash (bi\ f);skip \supset_i (bi\ g);skip$ **by** (*rule LeftChopImpChop*)
hence *4*: $\vdash empty \vee_i (bi\ f);skip \supset_i empty \vee_i (bi\ g);skip$ **by** *auto*
from *4* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *DsChopImpDsB*:
$\vdash ds\ (f;g) \supset_i ds\ f$
proof –
have *1*: $\vdash di(f;g) \supset_i di\ f$ **by** (*rule DiChopImpDiB*)
hence *2*: $\vdash (di(f;g));skip \supset_i (di\ f);skip$ **by** (*rule LeftChopImpChop*)
from *2* **show** *?thesis* **using** *DsDi* **by** (*metis itl-prop*(*31*) *prop02*)
qed

lemma *NotBsImpBsNotChop*:
$\vdash bs\ \neg_i\ f \supset_i bs\ (\ \neg_i(f;g))$
proof –
have *1*: $\vdash ds\ (f;g) \supset_i ds\ f$ **by** (*rule DsChopImpDsB*)
hence *2*: $\vdash \neg_i(ds\ f) \supset_i \neg_i(ds\ (f;g))$ **by** (*rule prop27*)
from *2* **show** *?thesis* **using** *NotDsEqvBsNot* **by** *auto*
qed

lemma *BsOrBsEqvBsBiOrBi*:
$\vdash bs\ f \vee_i bs\ g \equiv_i bs(bi\ f \vee_i bi\ g)$
proof –
have *1*: $\vdash bs\ f \vee_i bs\ g \equiv_i empty \vee_i (bi\ f);skip \vee_i empty \vee_i (bi\ g);skip$
by (*simp add*: *bs-d-def*)
have *2*: $\vdash empty \vee_i (bi\ f);skip \vee_i empty \vee_i (bi\ g);skip \equiv_i empty \vee_i (bi\ f);skip \vee_i (bi\ g);skip$
by *auto*
have *3*: $\vdash (bi\ f);skip \vee_i (bi\ g);skip \equiv_i (bi\ f \vee_i bi\ g);skip$
using *OrChopEqv* **using** *itl-prop*(*30*) **by** *blast*
hence *4*: $\vdash empty \vee_i (bi\ f);skip \vee_i (bi\ g);skip \equiv_i empty \vee_i (bi\ f \vee_i bi\ g);skip$
by *auto*
have *5*: $\vdash bi\ (\ bi\ f \vee_i bi\ g) \equiv_i bi\ f \vee_i bi\ g$
by (*rule BiBiOrEqvBi*)
hence *6*: $\vdash bi\ (\ bi\ f \vee_i bi\ g);skip \equiv_i (bi\ f \vee_i bi\ g);skip$
using *LeftChopEqvChop* **by** *blast*
hence *7*: $\vdash empty \vee_i bi\ (\ bi\ f \vee_i bi\ g);skip \equiv_i empty \vee_i (bi\ f \vee_i bi\ g);skip$
by *auto*
from *1 2 4 7* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *DiOrDsEqvDi*:
$\vdash di\ f \vee_i ds\ f \equiv_i di\ f$

proof −
have *1*: $\vdash \mathit{di}\ f \supset_i \mathit{di}\ f \vee_i \mathit{ds}\ f$ **by** *auto*
have *2*: $\vdash \mathit{di}\ f \supset_i \mathit{di}\ f$ **by** *auto*
have *3*: $\vdash \mathit{ds}\ f \supset_i \mathit{di}\ f$ **by** (*rule DsImpDi*)
have *4*: $\vdash \mathit{di}\ f \vee_i \mathit{ds}\ f \supset_i \mathit{di}\ f$ **using** *2 3* **by** *auto*
from *1 4* **show** *?thesis* **by** *auto*
qed

lemma *DiAndDsEqvDs*:
$\vdash \mathit{di}\ f \wedge_i \mathit{ds}\ f \equiv_i \mathit{ds}\ f$
proof −
have *1*: $\vdash \mathit{di}\ f \wedge_i \mathit{ds}\ f \supset_i \mathit{ds}\ f$ **by** *auto*
have *2*: $\vdash \mathit{ds}\ f \supset_i \mathit{ds}\ f$ **by** *auto*
have *3*: $\vdash \mathit{ds}\ f \supset_i \mathit{di}\ f$ **by** (*rule DsImpDi*)
have *4*: $\vdash \mathit{ds}\ f \supset_i \mathit{di}\ f \wedge_i \mathit{ds}\ f$ **using** *2 3* **by** *auto*
from *1 4* **show** *?thesis* **by** *auto*
qed

lemma *OrDsEqvDi*:
$\vdash f \vee_i \mathit{ds}\ f \equiv_i \mathit{di}\ f$
proof −
have *1*: $\vdash \mathit{ds}\ f \equiv_i (\mathit{di}\ f);\mathit{skip}$ **by** (*rule DsDi*)
hence *2*: $\vdash f \vee_i \mathit{ds}\ f \equiv_i f \vee_i (\mathit{di}\ f);\mathit{skip}$ **by** *auto*
from *2* **show** *?thesis* **using** *DiEqvOrDiChopSkipB itl-prop*(*30*) *prop03* **by** *blast*
qed

lemma *AndBsEqvBi*:
$\vdash f \wedge_i \mathit{bs}\ f \equiv_i \mathit{bi}\ f$
proof −
have *1*: $\vdash f \wedge_i \mathit{bs}\ f \equiv_i f \wedge_i (\mathit{empty} \vee_i (\mathit{bi}\ f);\mathit{skip})$ **by** (*simp add*: *bs-d-def*)
from *1* **show** *?thesis* **using** *BiEqvAndEmptyOrBiChopSkip* **by** (*metis bs-d-def itl-prop*(*30*))
qed

lemma *BsEqvBsBi*:
$\vdash \mathit{bs}\ f \equiv_i \mathit{bs}\ (\mathit{bi}\ f)$
proof −
have *1*: $\vdash \mathit{bs}\ f \equiv_i \mathit{empty} \vee_i (\mathit{bi}\ f);\mathit{skip}$ **by** (*simp add*: *bs-d-def*)
have *2*: $\vdash \mathit{bi}\ f \equiv_i \mathit{bi}\ (\ \mathit{bi}\ f)$ **by** (*rule BiEqvBiBi*)
hence *3*: $\vdash (\mathit{bi}\ f);\mathit{skip} \equiv_i \mathit{bi}\ (\mathit{bi}\ f);\mathit{skip}$ **using** *LeftChopEqvChop* **by** *blast*
hence *4*: $\vdash \mathit{empty} \vee_i (\mathit{bi}\ f);\mathit{skip} \equiv_i \mathit{empty} \vee_i \mathit{bi}\ (\mathit{bi}\ f);\mathit{skip}$ **by** *auto*
from *1 4* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *StateImpBs*:
$\vdash \mathit{init}\ w \supset_i \mathit{bs}\ (\mathit{init}\ w)$
proof −
have *1*: $\vdash \mathit{init}\ w \equiv_i \mathit{bi}\ (\mathit{init}\ w)$ **by** (*rule StateEqvBi*)
have *2*: $\vdash \mathit{bi}\ (\mathit{init}\ w) \supset_i \mathit{bs}\ (\mathit{init}\ w)$ **by** (*rule BiImpBs*)
from *1 2* **show** *?thesis* **using** *StateImpBi prop02* **by** *blast*
qed

lemma *DsAndDsEqvDsAndDiOrDsAndDi*:
$\vdash ds\ f \wedge_i ds\ g \equiv_i ds\ (f \wedge_i di\ g) \vee_i ds(g \wedge_i di\ f)$
proof –
have *1*: $\vdash di\ f \wedge_i di\ g \equiv_i di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f)$
  by (*rule DiAndDiEqvDiAndDiOrDiAndDi*)
hence *2*: $\vdash (di\ f \wedge_i di\ g);skip \equiv_i (di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f));skip$
  by (*rule LeftChopEqvChop*)
have *3*: $\vdash (di\ f \wedge_i di\ g);skip \equiv_i (di\ f);skip \wedge_i (di\ g);skip$
  using *ChopSkipAndChopSkip* **using** *itl-prop*(*30*) **by** *blast*
have *4*: $\vdash (di\ f);skip \wedge_i (di\ g);skip \equiv_i (di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f));skip$
  using *2 3* **by** *auto*
have *5*: $\vdash (di(f \wedge_i di\ g) \vee_i di(g \wedge_i di\ f));skip \equiv_i di(f \wedge_i di\ g);skip \vee_i di(g \wedge_i di\ f);skip$
  using *OrChopEqv* **by** *blast*
have *6*: $\vdash ds\ f \equiv_i (di\ f);skip$
  using *DsDi* **by** *blast*
have *7*: $\vdash ds\ g \equiv_i (di\ g);skip$
  using *DsDi* **by** *blast*
have *8*: $\vdash (di\ f);skip \wedge_i (di\ g);skip \equiv_i ds\ f \wedge_i ds\ g$
  using *6 7* **by** *auto*
have *9*: $\vdash ds(f \wedge_i di\ g) \equiv_i di(f \wedge_i di\ g);skip$
  using *DsDi* **by** *blast*
have *10*: $\vdash ds(g \wedge_i di\ f) \equiv_i di(g \wedge_i di\ f);skip$
  using *DsDi* **by** *blast*
have *11*: $\vdash di(f \wedge_i di\ g);skip \vee_i di(g \wedge_i di\ f);skip \equiv_i ds(f \wedge_i di\ g) \vee_i ds(g \wedge_i di\ f)$
  using *9 10* **by** *auto*
from *4 5 8 11* **show** *?thesis* **by** *simp*
qed

lemma *BsEqvBiMoreImpChop*:
$\vdash bs\ f \equiv_i bi(more \supset_i f;skip)$
proof –
have *1*: $\vdash bs\ f \equiv_i empty \vee_i (bi\ f;skip)$
  by (*simp add*: *bs-d-def*)
have *2*: $\vdash \neg_i(\neg_i(bi\ f);skip) \equiv_i empty \vee_i (bi\ f;skip)$
  using *NotNotChopSkip* **by** *blast*
have *3*: $\vdash \neg_i(\neg_i(bi\ f);skip) \equiv_i \neg_i(di\ \neg_i f;skip)$
  by *auto*
have *4*: $\vdash \neg_i(di\ \neg_i f;skip) \equiv_i \neg_i((\neg_i f\ ;true_i);skip)$
  by (*simp add*:*di-d-def*)
have *5*: $\vdash \neg_i((\neg_i f\ ;true_i);skip) \equiv_i \neg_i(\neg_i f\ ;(true_i;skip))$
  using *ChopAssocB prop01* **by** *blast*
have *6*: $\vdash \neg_i(\neg_i f\ ;(true_i;skip)) \equiv_i \neg_i(\neg_i f\ ;(skip;true_i))$
  using *SkipTrueEqvTrueSkip* **using** *TrueChopSkipEqvSkipChopTrue RightChopEqvChop prop01* **by** *blast*
have *7*: $\vdash \neg_i(\neg_i f\ ;(skip;true_i)) \equiv_i \neg_i((\neg_i f\ ;skip);true_i)$
  using *ChopAssoc prop01* **by** *blast*
have *8*: $\vdash \neg_i((\neg_i f\ ;skip);true_i) \equiv_i \neg_i(di\ (\neg_i f;skip))$
  by (*simp add*: *di-d-def*)
have *9*: $\vdash \neg_i(di\ (\neg_i f;skip)) \equiv_i bi\ (\neg_i(\neg_i f\ ;skip))$

using *NotDiEqvBiNot* **by** *blast*
have *10*: $\vdash bi\ (\neg_i(\neg_i\ f\ ;skip)) \equiv_i bi(\ empty \vee_i (f;skip))$
using *NotNotChopSkip* **using** *BiEqvBi* **by** *blast*
have *11*: $\vdash bi(\ empty \vee_i (f;skip)) \equiv_i bi(\ \neg_i\ more \vee_i (f;skip))$
by *auto*
from *1 2 3 4 5 6 7 8 9 10 11* **show** *?thesis* **by** *auto*
qed

lemma *BsFalseEqvEmpty*:
$\vdash bs\ false_i \equiv_i empty$
proof –
have *1*: $\vdash bs\ false_i \equiv_i empty \vee_i bi\ false_i;skip$ **by** (*simp add*: *bs-d-def*)
have *2*: $\vdash \neg_i(bi\ false_i;skip)$ **by** *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *BoxMoreStateEqvBsFinState*:
$\vdash \Box(more \supset_i \neg_i\ (init\ w)) \equiv_i bs(\neg_i(fin(init\ w)))$
proof –
have *1*: $\vdash \Box(more \supset_i \neg_i\ (init\ w)) \equiv_i \neg_i(\Diamond(\neg_i(more \supset_i \neg_i\ (init\ w))))$
by *auto*
have *2*: $\vdash \neg_i(\Diamond(\neg_i(more \supset_i \neg_i\ (init\ w)))) \equiv_i \neg_i(true_i;(init\ w \wedge_i\ more))$
by *auto*
have *3*: $\vdash more \equiv_i true_i;\ skip$
using *MoreEqvSkipChopTrue SkipTrueEqvTrueSkip prop03* **by** *blast*
have *4*: $\vdash init\ w \wedge_i\ more \equiv_i init\ w \wedge_i (true_i;\ skip)$
using *3* **by** *auto*
have *5*: $\vdash init\ w \wedge_i (true_i;\ skip) \quad \equiv_i ((init\ w \wedge_i empty);(true_i;skip))$
using *StateAndEmptyChop itl-prop*(*30*) **by** *blast*
have *6*: $\vdash init\ w \wedge_i\ more \equiv_i ((init\ w \wedge_i empty);(true_i;skip))$
using *4 5* **by** *auto*
have *7*: $\vdash\ (true_i;(init\ w \wedge_i\ more)) \equiv_i (true_i;((init\ w \wedge_i empty);(true_i;skip))\ )$
using *6 RightChopEqvChop* **by** *blast*
have *8*: $\vdash (true_i;((init\ w \wedge_i empty);(true_i;skip))\ ) \equiv_i$
$(((true_i;(init\ w \wedge_i empty));(true_i;skip))\ )$
using *ChopAssoc* **by** *blast*
have *9*: $\vdash (((true_i;(init\ w \wedge_i empty));(true_i;skip))\ ) \equiv_i$
$((((true_i;(init\ w \wedge_i empty));true_i);skip)\ )$
using *ChopAssoc* **by** *blast*
have *10*: $\vdash (true_i;(init\ w \wedge_i\ more)) \equiv_i$
$((((true_i;(init\ w \wedge_i empty));true_i);skip)\ )$
using *7 8 9* **by** *auto*
hence *11*: $\vdash \neg_i(true_i;(init\ w \wedge_i\ more)) \equiv_i$
$\neg_i((((true_i;(init\ w \wedge_i empty));true_i);skip)\ )$
by *auto*
have *12*: $\vdash \neg_i((((true_i;(init\ w \wedge_i empty));true_i);skip)\ ) \equiv_i$
$empty \vee_i (\neg_i((true_i;(init\ w \wedge_i empty));true_i);skip)$
using *NotChopNotSkip* **by** *blast*
have *13*: $\vdash (\neg_i((true_i;(init\ w \wedge_i empty));true_i)) \equiv_i bi(\Box\ \neg_i(init\ w \wedge_i empty))$
using *BiBoxNotEqvNotTrueChopChopTrue itl-prop*(*30*) **by** *blast*

hence *14*: $\vdash (\neg_i((true_i;(init\ w \wedge_i empty));true_i));skip \equiv_i$
$(bi(\Box\ \neg_i(init\ w \wedge_i empty)));skip$
using *RightChopEqvChop* **by** *auto*
hence *15*: $\vdash empty \vee_i (\neg_i((true_i;(init\ w \wedge_i empty));true_i));skip \equiv_i$
$empty \vee_i (bi(\Box\ \neg_i(init\ w \wedge_i empty)));skip$
by *auto*
have *16*: $\vdash \neg_i((((true_i;(init\ w \wedge_i empty));true_i);skip)\ ) \equiv_i$
$empty \vee_i (bi(\Box\ \neg_i(init\ w \wedge_i empty));skip)$
using *12 15* **by** *auto*
have *17*: $\vdash empty \vee_i (bi(\Box\ \neg_i(init\ w \wedge_i empty));skip) \equiv_i$
$empty \vee_i (bi(\Box\ (\neg_i(init\ w) \vee_i \neg_i empty));skip)$
by *auto*
have *18*: $\vdash \Box\ (\neg_i(init\ w) \vee_i \neg_i empty) \equiv_i \Box\ (\neg_i empty \vee_i \neg_i(init\ w)\ \ )$
by *auto*
have *19*: $\vdash \Box\ (\neg_i empty \vee_i \neg_i(init\ w)\ \ ) \equiv_i \Box(empty \supset_i \neg_i(init\ w)\ )$
by *auto*
have *20*: $\vdash \Box(empty \supset_i \neg_i(init\ w)\ ) \equiv_i fin\ (\neg_i(init\ w)\ )$
by (*simp add*: *fin-d-def*)
have *21*: $\vdash fin\ (\neg_i(init\ w)\ ) \equiv_i \neg_i(fin\ (init\ w))$
using *FinEqvFin FinNotStateEqvNotFinState Initprop*(*2*) *prop03* **by** *blast*
have *22*: $\vdash bi(\Box\ (\neg_i(init\ w) \vee_i \neg_i empty)) \equiv_i bi\ (\neg_i(fin\ (init\ w)))$
using *18 19 20 21 BiEqvBi* **using** *prop03* **by** *blast*
hence *23*: $\vdash (bi(\Box\ (\neg_i(init\ w) \vee_i \neg_i empty)));skip \equiv_i (bi\ (\neg_i(fin\ (init\ w))));skip$
using *RightChopEqvChop* **by** *auto*
hence *24*: $\vdash empty \vee_i (bi(\Box\ (\neg_i(init\ w) \vee_i \neg_i empty)));skip \equiv_i$
$empty \vee_i (bi\ (\neg_i(fin\ (init\ w))));skip$
by *auto*
hence *25*: $\vdash empty \vee_i (bi\ (\neg_i(fin\ (init\ w))));skip \equiv_i bs(\neg_i(fin\ (init\ w)))$
by (*simp add*:*bs-d-def*)
from *1 2 11 16 17 24 25* **show** *?thesis* **by** *auto*
qed

6.4.4 First occurrence

lemma *FstWithAndImp*:
$\vdash \rhd f \wedge_i g \supset_i \rhd\ (\ f \wedge_i g)$
proof –
have *1*: $\vdash \rhd f \wedge_i g \equiv_i f \wedge_i (bs\ \neg_i f) \wedge_i g$
by (*simp add*: *first-d-def*)
have *2*: $\vdash f \wedge_i (bs\ \neg_i f) \wedge_i g \equiv_i f \wedge_i \neg_i(ds\ f) \wedge_i g$
using *NotDsEqvBsNot* **using** *itl-prop*(*30*) *prop05 prop06* **by** *blast*
have *3*: $\vdash \neg_i(ds\ f) \supset_i \neg_i(ds(f \wedge_i g))$
using *DsAndImpElimL* **using** *prop27* **by** *blast*
hence *4*: $\vdash f \wedge_i \neg_i(ds\ f) \wedge_i g \supset_i f \wedge_i g \wedge_i \neg_i(ds(f \wedge_i g))$
by *auto*
have *5*: $\vdash f \wedge_i g \wedge_i \neg_i(ds(f \wedge_i g)) \equiv_i f \wedge_i g \wedge_i (bs\ \neg_i(f \wedge_i g))$
using *NotDsEqvBsNot* **using** *prop05* **by** *blast*
have *6*: $\vdash f \wedge_i g \wedge_i (bs\ \neg_i(f \wedge_i g)) \equiv_i \rhd(f \wedge_i g)$
by (*simp add*: *first-d-def*)
from *1 2 4 5 6* **show** *?thesis* **by** *auto*

qed

lemma *FstWithOrEqv*:
$\vdash \triangleright(f \vee_i g) \equiv_i (\triangleright f \wedge_i \mathit{bs}\ \neg_i g) \vee_i (\triangleright g \wedge_i \mathit{bs}\ \neg_i f)$
proof −
have *1*: $\vdash \triangleright(f \vee_i g) \equiv_i (f \vee_i g) \wedge_i \mathit{bs}\ \neg_i(f \vee_i g)$
by (*simp add*: *first-d-def*)
have *2*: $\vdash\ \neg_i(f \vee_i g) \equiv_i\ (\neg_i f \wedge_i \neg_i g)$
by *auto*
hence *3*: $\vdash \mathit{bs}\ \neg_i(f \vee_i g) \equiv_i\ \mathit{bs}\ (\neg_i f \wedge_i \neg_i g)$
using *BsEqvRule* **by** *blast*
have *4*: $\vdash\ \mathit{bs}\ (\neg_i f \wedge_i \neg_i g) \equiv_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g$
using *BsAndEqv itl-prop*(*30*) **by** *blast*
have *5*: $\vdash (f \vee_i g) \wedge_i \mathit{bs}\ \neg_i(f \vee_i g) \equiv_i (f \vee_i g) \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g$
using *3 4* **by** *auto*
have *6*: $\vdash (f \vee_i g) \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g \equiv_i$
$(f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g) \vee_i (g \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g)$
by *auto*
have *7*: $\vdash (f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g) \equiv_i \triangleright f \wedge_i\ \mathit{bs}\ \neg_i\ g$
by (*simp add*: *first-d-def*)
have *8*: $\vdash (g \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g) \equiv_i (g \wedge_i \mathit{bs}\ \neg_i\ g \wedge_i \mathit{bs}\ \neg_i\ f)$
by *auto*
have *9*: $\vdash (g \wedge_i \mathit{bs}\ \neg_i\ g \wedge_i \mathit{bs}\ \neg_i\ f) \equiv_i \triangleright g \wedge_i\ \mathit{bs}\ \neg_i\ f$
by (*simp add*: *first-d-def*)
have *10*: $\vdash (f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g) \vee_i (g \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{bs}\ \neg_i\ g) \equiv_i$
$(\triangleright f \wedge_i\ \mathit{bs}\ \neg_i\ g) \vee_i (\triangleright g \wedge_i\ \mathit{bs}\ \neg_i\ f)$
using *7 8 9* **by** *auto*
from *1 5 6 10* **show** *?thesis* **by** *auto*
qed

lemma *FstFstAndEqvFstAnd*:
$\vdash \triangleright(\triangleright f \wedge_i g) \equiv_i \triangleright f \wedge_i g$
proof −
have *1*: $\vdash \triangleright f \wedge_i g \equiv_i f \wedge_i (\mathit{bs}\ \neg_i f) \wedge_i g$ **by** (*simp add*: *first-d-def*)
hence *2*: $\vdash \triangleright f \wedge_i g \supset_i (\mathit{bs}\ \neg_i f)$ **by** *auto*
hence *3*: $\vdash \triangleright f \wedge_i g \supset_i \triangleright f \wedge_i g \wedge_i (\mathit{bs}\ \neg_i f)$ **by** *auto*
have *4*: $\vdash \neg_i\ f \supset_i \neg_i\ f \vee_i \neg_i(\mathit{bs}\ \neg_i f) \vee_i \neg_i\ g$ **by** *auto*
hence *5*: $\vdash \mathit{bs}\ (\neg_i\ f) \supset_i \mathit{bs}(\neg_i\ f \vee_i \neg_i(\mathit{bs}\ \neg_i f) \vee_i \neg_i\ g)$ **using** *BsImpBsRule* **by** *blast*
have *6*: $\vdash \neg_i\ f \vee_i \neg_i(\mathit{bs}\ \neg_i f) \vee_i \neg_i\ g \equiv_i \neg_i(f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i g)$ **by** *auto*
hence *7*: $\vdash \mathit{bs}(\neg_i\ f \vee_i \neg_i(\mathit{bs}\ \neg_i f) \vee_i \neg_i\ g) \equiv_i \mathit{bs}(\neg_i(f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i g))$ **using** *BsEqvRule* **by** *blast*
have *8*: $\vdash f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i g \equiv_i \triangleright f \wedge_i g$ **by** (*simp add*: *first-d-def*)
hence *9*: $\vdash \neg_i(f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i g) \equiv_i \neg_i(\triangleright f \wedge_i g)$ **by** *auto*
hence *10*: $\vdash \mathit{bs}\ \neg_i(f \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i g) \equiv_i \mathit{bs}\ \neg_i(\triangleright f \wedge_i g)$ **using** *BsEqvRule* **by** *blast*
have *11*: $\vdash \triangleright f \wedge_i g \supset_i \triangleright f \wedge_i g \wedge_i\ \ \mathit{bs}\ \neg_i(\triangleright f \wedge_i g)$ **using** *3 5 7 10* **by** *auto*
hence *12*: $\vdash \triangleright f \wedge_i g \supset_i \triangleright(\triangleright f \wedge_i g)$ **by** (*simp add*: *first-d-def*)
have *13*: $\vdash \triangleright(\triangleright f \wedge_i g) \equiv_i \triangleright f \wedge_i g \wedge_i\ \ \mathit{bs}\ \neg_i(\triangleright f \wedge_i g)$ **by** (*simp add*: *first-d-def*)
hence *14*: $\vdash \triangleright(\triangleright f \wedge_i g) \supset_i \triangleright f \wedge_i g$ **by** *auto*
from *12 14* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *FstTrue*:
$\vdash \rhd\ true_i \equiv_i empty$
proof −
have *1*: $\vdash \rhd\ true_i \equiv_i true_i \wedge_i bs\ \neg_i\ true_i$ **by** (*simp add*: *first-d-def*)
have *2*: $\vdash bs\ \neg_i\ true_i \equiv_i empty \vee_i (bi\ \neg_i\ true_i);skip$ **by** (*simp add*: *bs-d-def*)
have *3*: $\vdash \neg_i(bi\ \neg_i\ true_i)$ **by** *auto*
hence *4*: $\vdash \neg_i((bi\ \neg_i\ true_i);skip)$ **by** *auto*
have *5*: $\vdash bs\ \neg_i\ true_i \equiv_i empty$ **using** *2 4* **by** *auto*
from *1 5* **show** *?thesis* **by** *auto*
qed

lemma *FstFalse*:
$\vdash \neg_i(\rhd\ false_i)$
proof −
have *1*: $\vdash \rhd\ false_i \equiv_i false_i \wedge_i bs\ true_i$ **by** (*simp add*: *first-d-def*)
from *1* **show** *?thesis* **by** *auto*
qed

lemma *FstChopFalseEqvFalse*:
$\vdash \neg_i(\rhd f\ ;\ false_i)$
by *auto*

lemma *FstEmpty*:
$\vdash \rhd\ empty \equiv_i empty$
proof −
have *1*: $\vdash \rhd\ empty \equiv_i empty \wedge_i bs\ \neg_i\ empty$ **by** (*simp add*: *first-d-def*)
have *2*: $\vdash bs\ \neg_i\ empty \equiv_i empty \vee_i bi\ \neg_i\ empty;skip$ **by** (*simp add*: *bs-d-def*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *FstAndEmptyEqvAndEmpty*:
$\vdash \rhd f \wedge_i empty \equiv_i f \wedge_i empty$
proof −
have *1*: $\vdash \rhd f \wedge_i empty \equiv_i f \wedge_i bs\ \neg_i\ f \wedge_i empty$ **by** (*simp add*: *first-d-def*)
have *2*: $\vdash bs\ \neg_i\ f \equiv_i empty \vee_i bi\ \neg_i f;skip$ **by** (*simp add*: *bs-d-def*)
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *FstEmptyOrEqvEmpty*:
$\vdash \rhd(empty \vee_i f) \equiv_i empty$
proof −
have *1*: $\vdash \rhd(empty \vee_i f) \equiv_i (\rhd empty \wedge_i bs\ \neg_i\ f) \vee_i (\rhd f \wedge_i bs\ \neg_i\ empty)$ **using** *FstWithOrEqv* **by** *blast*
have *2*: $\vdash\ \neg_i\ empty \equiv_i more$ **by** *auto*
hence *3*: $\vdash bs\ \neg_i\ empty \equiv_i bs\ more$ **using** *BsEqvRule* **by** *blast*
have *4*: $\vdash bs\ more \equiv_i empty$ **using** *BsMoreEqvEmpty* **by** *blast*
have *5*: $\vdash \rhd f \wedge_i bs\ \neg_i\ empty \equiv_i (\rhd f \wedge_i empty)$ **using** *3 4* **by** *auto*
have *6*: $\vdash \rhd empty \equiv_i empty$ **using** *FstEmpty* **by** *blast*
hence *7*: $\vdash (\rhd empty \wedge_i bs\ \neg_i\ f) \equiv_i (empty \wedge_i bs\ \neg_i\ f)$ **by** *auto*
have *8*: $\vdash empty \wedge_i\ bs\ \neg_i\ f \equiv_i empty \wedge_i(empty \vee_i bi\ \neg_i\ f;skip)$ **by** (*simp add*:*bs-d-def*)
have *9*: $\vdash empty \wedge_i(empty \vee_i bi\ \neg_i\ f;skip) \equiv_i empty$ **by** *auto*

have *10*: $\vdash \mathit{empty} \wedge_i \ \mathit{bs} \neg_i f \equiv_i \mathit{empty}$ **using** *8 9* **by** *auto*
have *11*: $\vdash (\triangleright \mathit{empty} \wedge_i \mathit{bs} \neg_i f) \vee_i (\triangleright f \wedge_i \mathit{bs} \neg_i \mathit{empty}) \equiv_i$
$\mathit{empty} \vee_i (\triangleright f \wedge_i \mathit{empty})$ **using** *7 10 5* **by** *auto*
have *12*: $\vdash \mathit{empty} \vee_i (\triangleright f \wedge_i \mathit{empty}) \equiv_i \mathit{empty}$ **by** *auto*
from *1 11 12* **show** *?thesis* **by** *auto*
qed

lemma *FstChopEmptyEqvFstChopFstEmpty*:
$\vdash \triangleright f;g \wedge_i \mathit{empty} \equiv_i \triangleright f;\triangleright g \wedge_i \mathit{empty}$
proof –
have *1*: $\vdash \triangleright f;g \wedge_i \mathit{empty} \equiv_i \triangleright f \wedge_i g \wedge_i \mathit{empty}$ **using** *ChopEmptyAndEmpty* **by** *blast*
have *2*: $\vdash g \wedge_i \mathit{empty} \equiv_i \triangleright g \wedge_i \mathit{empty}$ **using** *FstAndEmptyEqvAndEmpty* **using** *itl-prop(30)* **by** *blast*
hence *3*: $\vdash \triangleright f \wedge_i g \wedge_i \mathit{empty} \equiv_i \triangleright f \wedge_i \triangleright g \wedge_i \mathit{empty}$ **by** *auto*
have *4*: $\vdash \triangleright f;\triangleright g \wedge_i \mathit{empty} \equiv_i \triangleright f \wedge_i \triangleright g \wedge_i \mathit{empty}$ **using** *ChopEmptyAndEmpty* **by** *blast*
from *1 3 4* **show** *?thesis* **by** *auto*
qed

lemma *FstMoreEqvSkip*:
$\vdash \triangleright \ \mathit{more} \equiv_i \mathit{skip}$
proof –
have *1*: $\vdash \triangleright \ \mathit{more} \equiv_i \mathit{more} \wedge_i \mathit{bs} \neg_i \mathit{more}$ **by** (*simp add*: *first-d-def*)
have *2*: $\vdash \mathit{more} \wedge_i \mathit{bs} \neg_i \mathit{more} \equiv_i \mathit{more} \wedge_i (\mathit{empty} \vee_i \mathit{bi} \neg_i \mathit{more};\mathit{skip})$ **by** (*simp add*:*bs-d-def*)
have *3*: $\vdash \mathit{more} \wedge_i (\mathit{empty} \vee_i \mathit{bi} \neg_i \mathit{more};\mathit{skip}) \equiv_i \mathit{more} \wedge_i \mathit{bi} \neg_i \mathit{more};\mathit{skip}$ **by** *auto*
have *4*: $\vdash \mathit{more} \wedge_i ((\mathit{bi} \neg_i \mathit{more});\mathit{skip}) \equiv_i ((\mathit{bi} \neg_i \mathit{more});\mathit{skip})$ **using** *ChopSkipImpMore* **by** *auto*
have *5*: $\vdash ((\mathit{bi} \neg_i \mathit{more});\mathit{skip}) \equiv_i \mathit{bi}\ \mathit{empty};\mathit{skip}$ **by** *auto*
have *6*: $\vdash \mathit{bi}\ \mathit{empty} \equiv_i \mathit{empty}$ **using** *BiEmptyEqvEmpty* **by** *auto*
hence *7*: $\vdash \mathit{bi}\ \mathit{empty};\mathit{skip} \equiv_i \mathit{empty};\mathit{skip}$ **using** *LeftChopEqvChop* **by** *blast*
have *8*: $\vdash \mathit{empty};\mathit{skip} \equiv_i \mathit{skip}$ **using** *EmptyChop* **by** *blast*
from *1 2 3 4 5 7 8* **show** *?thesis* **by** (*metis prop03*)
qed

lemma *FstEqvBsNotAndDi*:
$\vdash \triangleright f \equiv_i \mathit{bs} \neg_i f \wedge_i \mathit{di}\ f$
proof –
have *1*: $\vdash \mathit{bs} \neg_i f \equiv_i \neg_i(\mathit{ds}\ f)$ **by** (*simp add*: *ds-d-def*)
hence *2*: $\vdash \mathit{bs} \neg_i f \wedge_i \mathit{di}\ f \equiv_i \neg_i(\mathit{ds}\ f) \wedge_i \mathit{di}\ f$ **by** *auto*
have *3*: $\vdash \mathit{di}\ f \equiv_i (\mathit{ds}\ f \vee_i f)$ **using** *OrDsEqvDi* **by** *auto*
hence *4* : $\vdash \neg_i(\mathit{ds}\ f) \wedge_i \mathit{di}\ f \equiv_i \neg_i(\mathit{ds}\ f) \wedge_i (\mathit{ds}\ f \vee_i f)$ **by** *auto*
have *5* : $\vdash \neg_i(\mathit{ds}\ f) \wedge_i (\mathit{ds}\ f \vee_i f) \equiv_i \neg_i(\mathit{ds}\ f) \wedge_i f$ **by** *auto*
have *6* : $\vdash \neg_i(\mathit{ds}\ f) \wedge_i f \equiv_i f \wedge_i \ \mathit{bs} \neg_i f$ **using** *1* **by** *auto*
from *2 4 5 6* **show** *?thesis* **by** (*simp add*: *first-d-def*)
qed

lemma *FstOrDiEqvDi*:
$\vdash \triangleright f \vee_i \mathit{di}\ f \equiv_i \mathit{di}\ f$
proof –
have *1*: $\vdash \triangleright f \vee_i \mathit{di}\ f \equiv_i (f \wedge_i \mathit{bs} \neg_i f) \vee_i \mathit{di}\ f$ **by** (*simp add*: *first-d-def*)
have *2*: $\vdash (f \wedge_i \mathit{bs} \neg_i f) \vee_i \mathit{di}\ f \equiv_i (f \vee_i \mathit{di}\ f) \wedge_i (\mathit{bs} \neg_i f \vee_i \mathit{di}\ f)$ **by** *auto*
have *3*: $\vdash (f \vee_i \mathit{di}\ f) \equiv_i \mathit{di}\ f$ **by** *auto*
hence *4*: $\vdash (f \vee_i \mathit{di}\ f) \wedge_i (\mathit{bs} \neg_i f \vee_i \mathit{di}\ f) \equiv_i \mathit{di}\ f \wedge_i (\mathit{bs} \neg_i f \vee_i \mathit{di}\ f)$ **by** *auto*

have *5*: $\vdash \mathit{di}\ f \wedge_i (\mathit{bs}\ \neg_i\ f \vee_i \mathit{di}\ f) \equiv_i \mathit{di}\ f$ **by** *auto*
from *1 2 4 5* **show** *?thesis* **by** *auto*
qed

lemma *FstAndDiEqvFst*:
$\vdash \rhd f \wedge_i \mathit{di}\ f \equiv_i \rhd f$
proof –
have *1*: $\vdash \rhd f \wedge_i \mathit{di}\ f \equiv_i f \wedge_i \mathit{bs}\ \neg_i f \wedge_i \mathit{di}\ f$ **by** (*simp add*: *first-d-def*)
have *2*: $\vdash f \wedge_i \mathit{di}\ f \equiv_i f$ **by** *auto*
hence *3*: $\vdash f \wedge_i \mathit{bs}\ \neg_i f \wedge_i \mathit{di}\ f \equiv_i f \wedge_i \mathit{bs}\ \neg_i f$ **by** *auto*
from *1 3* **show** *?thesis* **by** (*simp add*: *first-d-def*)
qed

lemma *DiEqvDiFst*:
$\vdash \mathit{di}\ f \equiv_i \mathit{di}\ (\rhd\ f)$
proof –
have *1*: $\vdash\ \mathit{di}\ (\rhd\ f) \equiv_i \mathit{di}\ (f \wedge_i \mathit{bs}\ \neg_i f)$
by (*simp add*: *first-d-def*)
have *2*: $\vdash \mathit{di}\ (f \wedge_i \mathit{bs}\ \neg_i f) \supset_i \mathit{di}\ f \wedge_i \mathit{di}\ (\mathit{bs}\ \neg_i\ f)$
using *DiAndImpAnd* **by** *auto*
hence *3*: $\vdash \mathit{di}\ (f \wedge_i \mathit{bs}\ \neg_i f) \supset_i \mathit{di}\ f$
by *auto*
have *4*: $\vdash \mathit{di}\ (\rhd\ f) \supset_i \mathit{di}\ f$ **using** *1 3*
by *auto*
have *5*: $\vdash \mathit{di}\ f \wedge_i \mathit{empty} \equiv_i f \wedge_i \mathit{empty}$
using *DiAndEmptyEqvAndEmpty* **by** *blast*
have *6*: $\vdash \rhd\ f \wedge_i \mathit{empty} \equiv_i f \wedge_i \mathit{empty}$
using *FstAndEmptyEqvAndEmpty* **by** *auto*
have *7*: $\vdash \mathit{di}\ f \wedge_i \mathit{empty} \supset_i \rhd f$
using *5 6* **by** *auto*
have *8*: $\vdash \rhd f \supset_i \mathit{di}\ (\rhd\ f)$
using *DiIntro* **by** *auto*
have *9*: $\vdash\ \mathit{di}\ f \wedge_i \mathit{empty} \supset_i \mathit{di}\ (\rhd\ f)$
using *7 8* **using** *prop02* **by** *blast*
hence *10*: $\vdash \mathit{empty} \supset_i (\mathit{di}\ f \supset_i \mathit{di}\ (\rhd\ f))$
by *auto*
have *11*: $\vdash \mathit{prev}\ (\ \mathit{di}\ f \supset_i \mathit{di}\ (\rhd\ f)) \supset_i \mathit{more}$
by *auto*
have *12*: $\vdash \mathit{more} \supset_i (\ \mathit{prev}\ (\ \mathit{di}\ f \supset_i \mathit{di}\ (\rhd\ f)) \equiv_i (\mathit{prev}(\mathit{di}\ f) \supset_i \mathit{prev}(\mathit{di}\ (\rhd f))))$
using *MoreImpImpPrevEqv* **by** *auto*
have *13*: $\vdash \mathit{more} \wedge_i \mathit{prev}\ (\ \mathit{di}\ f \supset_i \mathit{di}\ (\rhd\ f)) \equiv_i \mathit{more} \wedge_i (\mathit{prev}(\mathit{di}\ f) \supset_i \mathit{prev}(\mathit{di}\ (\rhd f)))$
using *12 prop31* **by** *auto*
have *14*: $\vdash \mathit{prev}\ (\ \mathit{di}\ f \supset_i \mathit{di}\ (\rhd\ f)) \equiv_i \mathit{more} \wedge_i (\mathit{prev}(\mathit{di}\ f) \supset_i \mathit{prev}(\mathit{di}\ (\rhd f)))$
using *11* **by** *auto*
have *15*: $\vdash \mathit{di}\ f \equiv_i f \vee_i \mathit{ds}\ f$
using *OrDsEqvDi* **by** *auto*
have *16*: $\vdash \mathit{di}\ f \equiv_i \mathit{di}\ f \wedge_i (\mathit{bs}\ \neg_i\ f \vee_i \neg_i(\mathit{bs}\ \neg_i\ f))$
by *auto*
have *17*: $\vdash \mathit{di}\ f \wedge_i (\mathit{bs}\ \neg_i\ f \vee_i \neg_i(\mathit{bs}\ \neg_i\ f)) \equiv_i (\mathit{di}\ f \wedge_i \mathit{bs}\ \neg_i\ f) \vee_i (\mathit{di}\ f \wedge_i \neg_i(\mathit{bs}\ \neg_i\ f))$
by *auto*

have *18*: $\vdash (di\ f \wedge_i bs\ \neg_i\ f) \equiv_i (f \vee_i ds\ f) \wedge_i bs\ \neg_i\ f$

using *15* **by** *auto*

have *19*: $\vdash (f \vee_i ds\ f) \wedge_i bs\ \neg_i\ f \equiv_i (f \wedge_i bs\ \neg_i\ f) \vee_i (ds\ f \wedge_i bs\ \neg_i\ f)$

by *auto*

have *20*: $\vdash \neg_i(ds\ f \wedge_i bs\ \neg_i\ f)$

by (*simp add*: *ds-d-def*)

have *21*: $\vdash (f \wedge_i bs\ \neg_i\ f) \vee_i (ds\ f \wedge_i bs\ \neg_i\ f) \equiv_i (f \wedge_i bs\ \neg_i\ f)$

using *20* **by** *auto*

have *22*: $\vdash (di\ f \wedge_i bs\ \neg_i\ f) \equiv_i (f \wedge_i bs\ \neg_i\ f)$

using *18 19 21* **by** *auto*

have *23*: $\vdash (f \wedge_i bs\ \neg_i\ f) \equiv_i \triangleright f$

by (*simp add*: *first-d-def*)

have *24*: $\vdash (\triangleright f) \supset_i di\ (\triangleright f)$

using *DiIntro* **by** *auto*

have *25*: $\vdash (f \wedge_i bs\ \neg_i\ f) \supset_i di\ (\triangleright f)$

using *23 24* **by** *auto*

have *26*: $\vdash (di\ f \wedge_i bs\ \neg_i\ f) \supset_i di\ (\triangleright f)$

using *25 22* **by** *auto*

hence *27*: $\vdash (di\ f \wedge_i bs\ \neg_i\ f \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright\ f)))) \supset_i di\ (\triangleright f)$

by *auto*

have *28*: $\vdash di\ f \wedge_i \neg_i(bs\ \neg_i\ f) \equiv_i di\ f \wedge_i ds\ f$

by (*simp add*: *ds-d-def*)

hence *29*: $\vdash di\ f \wedge_i \neg_i(bs\ \neg_i\ f) \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright\ f)))\ \equiv_i$
$di\ f \wedge_i ds\ f \wedge_i\ (prev\ (di\ f \supset_i di\ (\triangleright\ f)))$

by *auto*

have *30*: $\vdash\ ds\ f \equiv_i prev(di\ f)$

using *DsDi* **by** (*metis prev-d-def*)

hence *31*: $\vdash di\ f \wedge_i ds\ f \wedge_i\ (prev\ (di\ f \supset_i di\ (\triangleright\ f))) \equiv_i$
$di\ f \wedge_i prev(di\ f) \wedge_i\ (prev\ (di\ f \supset_i di\ (\triangleright\ f)))$

by *auto*

have *32*: $\vdash prev\ (\ di\ f \supset_i di\ (\triangleright\ f)) \supset_i\ (prev(di\ f) \supset_i prev(di\ (\triangleright f)))$

using *14* **by** *auto*

hence *33*: $\vdash di\ f \wedge_i prev(di\ f) \wedge_i prev\ (\ di\ f \supset_i di\ (\triangleright\ f)) \supset_i$
$di\ f \wedge_i prev(di\ f) \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f)))$

by *auto*

have *34*: $\vdash di\ f \wedge_i prev(di\ f) \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f))) \supset_i prev(di\ (\triangleright f))$

by *auto*

have *35*: $\vdash prev(di\ (\triangleright f)) \equiv_i (di\ (\triangleright f));skip$

by (*simp add*: *prev-d-def*)

have *36*: $\vdash (di\ (\triangleright f));skip \supset_i di(di\ (\triangleright f))$

using *ChopImpDi* **by** *auto*

have *37*: $\vdash di(di\ (\triangleright f)) \equiv_i di\ (\triangleright f)$

using *DiEqvDiDi* **using** *itl-prop*(*30*) **by** *blast*

have *38*: $\vdash di\ f \wedge_i prev(di\ f) \wedge_i (prev(di\ f) \supset_i prev(di\ (\triangleright f))) \supset_i di\ (\triangleright f)$

using *37 36 35 34 itl-prop*(*31*) *prop02* **by** *blast*

have *39*: $\vdash di\ f \wedge_i \neg_i(bs\ \neg_i\ f) \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright\ f))) \supset_i di\ (\triangleright f)$

using *29 31 33 38* **by** (*meson itl-prop*(*31*) *prop02*)

hence *40*: $\vdash \neg_i(bs\ \neg_i\ f) \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright\ f))) \supset_i (di\ f \supset_i di\ (\triangleright f))$

using *prop32* **by** *blast*

have *41*: $\vdash bs\ \neg_i\ f \wedge_i (prev\ (di\ f \supset_i di\ (\triangleright\ f))) \supset_i (di\ f \supset_i di\ (\triangleright f))$

using *27 prop32* **by** *blast*

have *42*: $\vdash (\neg_i(\mathit{bs}\ \neg_i\ f) \vee_i \mathit{bs}\ \neg_i\ f) \wedge_i (\mathit{prev}\ (\mathit{di}\ f \supset_i \mathit{di}\ (\rhd\ f))) \supset_i(\mathit{di}\ f \supset_i \mathit{di}\ (\rhd f))$

using *40 41 prop33* **by** *blast*

have *43*: $\vdash (\neg_i(\mathit{bs}\ \neg_i\ f) \vee_i \mathit{bs}\ \neg_i\ f)$

by *auto*

have *44*: $\vdash (\mathit{prev}\ (\mathit{di}\ f \supset_i \mathit{di}\ (\rhd\ f))) \supset_i(\mathit{di}\ f \supset_i \mathit{di}\ (\rhd f))$

using *42 43 prop34* **by** *blast*

have *45*: $\vdash \mathit{di}\ f \supset_i \mathit{di}\ (\rhd f)$

using *10 44 EmptyChopSkipInduct* **by** *blast*

from *4 45* **show** *?thesis* **by** *auto*

qed

lemma *FstDiEqvFst*:

$\vdash \rhd(\mathit{di}\ f) \equiv_i \rhd f$

proof $-$

have *1*: $\vdash \rhd(\mathit{di}\ f) \equiv_i \mathit{di}\ f \wedge_i \mathit{bs}\ \neg_i\ (\mathit{di}\ f)$ **by** (*simp add*: *first-d-def*)

have *2*: $\vdash\ \neg_i\ (\mathit{di}\ f) \equiv_i \mathit{bi}\ \neg_i\ f$ **by** *auto*

hence *3*: $\vdash \mathit{bs}\ \neg_i\ (\mathit{di}\ f) \equiv_i \mathit{bs}\ (\mathit{bi}\ \neg_i\ f)$ **using** *BsEqvRule* **by** *blast*

have *4*: $\vdash \mathit{bs}\ (\mathit{bi}\ \neg_i\ f) \equiv_i \mathit{bs}\ (\ \neg_i\ f)$ **using** *BsEqvBsBi* **using** *itl-prop*(*30*) **by** *blast*

hence *5*: $\vdash \mathit{di}\ f \wedge_i \mathit{bs}\ \neg_i\ (\mathit{di}\ f) \equiv_i \mathit{di}\ f \wedge_i \mathit{bs}\ (\ \neg_i\ f)$ **using** *3* **by** *auto*

have *6* : $\vdash \mathit{di}\ f \equiv_i f \vee_i \mathit{ds}\ f$ **using** *OrDsEqvDi* **using** *itl-prop*(*30*) **by** *blast*

hence *7*: $\vdash \mathit{di}\ f \wedge_i \mathit{bs}\ (\ \neg_i\ f) \equiv_i (f \vee_i \mathit{ds}\ f) \wedge_i \mathit{bs}\ (\ \neg_i\ f)$ **by** *auto*

have *8*: $\vdash (f \vee_i \mathit{ds}\ f) \wedge_i \mathit{bs}\ (\ \neg_i\ f) \equiv_i (f \wedge_i \mathit{bs}\ (\ \neg_i\ f)) \vee_i (\mathit{ds}\ f \wedge_i \mathit{bs}\ (\ \neg_i\ f))$ **by** *auto*

have *9*: $\vdash \neg_i(\mathit{ds}\ f \wedge_i \mathit{bs}\ (\ \neg_i\ f))$ **by** (*simp add*: *ds-d-def*)

have *10*: $\vdash (f \wedge_i \mathit{bs}\ (\ \neg_i\ f)) \equiv_i \rhd f$ **by** (*simp add*: *first-d-def*)

from *1 5 7 8 9 10* **show** *?thesis* **by** *auto*

qed

lemma *DiAndFstOrEqvFstOrDiAnd*:

$\vdash \mathit{di}\ f \wedge_i (\rhd f \vee_i g) \equiv_i \rhd f \vee_i (\mathit{di}\ f \wedge_i g)$

proof $-$

have *1*: $\vdash \mathit{di}\ f \wedge_i (\rhd f \vee_i g) \equiv_i (\rhd f \wedge_i \mathit{di}\ f) \vee_i (\mathit{di}\ f \wedge_i g)$ **by** *auto*

have *2*: $\vdash (\rhd f \wedge_i \mathit{di}\ f) \equiv_i \rhd f$ **using** *FstAndDiEqvFst* **by** *blast*

from *1 2* **show** *?thesis* **by** *auto*

qed

lemma *DiOrFstAndEqvDi*:

$\vdash \mathit{di}\ f \vee_i (\rhd f \wedge_i g) \equiv_i \mathit{di}\ f$

proof $-$

have *1*: $\vdash \mathit{di}\ f \vee_i (\rhd f \wedge_i g) \equiv_i (\rhd f \vee_i \mathit{di}\ f) \wedge_i (\mathit{di}\ f \vee_i g)$ **by** *auto*

have *2*: $\vdash (\rhd f \vee_i \mathit{di}\ f) \equiv_i \mathit{di}\ f$ **using** *FstOrDiEqvDi* **by** *blast*

from *1 2* **show** *?thesis* **by** *auto*

qed

lemma *FstDiAndDiEqv*:

$\vdash \rhd(\mathit{di}\ f \wedge_i \mathit{di}\ g) \equiv_i (\rhd f \wedge_i \mathit{di}\ g) \vee_i (\rhd g \wedge_i \mathit{di}\ f)$

proof $-$

have *1*: $\vdash \rhd(\mathit{di}\ f \wedge_i \mathit{di}\ g) \equiv_i \mathit{di}\ f \wedge_i \mathit{di}\ g \wedge_i \mathit{bs}\ \neg_i(\mathit{di}\ f \wedge_i \mathit{di}\ g)$ **by** (*simp add*: *first-d-def*)

have *2*: $\vdash \neg_i(\mathit{di}\ f \wedge_i \mathit{di}\ g) \equiv_i \mathit{bi}\ \neg_i f \vee_i \mathit{bi}\ \neg_i g$ **by** *auto*

hence *3*: $\vdash \mathit{bs}\ \neg_i(\mathit{di}\ f \wedge_i \mathit{di}\ g) \equiv_i \mathit{bs}(\mathit{bi}\ \neg_i f \vee_i \mathit{bi}\ \neg_i g)$ **using** *BsEqvRule* **by** *blast*

hence *4*: $\vdash\ di\ f \wedge_i di\ g \wedge_i bs\ \neg_i(di\ f \wedge_i di\ g) \equiv_i$
$di\ f \wedge_i di\ g \wedge_i bs(bi\ \neg_i f \vee_i bi\ \neg_i g)$ **by** *auto*
have *5*: $\vdash bs\ \neg_i f \vee_i bs\ \neg_i g \equiv_i bs(bi\ \neg_i f \vee_i bi\ \neg_i g)$ **using** *BsOrBsEqvBsBiOrBi* **by** *blast*
hence *6*: $\vdash di\ f \wedge_i di\ g \wedge_i bs(bi\ \neg_i f \vee_i bi\ \neg_i g) \equiv_i$
$di\ f \wedge_i di\ g \wedge_i (bs\ \neg_i f \vee_i bs\ \neg_i g)$ **by** *auto*
have *7*: $\vdash di\ f \wedge_i di\ g \wedge_i (bs\ \neg_i f \vee_i bs\ \neg_i g) \equiv_i$
$(\ bs\ \neg_i f \wedge_i di\ f \wedge_i di\ g) \vee_i (di\ f \wedge_i bs\ \neg_i g \wedge_i di\ g\ )$ **by** *auto*
have *8*: $\vdash \rhd f \equiv_i\ bs\ \neg_i f\ \wedge_i di\ f$ **using** *FstEqvBsNotAndDi* **by** *blast*
hence *9*: $\vdash bs\ \neg_i f \wedge_i di\ f \wedge_i di\ g \equiv_i \rhd f \wedge_i di\ g$ **by** *auto*
have *10*: $\vdash \rhd g \equiv_i\ bs\ \neg_i g\ \wedge_i di\ g$ **using** *FstEqvBsNotAndDi* **by** *blast*
hence *11*: $\vdash di\ f \wedge_i bs\ \neg_i g \wedge_i di\ g \equiv_i di\ f \wedge_i \rhd g$ **by** *auto*
have *12*: $\vdash di\ f \wedge_i di\ g \wedge_i (bs\ \neg_i f \vee_i bs\ \neg_i g) \equiv_i$
$(\rhd f \wedge_i di\ g) \vee_i (di\ f \wedge_i \rhd g)$ **using** *7 9 11* **by** *auto*
from *1 4 6 12* **show** *?thesis* **by** *auto*
qed

lemma *BiNotFstEqvBiNot*:
$\vdash bi\ \neg_i\ (\rhd f) \equiv_i bi\ \neg_i\ f$
proof –
have *1*: $\vdash di\ f \equiv_i di\ (\rhd f)$ **using** *DiEqvDiFst* **by** *blast*
hence *2*: $\vdash \neg_i(di\ f) \equiv_i \neg_i(\ di\ (\rhd f))$ **by** *auto*
from *1 2* **show** *?thesis* **using** *NotDiEqvBiNot itl-prop*(*30*) *prop03* **by** *blast*
qed

lemma *BsNotFstEqvBsNot*:
$\vdash bs\ \neg_i\ (\rhd f) \equiv_i bs\ \neg_i\ f$
proof –
have *1*: $\vdash bs\ \neg_i\ (\rhd f) \equiv_i empty \vee_i bi\ \neg_i\ (\rhd f);skip$ **by** (*simp add*: *bs-d-def*)
have *2*: $\vdash bi\ \neg_i\ (\rhd f) \equiv_i bi\ \neg_i\ f$ **using** *BiNotFstEqvBiNot* **by** *blast*
hence *3*: $\vdash bi\ \neg_i\ (\rhd f);skip \equiv_i bi\ \neg_i\ f;skip$ **using** *LeftChopEqvChop* **by** *blast*
hence *4*: $\vdash empty \vee_i bi\ \neg_i\ (\rhd f);skip \equiv_i empty \vee_i bi\ \neg_i\ f;skip$ **by** *auto*
from *1 4* **show** *?thesis* **by** (*simp add*: *bs-d-def*)
qed

lemma *FstState*:
$\vdash \rhd\ (init\ w) \equiv_i empty \wedge_i init\ w$
proof –
have *1*: $\vdash \rhd\ (init\ w) \equiv_i init\ w \wedge_i bs\ \neg_i(init\ w)$ **by** (*simp add*: *first-d-def*)
hence *2*: $\vdash \rhd\ (init\ w) \supset_i init\ w$ **by** *auto*
have *3*: $\vdash init\ w \supset_i bs\ (init\ w)$ **using** *StateImpBs* **by** *auto*
have *4*: $\vdash \rhd\ (init\ w) \supset_i bs\ (init\ w)$ **using** *2 3* **by** *auto*
have *5*: $\vdash \rhd\ (init\ w) \supset_i bs\ \neg_i(init\ w)$ **using** *1* **by** *auto*
have *6*: $\vdash \rhd\ (init\ w) \supset_i bs\ (init\ w) \wedge_i bs\ \neg_i(init\ w)$ **using** *4 5* **by** *auto*
have *7*: $\vdash bs\ (init\ w) \wedge_i bs\ \neg_i(init\ w) \equiv_i bs((init\ w) \wedge_i \neg_i(init\ w))$ **using** *BsAndEqv* **by** *blast*
have *8*: $\vdash (init\ w) \wedge_i \neg_i(init\ w) \equiv_i\ false_i$ **by** *auto*
hence *9*: $\vdash bs((init\ w) \wedge_i \neg_i(init\ w)) \equiv_i\ bs\ false_i$ **using** *BsEqvRule* **by** *blast*
have *10*: $\vdash bs\ false_i \equiv_i empty$ **using** *BsFalseEqvEmpty* **by** *auto*
have *11*: $\vdash \rhd\ (init\ w) \supset_i empty$ **using** *10 9 7 6* **by** *auto*
have *12*: $\vdash \rhd\ (init\ w) \supset_i empty \wedge_i init\ w$ **using** *11 2* **by** *auto*
have *13*: $\vdash empty \wedge_i init\ w \supset_i empty$ **by** *auto*

hence *14*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \supset_i \mathit{empty} \vee_i \mathit{bi}\ \neg_i(\mathit{init}\ w);\mathit{skip}$ **by** *auto*

hence *15*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \supset_i \mathit{bs}\ \neg_i(\mathit{init}\ w)$ **by** (*simp add*: *bs-d-def*)

have *16*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \supset_i \mathit{init}\ w$ **by** *auto*

have *17*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \supset_i \mathit{init}\ w \wedge_i \mathit{bs}\ \neg_i(\mathit{init}\ w)$ **using** *16 15* **by** *auto*

hence *18*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \supset_i \rhd(\mathit{init}\ w)$ **by** (*simp add*: *first-d-def*)

from *12 18* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*

qed

lemma *FstStateAndBsNotEmpty*:

$\vdash \rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \neg_i\ \mathit{empty} \equiv_i \rhd\ (\mathit{init}\ w)$

proof –

have *1*: $\vdash \rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \neg_i\ \mathit{empty} \equiv_i \rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \mathit{more}$

using *BsEqvRule NotEmptyEqvMore prop05* **by** *blast*

have *2*: $\vdash \rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \mathit{more} \equiv_i \rhd\ (\mathit{init}\ w) \wedge_i \mathit{empty}$

using *BsMoreEqvEmpty prop05* **by** *blast*

have *3*: $\vdash \rhd\ (\mathit{init}\ w) \equiv_i \mathit{empty} \wedge_i (\mathit{init}\ w)$

using *FstState* **by** *blast*

hence *4*: $\vdash \rhd\ (\mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i \mathit{empty} \wedge_i (\mathit{init}\ w) \wedge_i \mathit{empty}$

by *auto*

have *5*: $\vdash \mathit{empty} \wedge_i (\mathit{init}\ w) \wedge_i \mathit{empty} \equiv_i \mathit{empty} \wedge_i (\mathit{init}\ w)$

by *auto*

have *6*: $\vdash \mathit{empty} \wedge_i (\mathit{init}\ w) \equiv_i \rhd(\mathit{init}\ w)$

using *FstState* **using** *itl-prop*(*30*) **by** *blast*

from *1 2 4 5 6* **show** *?thesis* **by** *auto*

qed

lemma *FstStateImpFstStateOr*:

$\vdash \rhd(\mathit{init}\ w) \supset_i \rhd(\mathit{init}\ w \vee_i f)$

proof –

have *1*: $\vdash \rhd(\mathit{init}\ w) \equiv_i \mathit{empty} \wedge_i \mathit{init}\ w$

using *FstState* **by** *blast*

have *2*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \equiv_i \mathit{empty} \wedge_i (\mathit{empty} \vee_i \mathit{bi}\ \neg_i\ f;\mathit{skip}) \wedge_i \mathit{init}\ w$

by *auto*

have *3*: $\vdash \mathit{empty} \wedge_i (\mathit{empty} \vee_i \mathit{bi}\ \neg_i\ f;\mathit{skip}) \wedge_i \mathit{init}\ w \equiv_i$

$\mathit{empty} \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{init}\ w$

by (*simp add*: *bs-d-def*)

have *4*: $\vdash \mathit{empty} \wedge_i \mathit{bs}\ \neg_i\ f \wedge_i \mathit{init}\ w \equiv_i \mathit{empty} \wedge_i \mathit{init}\ w \wedge_i \mathit{bs}\ \neg_i\ f$

by *auto*

have *5*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \equiv_i \rhd\ (\mathit{init}\ w)$

using *FstState itl-prop*(*30*) **by** *blast*

hence *6*: $\vdash \mathit{empty} \wedge_i \mathit{init}\ w \wedge_i \mathit{bs}\ \neg_i\ f \equiv_i \rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \neg_i\ f$

by *auto*

have *7*: $\vdash \rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \neg_i\ f \supset_i (\rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \neg_i\ f) \vee_i (\rhd f \wedge_i \mathit{bs}\ \neg_i(\mathit{init}\ w))$

by *auto*

have *8*: $\vdash \rhd(\mathit{init}\ w \vee_i f) \equiv_i (\rhd\ (\mathit{init}\ w) \wedge_i \mathit{bs}\ \neg_i\ f) \vee_i (\rhd f \wedge_i \mathit{bs}\ \neg_i(\mathit{init}\ w))$

using *FstWithOrEqv* **by** *blast*

from *1 2 3 4 5 6 7 8* **show** *?thesis* **by** *auto*

qed

lemma *FstLenSame*:

```
  (∀ σ. (σ ⊨ di (▷f ∧_i len(i)) ∧_i di (▷f ∧_i len(j))) ⟶ (i=j))
using FstLenSamesem DiLenFstsem by (metis and-defs)

lemma FstAndLenSame:
  (∀ σ. (σ ⊨ di (▷f ∧_i g1 ∧_i len(i)) ∧_i di (▷f ∧_i g2 ∧_i len(j))) ⟶ (i=j))
using  DiLenFstAndsem by (metis and-defs linorder-neqE-nat not-defs)

lemma FstLenSameChop:
  (∀ σ. (σ ⊨ (▷f ∧_i g1 ∧_i len(i));h1 ∧_i (▷f ∧_i g2 ∧_i len(j));h2) ⟶ (i=j))
proof
  fix σ
  show (σ ⊨ (▷f ∧_i g1 ∧_i len(i));h1 ∧_i (▷f ∧_i g2 ∧_i len(j));h2) ⟶ (i=j)
  proof
  assume 0: (σ ⊨ (▷f ∧_i g1 ∧_i len(i));h1 ∧_i (▷f ∧_i g2 ∧_i len(j));h2)
   have 1: (σ ⊨ (▷f ∧_i g1 ∧_i len(i));h1)  using 0 by auto
   have 2: (σ ⊨ (▷f ∧_i g1 ∧_i len(i));h1) ⟶
          (σ ⊨ (▷f ∧_i g1 ∧_i len(i));true_i)  by auto
   have 3: (σ ⊨ di(▷f ∧_i g1 ∧_i len(i)))  using 1 2 by auto
   have 4: (σ ⊨ (▷f ∧_i g2 ∧_i len(j));h2)  using 0 by auto
   have 5: (σ ⊨ (▷f ∧_i g2 ∧_i len(j));h2) ⟶
          (σ ⊨ (▷f ∧_i g2 ∧_i len(j));true_i)  by auto
   have 6: (σ ⊨ di(▷f ∧_i g2 ∧_i len(j)))  using 4 5 by auto
   have 7: (σ ⊨ di(▷f ∧_i g1 ∧_i len(i)) ∧_i di(▷f ∧_i g2 ∧_i len(j)))  using 3 6 by auto
   thus (i=j) using FstAndLenSame by blast
   qed
qed

lemma DiImpExistsOneDiLenAndFst:
 (∀σ. (σ⊨ di f) ⟶ (∃! k. (σ ⊨ di(  ▷f ∧_i len(k)))))
proof
 fix σ
 show (σ⊨ di f) ⟶ (∃! k. (σ ⊨ di(  ▷f ∧_i len(k))))
 proof
  assume 0: (σ⊨ di f)
  have 1: (σ ⊨ di (▷ f))  using 0 DiEqvDiFst valid-def by auto
  have 2: (σ ⊨ ▷ f) = ( (σ⊨▷ f) ∧ (∃k. (σ ⊨ len(k))))   using AndExistsLen valid-def by auto
  have 3: ((σ⊨▷ f) ∧ (∃k. (σ ⊨ len(k)))) =
         (∃k.  (σ⊨▷ f) ∧  (σ ⊨ len(k)))  by auto
  have 4: (σ ⊨ di(▷ f)) = (∃k. (σ ⊨ di(▷f ∧_i len(k)))) using 2 3 DiEqvDi by auto
  have 5: (∃k. (σ ⊨ di(▷f ∧_i len(k))))  using 1 by auto
  then obtain i where 6: (σ ⊨ di(▷f ∧_i len(i)))  by blast
  from 5 obtain j where 7: (σ ⊨ di(▷f ∧_i len(j)))  by blast
  have 8: (σ ⊨ di(▷f ∧_i len(i))) ∧ (σ ⊨ di(▷f ∧_i len(j)))  using 6 7 by auto
  hence 9: (σ ⊨ di(▷f ∧_i len(i)) ∧_i di(▷f ∧_i len(j)))   by simp
  hence 10: i=j  using FstLenSame by blast
  have 11: ⋀j. (σ ⊨ di(▷f ∧_i len(j))) ⟶ (j=i)  using 9 10 using FstLenSame and-defs by blast
  thus (∃! k. (σ ⊨ di(  ▷f ∧_i len(k) ))) using 11 5 by blast
  qed
qed
```

```
lemma LFstAndDist-help:
 (σ ⊨((▷f ∧_i g1) ∧_i len(k));h1  ∧_i ((▷f ∧_i g2) ∧_i len(k));h2) =
  (σ ⊨((▷f ∧_i g1) ∧_i (▷f ∧_i g2) ∧_i len(k));(h1 ∧_i h2) )
using LFixedAndDistr using itl-eq by blast

lemma LFstAndDist-help-1:
 (∃ k. (σ ⊨((▷f ∧_i g1) ∧_i len(k));h1  ∧_i ((▷f ∧_i g2) ∧_i len(k));h2)) =
           (∃ k. (σ ⊨((▷f ∧_i g1) ∧_i (▷f ∧_i g2) ∧_i len(k));(h1 ∧_i h2) ))
proof
  assume 0: ∃k. σ ⊨  ((▷ f ∧_i g1) ∧_i len k) ; h1  ∧_i  ((▷ f ∧_i g2) ∧_i len k) ; h2
  obtain k where 1:  σ ⊨  ((▷ f ∧_i g1) ∧_i len k) ; h1  ∧_i  ((▷ f ∧_i g2) ∧_i len k) ; h2
  using 0 by auto
  hence 2: (σ ⊨((▷f ∧_i g1) ∧_i (▷f ∧_i g2) ∧_i len(k));(h1 ∧_i h2))
  using  LFstAndDist-help by blast
  show (∃ k. (σ ⊨((▷f ∧_i g1) ∧_i (▷f ∧_i g2) ∧_i len(k));(h1 ∧_i h2) ))
  using 2 by auto
  next
  assume 3: (∃ k. (σ ⊨((▷f ∧_i g1) ∧_i (▷f ∧_i g2) ∧_i len(k));(h1 ∧_i h2) ))
  obtain k where 4: (σ ⊨((▷f ∧_i g1) ∧_i (▷f ∧_i g2) ∧_i len(k));(h1 ∧_i h2) )
  using 3 by auto
  hence 5: (σ ⊨((▷f ∧_i g1) ∧_i len(k));h1  ∧_i ((▷f ∧_i g2) ∧_i len(k));h2)
  using  LFstAndDist-help by blast
  show (∃ k. (σ ⊨((▷f ∧_i g1) ∧_i len(k));h1  ∧_i ((▷f ∧_i g2) ∧_i len(k));h2))
  using 5 by auto
qed

lemma LFstAndDistrsem:
 (∀ σ. (σ ⊨(▷f ∧_i g1);h1 ∧_i (▷f ∧_i g2);h2 ≡_i (▷f ∧_i g1 ∧_i g2);(h1 ∧_i h2)))
proof
 fix σ
 show (σ ⊨(▷f ∧_i g1);h1 ∧_i (▷f ∧_i g2);h2 ≡_i (▷f ∧_i g1 ∧_i g2);(h1 ∧_i h2))
 proof -
  have 1: (σ ⊨(▷f ∧_i g1);h1) = (∃ i. (σ ⊨(▷f ∧_i g1 ∧_i len(i));h1) )
  using AndExistsLenChop by auto
  have 2: (σ ⊨(▷f ∧_i g2);h2) = (∃ j. (σ ⊨(▷f ∧_i g2 ∧_i len(j));h2) )
  using AndExistsLenChop by auto
  have 3: (σ ⊨(▷f ∧_i g1);h1 ∧_i (▷f ∧_i g2);h2) =
        ( (∃ i j. (σ ⊨(▷f ∧_i g1 ∧_i len(i));h1  ∧_i
                        (▷f ∧_i g2 ∧_i len(j));h2) )
        )
  using 1 2 by auto
  have 4: ( (∃ i j. (σ ⊨(▷f ∧_i g1 ∧_i len(i));h1  ∧_i
                        (▷f ∧_i g2 ∧_i len(j));h2) )
        ) =
        ( (∃ k. (σ ⊨(▷f ∧_i g1 ∧_i len(k));h1  ∧_i
                        (▷f ∧_i g2 ∧_i len(k));h2) )
        )
  using FstLenSameChop by blast
  have 5: (∃ k. (σ ⊨((▷f ∧_i g1) ∧_i len(k));h1  ∧_i ((▷f ∧_i g2) ∧_i len(k));h2)) =
          (∃ k. (σ ⊨((▷f ∧_i g1) ∧_i (▷f ∧_i g2) ∧_i len(k));(h1 ∧_i h2) ))
  using LFstAndDist-help-1 by blast
  have 6 :  (∃ k. (σ ⊨(▷f ∧_i g1 ∧_i ▷f ∧_i g2 ∧_i len(k));(h1 ∧_i h2) )) =
           (σ ⊨(▷f ∧_i g1 ∧_i ▷f ∧_i g2);(h1 ∧_i h2))
  using AndExistsLenChop by auto
  have 7 : (σ ⊨(▷f ∧_i g1 ∧_i ▷f ∧_i g2);(h1 ∧_i h2)) =
          (σ ⊨(▷f ∧_i g1 ∧_i g2);(h1 ∧_i h2))
  by auto
  from 3 4 5 6 7 show ?thesis by auto
  qed
qed

lemma LFstAndDistr:
 ⊢ (▷f ∧_i g1);h1 ∧_i (▷f ∧_i g2);h2 ≡_i (▷f ∧_i g1 ∧_i g2);(h1 ∧_i h2)
using  LFstAndDistrsem by simp

lemma LFstAndDistrA:
 ⊢ (▷f ∧_i g1);h ∧_i (▷f ∧_i g2);h ≡_i (▷f ∧_i g1 ∧_i g2);h
proof –
 have 1: ⊢ (▷f ∧_i g1);h ∧_i (▷f ∧_i g2);h ≡_i (▷f ∧_i g1 ∧_i g2);(h ∧_i h)  using LFstAndDistr by blast
 have 2: ⊢ (▷f ∧_i g1 ∧_i g2);(h ∧_i h) ≡_i (▷f ∧_i g1 ∧_i g2);h  by auto
 from 1 2 show ?thesis by auto
qed

lemma LFstAndDistrB:
 ⊢ (▷f ∧_i g);h1 ∧_i (▷f ∧_i g);h2 ≡_i (▷f ∧_i g);(h1 ∧_i h2)
proof –
 have 1: ⊢ (▷f ∧_i g);h1 ∧_i (▷f ∧_i g);h2 ≡_i (▷f ∧_i g ∧_i g);(h1 ∧_i h2)  using LFstAndDistr by blast
 have 2: ⊢ (▷f ∧_i g ∧_i g);(h1 ∧_i h2) ≡_i (▷f ∧_i g);(h1 ∧_i h2)  by auto
 from 1 2 show ?thesis by auto
qed

lemma LFstAndDistrC:
  ⊢ (▷f );h1 ∧_i (▷f );h2 ≡_i (▷f);(h1 ∧_i h2)
proof –
 have 1: ⊢ (▷f ∧_i true_i);h1 ∧_i (▷f ∧_i true_i);h2 ≡_i (▷f ∧_i true_i ∧_i true_i);(h1 ∧_i h2)
     using LFstAndDistr by blast
 have 2: ⊢ (▷f ∧_i true_i);h1 ≡_i (▷f );h1
     by auto
 have 3: ⊢ (▷f ∧_i true_i);h2 ≡_i (▷f );h2
     by auto
 have 4: ⊢ (▷f ∧_i true_i ∧_i true_i);(h1 ∧_i h2) ≡_i (▷f);(h1 ∧_i h2)
     by auto
 from 1 2 3 4 show ?thesis by auto
qed

lemma LFstAndDistrD:
 ⊢ di(▷f ∧_i g1) ∧_i di(▷f ∧_i g2) ≡_i di(▷f ∧_i g1 ∧_i g2)
proof –
 have 1: ⊢ (▷f ∧_i g1);true_i ∧_i (▷f ∧_i g2);true_i ≡_i (▷f ∧_i g1 ∧_i g2);(true_i ∧_i true_i)
     using LFstAndDistr by blast
```

have *2*: $\vdash (\rhd f \wedge_i g1);true_i \equiv_i di(\rhd f \wedge_i g1)$
by (*simp add*: *di-d-def*)
have *3*: $\vdash (\rhd f \wedge_i g2);true_i \equiv_i di(\rhd f \wedge_i g2)$
by (*simp add*: *di-d-def*)
have *4*: $\vdash (\rhd f \wedge_i g1 \wedge_i g2);(true_i \wedge_i true_i) \equiv_i di(\rhd f \wedge_i g1 \wedge_i g2)$
by (*simp add*: *di-d-def*)
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *LstAndDistr*:
$\vdash h1;(\lhd f \wedge_i g1) \wedge_i h2;(\lhd f \wedge_i g2) \equiv_i (h1 \wedge_i h2);(\lhd f \wedge_i g1 \wedge_i g2)$
proof −
have *1*: $\vdash (\rhd(f^r) \wedge_i g1^r);(h1^r) \wedge_i (\rhd(f^r) \wedge_i (g2^r));(h2^r) \equiv_i$
$(\rhd(f^r) \wedge_i (g1^r) \wedge_i (g2^r));((h1^r) \wedge_i (h2^r))$ **using** *LFstAndDistr* **by** *blast*
hence *2*: $\vdash ((\rhd(f^r) \wedge_i g1^r);(h1^r) \wedge_i (\rhd(f^r) \wedge_i (g2^r));(h2^r))^r \equiv_i$
$((\rhd(f^r) \wedge_i (g1^r) \wedge_i (g2^r));((h1^r) \wedge_i (h2^r)))^r$ **using** *1 REqvRule* **by** *blast*
have *3*: $\vdash ((\rhd(f^r) \wedge_i g1^r);(h1^r) \wedge_i (\rhd(f^r) \wedge_i (g2^r));(h2^r))^r \equiv_i$
$((\rhd(f^r) \wedge_i g1^r);(h1^r))^r \wedge_i ((\rhd(f^r) \wedge_i (g2^r));(h2^r))^r$
using *RAnd* **by** *blast*
have *4*: $\vdash ((\rhd(f^r) \wedge_i g1^r);(h1^r))^r \wedge_i ((\rhd(f^r) \wedge_i (g2^r));(h2^r))^r \equiv_i$
$(h1^r)^r;(\rhd(f^r) \wedge_i g1^r)^r \wedge_i (h2^r)^r;(\rhd(f^r) \wedge_i (g2^r))^r$
by *simp*
have *5*: $\vdash (h1^r)^r \equiv_i h1$ **using** *EqvReverseReverse itl-prop*(*30*) **by** *blast*
have *6*: $\vdash (h2^r)^r \equiv_i h2$ **using** *EqvReverseReverse itl-prop*(*30*) **by** *blast*
have *7*: $\vdash (g1^r)^r \equiv_i g1$ **using** *EqvReverseReverse itl-prop*(*30*) **by** *blast*
have *8*: $\vdash (g2^r)^r \equiv_i g2$ **using** *EqvReverseReverse itl-prop*(*30*) **by** *blast*
have *9*: $\vdash (f^r)^r \equiv_i f$**using** *EqvReverseReverse itl-prop*(*30*) **by** *blast*
have *10*: $\vdash (\rhd(f^r) \wedge_i g1^r)^r \equiv_i (\rhd(f^r))^r \wedge_i (g1^r)^r$ **using** *RAnd* **by** *blast*
have *11*: $\vdash (\rhd(f^r) \wedge_i g2^r)^r \equiv_i (\rhd(f^r))^r \wedge_i (g2^r)^r$ **using** *RAnd* **by** *blast*
have *12*: $\vdash (\rhd(f^r))^r \equiv_i \lhd (f)$ **using** *RRFirstEqvLast* **by** *blast*
have *13*: $\vdash (\rhd(f^r))^r \wedge_i (g1^r)^r \equiv_i \lhd f \wedge_i g1$ **using** *12 7* **by** *auto*
have *14*: $\vdash (\rhd(f^r))^r \wedge_i (g2^r)^r \equiv_i \lhd f \wedge_i g2$ **using** *12 8* **by** *auto*
have *15*: $\vdash (h1^r)^r;(\rhd(f^r) \wedge_i g1^r)^r \wedge_i (h2^r)^r;(\rhd(f^r) \wedge_i (g2^r))^r \equiv_i$
$h1;(\lhd f \wedge_i g1) \wedge_i h2;(\lhd f \wedge_i g2)$ **using** *14 13 10 11 5 6* **by** *auto*
have *16*: $\vdash ((\rhd(f^r) \wedge_i (g1^r) \wedge_i (g2^r));((h1^r) \wedge_i (h2^r)))^r \equiv_i$
$((h1^r) \wedge_i (h2^r))^r;((\rhd (f^r)) \wedge_i (g1^r) \wedge_i (g2^r))^r$ **by** *simp*
have *17*: $\vdash ((\rhd (f^r)) \wedge_i (g1^r) \wedge_i (g2)^r)^r \equiv_i ((\rhd (f^r))^r \wedge_i (g1^r)^r \wedge_i (g2^r)^r)$ **using** *RAnd* **by** *auto*
have *18*: $\vdash ((\rhd (f^r))^r \wedge_i (g1^r)^r \wedge_i (g2^r)^r) \equiv_i \lhd f \wedge_i g1 \wedge_i g2$ **using** *12 7 8* **by** *auto*
have *19*: $\vdash ((h1^r) \wedge_i (h2^r))^r \equiv_i h1 \wedge_i h2$ **using** *RRAnd* **by** *auto*
have *20*: $\vdash ((h1^r) \wedge_i (h2^r))^r;((\rhd (f^r)) \wedge_i (g1^r) \wedge_i (g2^r))^r \equiv_i$
$(h1 \wedge_i h2);(\lhd f \wedge_i g1 \wedge_i g2)$ **using** *19 17 18* **by** *auto*
from *20 15 1 2 3 4* **show** *?thesis* **by** *simp*
qed

lemma *LstAndDistrA*:
$\vdash h;(\lhd f \wedge_i g1) \wedge_i h;(\lhd f \wedge_i g2) \equiv_i h;(\lhd f \wedge_i g1 \wedge_i g2)$
proof −
have *1*: $\vdash h;(\lhd f \wedge_i g1) \wedge_i h;(\lhd f \wedge_i g2) \equiv_i (h \wedge_i h);(\lhd f \wedge_i g1 \wedge_i g2)$
using *LstAndDistr* **by** *blast*

have *2*: $\vdash (h \wedge_i h);(\lhd f \wedge_i g1 \wedge_i g2) \equiv_i h;(\lhd f \wedge_i g1 \wedge_i g2)$
  by *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *LstAndDistrB*:
$\vdash h1;(\lhd f \wedge_i g) \wedge_i h2;(\lhd f \wedge_i g) \equiv_i (h1 \wedge_i h2);(\lhd f \wedge_i g)$
proof –
have *1*: $\vdash h1;(\lhd f \wedge_i g) \wedge_i h2;(\lhd f \wedge_i g) \equiv_i (h1 \wedge_i h2);(\lhd f \wedge_i g \wedge_i g)$
  using *LstAndDistr* **by** *blast*
have *2*: $\vdash (h1 \wedge_i h2);(\lhd f \wedge_i g \wedge_i g) \equiv_i (h1 \wedge_i h2);(\lhd f \wedge_i g)$
  by *auto*
from *1 2* **show** *?thesis* **by** *auto*
qed

lemma *LstAndDistrC*:
$\vdash h1;(\lhd f) \wedge_i h2;(\lhd f) \equiv_i (h1 \wedge_i h2);(\lhd f)$
proof –
have *1*: $\vdash h1;(\lhd f \wedge_i true_i) \wedge_i h2;(\lhd f \wedge_i true_i) \equiv_i (h1 \wedge_i h2);(\lhd f \wedge_i true_i \wedge_i true_i)$
  using *LstAndDistr* **by** *blast*
have *2*: $\vdash (h1 \wedge_i h2);(\lhd f \wedge_i true_i \wedge_i true_i) \equiv_i (h1 \wedge_i h2);(\lhd f)$
  by *auto*
have *3*: $\vdash h1;(\lhd f \wedge_i true_i) \equiv_i h1;(\lhd f)$
  by *auto*
have *4*: $\vdash h2;(\lhd f \wedge_i true_i) \equiv_i h2;(\lhd f)$
  by *auto*
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *LstAndDistrD*:
$\vdash \Diamond(\lhd f \wedge_i g1) \wedge_i \Diamond(\lhd f \wedge_i g2) \equiv_i \Diamond(\lhd f \wedge_i g1 \wedge_i g2)$
proof –
have *1*: $\vdash true_i;(\lhd f \wedge_i g1) \wedge_i true_i;(\lhd f \wedge_i g2) \equiv_i (true_i \wedge_i true_i);(\lhd f \wedge_i g1 \wedge_i g2)$
  using *LstAndDistr* **by** *blast*
have *2*: $\vdash (true_i \wedge_i true_i);(\lhd f \wedge_i g1 \wedge_i g2) \equiv_i \Diamond(\lhd f \wedge_i g1 \wedge_i g2)$
  by (*simp add*: *sometimes-d-def*)
have *3*: $\vdash true_i;(\lhd f \wedge_i g1) \equiv_i \Diamond(\lhd f \wedge_i g1)$
  by (*simp add*: *sometimes-d-def*)
have *4*: $\vdash true_i;(\lhd f \wedge_i g2) \equiv_i \Diamond(\lhd f \wedge_i g2)$
  by (*simp add*: *sometimes-d-def*)
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *NotFstChop*:
$\vdash \neg_i(\rhd f ;g) \equiv_i \neg_i(di\ (\rhd f)) \vee_i (\rhd f;\neg_i g)$
proof –
have *1*: $\vdash g \supset_i true_i$ **by** *auto*
hence *2*: $\vdash \rhd f;g \supset_i \rhd f;true_i$ **using** *RightChopImpChop* **by** *blast*
hence *3*: $\vdash \rhd f;g \supset_i di(\rhd f)$ **by** (*simp add*:*di-d-def*)
hence *4*: $\vdash \neg_i(di(\rhd f)) \supset_i \neg_i(\rhd f;g)$ **by** *auto*

have *5*: $\vdash (\triangleright f;\ \neg_i g \supset_i \neg_i(\triangleright f;g)) \equiv_i ((\triangleright f;\ \neg_i g) \wedge_i (\triangleright f;g) \supset_i \mathit{false}_i\ )$ **by** *auto*

have *6*: $\vdash (\triangleright f;\ \neg_i g) \wedge_i (\triangleright f;g) \equiv_i \triangleright f;(\neg_i g \wedge_i g)$ **using** *LFstAndDistrC* **by** *blast*

have *7*: $\vdash \neg_i(\triangleright f;(\neg_i g \wedge_i g))$ **by** *auto*

have *8*: $\vdash \triangleright f;\ \neg_i g \supset_i \neg_i(\triangleright f;g)$ **using** *5 6 7* **by** *auto*

have *9*: $\vdash \neg_i(di(\triangleright f)) \vee_i (\triangleright f;\ \neg_i g) \supset_i \neg_i(\triangleright f;g)$ **using** *4 8* **by** *auto*

have *10*: $\vdash di(\triangleright f) \vee_i \neg_i(di(\triangleright f))$ **by** *auto*

hence *11*: $\vdash (\triangleright f;\mathit{true}_i) \vee_i \neg_i(di(\triangleright f))$ **by** (*simp add*: *di-d-def*)

hence *12*: $\vdash (\triangleright f;(g \vee_i \neg_i g)) \vee_i \neg_i(di(\triangleright f))$ **by** *auto*

have *13*: $\vdash (\triangleright f;(g \vee_i \neg_i g)) \equiv_i (\triangleright f;g) \vee_i (\triangleright f;\neg_i g)$ **using** *ChopOrEqv* **by** *auto*

have *14*: $\vdash ((\triangleright f;g) \vee_i (\triangleright f;\neg_i g)) \vee_i \neg_i(di(\triangleright f))$ **using** *12 13* **by** *auto*

hence *15*: $\vdash \neg_i(\triangleright f;g) \supset_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f;\ \neg_i g)$ **by** *auto*

from *9 15* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*

qed

lemma *BsNotFstChop*:
$\vdash bs(\neg_i(\triangleright f;g)) \equiv_i \mathit{empty} \vee_i \neg_i(di(\triangleright f)) \vee_i (\triangleright f;bs\neg_i g)$

proof –

have *1*: $\vdash bs(\neg_i(\triangleright f;g))\equiv_i \mathit{empty} \vee_i bi\neg_i(\triangleright f;g);\mathit{skip}$
by (*simp add*:*bs-d-def*)

have *2*: $\vdash \mathit{empty} \vee_i bi\neg_i(\triangleright f;g);\mathit{skip} \equiv_i \mathit{empty} \vee_i \neg_i(di(\triangleright f;g));\mathit{skip}$
by *auto*

have *3*: $\vdash \mathit{empty} \vee_i \neg_i(di(\triangleright f;g));\mathit{skip} \equiv_i \mathit{empty} \vee_i \neg_i((\triangleright f;g);\mathit{true}_i);\mathit{skip}$
by *auto*

have *4*: $\vdash\ \neg_i((\triangleright f;g);\mathit{true}_i);\mathit{skip} \equiv_i\ \neg_i(\triangleright f;(g;\mathit{true}_i));\mathit{skip}$
using *ChopAssocB* **using** *LeftChopEqvChop itl-prop*(*33*) **by** *blast*

hence *5*: $\vdash \mathit{empty} \vee_i \neg_i((\triangleright f;g);\mathit{true}_i);\mathit{skip} \equiv_i\ \mathit{empty} \vee_i \neg_i(\triangleright f;(g;\mathit{true}_i));\mathit{skip}$
by *auto*

have *6*: $\vdash \mathit{empty} \vee_i \neg_i(\triangleright f;(g;\mathit{true}_i));\mathit{skip} \equiv_i \mathit{empty} \vee_i \neg_i(\triangleright f;di(g));\mathit{skip}$
by (*simp add*: *di-d-def*)

have *7*: $\vdash \mathit{empty} \vee_i \neg_i(\triangleright f;di(g));\mathit{skip} \equiv_i \mathit{empty} \vee_i \neg_i(\neg_i(\neg_i(\triangleright f;di(g));\mathit{skip}))$
by *auto*

have *8*: $\vdash \neg_i(\neg_i(\neg_i(\triangleright f;di(g));\mathit{skip})) \equiv_i\ \neg_i(\mathit{empty} \vee_i (\triangleright f;di(g));\mathit{skip}\ )$
using *NotNotChopSkip* **using** *prop01* **by** *blast*

hence *9*: $\vdash \mathit{empty} \vee_i \neg_i(\neg_i(\neg_i(\triangleright f;di(g));\mathit{skip})) \equiv_i\ \mathit{empty} \vee_i \neg_i(\mathit{empty} \vee_i (\triangleright f;di(g));\mathit{skip}\ )$
by *auto*

have *10*: $\vdash \mathit{empty} \vee_i \neg_i(\mathit{empty} \vee_i (\triangleright f;di(g));\mathit{skip}\ ) \equiv_i \mathit{empty} \vee_i (\mathit{more} \wedge_i \neg_i((\triangleright f;di(g));\mathit{skip}\ ))$
by *auto*

have *11*: $\vdash \mathit{empty} \vee_i (\mathit{more} \wedge_i \neg_i((\triangleright f;di(g));\mathit{skip}\ )) \equiv_i \mathit{empty} \vee_i \neg_i((\triangleright f;di(g));\mathit{skip}\ )$
by *auto*

have *12*: $\vdash\ \mathit{empty} \vee_i \neg_i((\triangleright f;di(g));\mathit{skip}\ ) \equiv_i\ \mathit{empty} \vee_i \neg_i(\triangleright f;(di(g);\mathit{skip})\ )$
using *ChopAssocB 11 itl-prop*(*30*) *prop01 prop03 prop28* **by** *blast*

have *13*: $\vdash\ \neg_i(\triangleright f;(di(g);\mathit{skip})\ ) \equiv_i\ \neg_i(\triangleright f;(ds(g))\ )$
using *DsDi* **using** *RightChopEqvChop itl-prop*(*30*) *itl-prop*(*33*) **by** *blast*

hence *14*: $\vdash \mathit{empty} \vee_i \neg_i(\triangleright f;(di(g);\mathit{skip})\ ) \equiv_i\ \mathit{empty} \vee_i \neg_i(\triangleright f;(ds(g))\ )$
by *auto*

have *15*: $\vdash \mathit{empty} \vee_i \neg_i(\triangleright f;(ds(g))\ ) \equiv_i \mathit{empty} \vee_i \neg_i(di\ (\triangleright f)) \vee_i (\triangleright f;\neg_i(ds\ g))$
using *NotFstChop* **by** *auto*

have *16*: $\vdash\ (\triangleright f;\neg_i(ds\ g))\equiv_i\ (\triangleright f;(bs\ \neg_i g))$
using *NotDsEqvBsNot RightChopEqvChop* **by** *blast*

hence *17*: $\vdash (\mathit{empty} \vee_i \neg_i(di\ (\triangleright f))) \vee_i (\triangleright f;\neg_i(ds\ g))\equiv_i (\mathit{empty} \vee_i \neg_i(di\ (\triangleright f))) \vee_i\ (\triangleright f;(bs\ \neg_i g))$

by *auto*
from *1 2 3 5 6 7 9 10 11 12 14 15 17* **show** *?thesis* **by** *auto*
qed

lemma *FstFstChopEqvFstChopFst*:
$\vdash \rhd(\rhd f;g) \equiv_i \rhd f;\rhd g$
proof −
have *1*: $\vdash \rhd(\rhd f;g) \equiv_i (\rhd f;g) \wedge_i \mathit{bs}\ \neg_i(\rhd f;g)$
by (*simp add*: *first-d-def*)
have *2*: $\vdash \mathit{bs}\ \neg_i(\rhd f;g) \equiv_i \mathit{empty} \vee_i \neg_i(\mathit{di}(\rhd f)) \vee_i (\rhd f;\mathit{bs}\neg_i g)$
using *BsNotFstChop* **by** *auto*
hence *3*: $\vdash (\rhd f;g) \wedge_i \mathit{bs}\ \neg_i(\rhd f;g) \equiv_i (\rhd f;g) \wedge_i (\mathit{empty} \vee_i \neg_i(\mathit{di}(\rhd f)) \vee_i (\rhd f;\mathit{bs}\neg_i g))$
by *auto*
have *4*: $\vdash (\rhd f;g) \wedge_i (\mathit{empty} \vee_i \neg_i(\mathit{di}(\rhd f)) \vee_i (\rhd f;\mathit{bs}\neg_i g)) \equiv_i$
$((\rhd f;g) \wedge_i \mathit{empty}) \vee_i ((\rhd f;g) \wedge_i \neg_i(\mathit{di}(\rhd f))) \vee_i ((\rhd f;g) \wedge_i (\rhd f;\mathit{bs}\neg_i g))$
by *auto*
have *5*: $\vdash \neg_i((\rhd f;g) \wedge_i \neg_i(\mathit{di}(\rhd f)))$
by *auto*
hence *6*: $\vdash ((\rhd f;g) \wedge_i \mathit{empty}) \vee_i ((\rhd f;g) \wedge_i \neg_i(\mathit{di}(\rhd f))) \vee_i ((\rhd f;g) \wedge_i (\rhd f;\mathit{bs}\neg_i g)) \equiv_i$
$((\rhd f;g) \wedge_i \mathit{empty}) \vee_i ((\rhd f;g) \wedge_i (\rhd f;\mathit{bs}\neg_i g))$
by *auto*
have *7*: $\vdash ((\rhd f;g) \wedge_i (\rhd f;(\mathit{bs}\neg_i g))) \equiv_i ((\rhd f;(g \wedge_i (\mathit{bs}\neg_i g))))$
using *LFstAndDistrC* **by** *blast*
hence *8*: $\vdash ((\rhd f;g) \wedge_i \mathit{empty}) \vee_i ((\rhd f;g) \wedge_i (\rhd f;(\mathit{bs}\neg_i g))) \equiv_i$
$((\rhd f;g) \wedge_i \mathit{empty}) \vee_i ((\rhd f;(g \wedge_i (\mathit{bs}\neg_i g))))$
by *auto*
have *9*: $\vdash ((\rhd f;g) \wedge_i \mathit{empty}) \vee_i ((\rhd f;(g \wedge_i (\mathit{bs}\neg_i g)))) \equiv_i ((\rhd f;g) \wedge_i \mathit{empty}) \vee_i \rhd f;\rhd g$
by (*simp add*: *first-d-def*)
have *10*: $\vdash ((\rhd f;g) \wedge_i \mathit{empty}) \equiv_i ((\rhd f;\rhd g) \wedge_i \mathit{empty})$
using *FstChopEmptyEqvFstChopFstEmpty* **by** *blast*
hence *11*: $\vdash ((\rhd f;g) \wedge_i \mathit{empty}) \vee_i \rhd f;\rhd g \equiv_i ((\rhd f;\rhd g) \wedge_i \mathit{empty}) \vee_i \rhd f;\rhd g$
by *auto*
have *12*: $\vdash ((\rhd f;\rhd g) \wedge_i \mathit{empty}) \vee_i \rhd f;\rhd g \equiv_i \rhd f;\rhd g$
by *auto*
from *1 3 4 6 8 9 11 12* **show** *?thesis* **by** *auto*
qed

lemma *FstFixFst*:
$\vdash \rhd(\rhd f) \equiv_i \rhd f$
proof −
have *1*: $\vdash \rhd f \equiv_i (\rhd f);\mathit{empty}$ **using** *ChopEmpty* **using** *itl-prop*(*30*) **by** *blast*
hence *2*: $\vdash \rhd(\ \rhd f\ ) \equiv_i \rhd((\rhd f);\mathit{empty})$ **using** *FstEqvRule* **by** *blast*
have *3*: $\vdash \rhd((\rhd f);\mathit{empty}) \equiv_i \rhd f;\rhd \mathit{empty}$ **using** *FstFstChopEqvFstChopFst* **by** *auto*
have *4*: $\vdash \rhd f;\rhd \mathit{empty} \equiv_i \rhd f;\mathit{empty}$ **using** *FstEmpty* **by** *auto*
have *5*: $\vdash \rhd f;\mathit{empty} \equiv_i \rhd f$ **using** *ChopEmpty* **by** *blast*
from *2 3 4 5* **show** *?thesis* **by** *auto*
qed

lemma *FstCSEqvEmpty*:
$\vdash \rhd(f^{\star}) \equiv_i \mathit{empty}$

```
proof −
 have 1: ⊢ ▷(f⋆) ≡_i ▷(empty ∨_i ((f ∧_i more);f⋆)) using ChopstarEqv FstEqvRule by blast
 from 1 show ?thesis using FstEmptyOrEqvEmpty by auto
qed

lemma FstIterFixFst:
⊢ power (▷ f) n ≡_i ▷(power (▷ f) n)
proof
 (induct n)
 case 0
 then show ?case
 proof −
  have 1: ⊢ power (▷ f) 0 ≡_i empty by auto
  have 2: ⊢ empty ≡_i ▷ empty using FstEmpty by auto
  have 3: ⊢ ▷ empty ≡_i ▷(power (▷f) 0) by auto
  from 1 2 3 show ?thesis by auto
 qed
 next
  case (Suc n)
  then show ?case
  proof −
   have 4: ⊢ (power (▷ f) (Suc n)) ≡_i (▷ f) ;(power (▷f) n)
   using pow-Suc by simp
   have 5: ⊢ (▷ f) ;(power (▷f) n) ≡_i (▷ f) ; ▷ (power (▷ f) n)
   using RightChopEqvChop Suc.hyps by blast
   have 6: ⊢ (▷ f) ; ▷ (power (▷ f) n) ≡_i ▷(▷f;(power (▷ f) n))
   using FstFstChopEqvFstChopFst by auto
   have 7: ⊢ ▷(▷f;(power (▷ f) n)) ≡_i ▷(power (▷ f) (Suc n))
   using pow-Suc by simp
   from 4 5 6 7 show ?thesis by auto
  qed
qed

lemma DsImpNotFst:
⊢ ds f ⊃_i (¬_i(▷f))
proof −
 have 1: ⊢ ds f ∧_i ▷f ≡_i ds f ∧_i (f ∧_i bs ¬_i f) by (simp add: first-d-def)
 have 2: ⊢ ds f ∧_i (f ∧_i bs ¬_i f) ≡_i ds f ∧_i f ∧_i ¬_i(ds f)  using NotDsEqvBsNot by auto
 from 1 2 show ?thesis by auto
qed

lemma FstLenAndEqvLenAnd:
⊢ ▷(len(k) ∧_i f) ≡_i len(k) ∧_i f
proof −
 have 1: ⊢ len(k) ∧_i f ∧_i ds(len(k)∧_i f) ⊃_i ds (len(k))
      using DsAndImpElimL by auto
 hence 2: ⊢ len(k) ∧_i f ∧_i ds(len(k)∧_i f) ⊃_i (di(len(k)));skip
      using DsDi itl-prop(31) prop02 by blast
 hence 3: ⊢ len(k) ∧_i f ∧_i ds(len(k)∧_i f) ⊃_i ((len(k);true_i));skip
      by (simp add: di-d-def)
```

hence *4*: $\vdash len(k) \wedge_i f \wedge_i ds(len(k) \wedge_i f) \supset_i (len(k);(true_i;skip))$
using *ChopAssocB itl-prop*(*31*) *prop02* **by** *blast*
hence *5*: $\vdash len(k) \wedge_i f \wedge_i ds(len(k) \wedge_i f) \supset_i (len(k);(skip;true_i))$
using *SkipTrueEqvTrueSkip TrueChopSkipEqvSkipChopTrue RightChopEqvChop itl-prop*(*31*) *prop02* **by** *blast*
hence *6*: $\vdash len(k) \wedge_i f \wedge_i ds(len(k) \wedge_i f) \supset_i (len(k);(skip;true_i)) \wedge_i len(k)$
by *auto*
hence *7*: $\vdash len(k) \wedge_i f \wedge_i ds(len(k) \wedge_i f) \supset_i (len(k);(skip;true_i)) \wedge_i len(k);empty$
using *ChopEmpty* **by** (*metis itl-prop*(*31*) *itl-prop*(*32*) *prop02*)
hence *8*: $\vdash len(k) \wedge_i f \wedge_i ds(len(k) \wedge_i f) \supset_i (len(k);((skip;true_i) \wedge_i empty))$
using *LFixedAndDistrB1* **using** *itl-prop*(*31*) *prop02* **by** *blast*
have *9*: $\vdash \neg_i(len(k);((skip;true_i) \wedge_i empty))$
by *auto*
have *10*: $\vdash len(k) \wedge_i f \supset_i \neg_i(ds(len(k) \wedge_i f))$
using *8 9* **by** *auto*
hence *11*: $\vdash len(k) \wedge_i f \supset_i bs\ \neg_i(len(k) \wedge_i f)$
using *NotDsEqvBsNot* **by** *auto*
hence *12*: $\vdash len(k) \wedge_i f \supset_i len(k) \wedge_i f \wedge_i bs\ \neg_i(len(k) \wedge_i f)$
by *auto*
hence *13*: $\vdash len(k) \wedge_i f \supset_i \triangleright(len(k) \wedge_i f)$
by (*simp add*: *first-d-def*)
have *14*: $\vdash \triangleright(len(k) \wedge_i f) \supset_i len(k) \wedge_i f$
by (*simp add*: *first-d-def*)
from *13 14* **show** *?thesis* **using** *itl-prop*(*31*) **by** *blast*
qed

lemma *FstAndElimL*:
$\vdash \triangleright f \supset_i f$
by (*simp add*: *first-d-def*)

lemma *FstImpNotDiChopSkip*:
$\vdash \triangleright f \supset_i \neg_i(di\ f;skip)$
proof −
have *1*: $\vdash \triangleright f \supset_i bs\ \neg_i f$ **by** (*simp add*: *first-d-def*)
hence *2*: $\vdash \triangleright f \supset_i \neg_i(ds\ f)$ **using** *NotDsEqvBsNot* **by** *auto*
have *3*: $\vdash ds\ f \equiv_i di\ f\ ;\ skip$ **using** *DsDi* **by** *blast*
hence *4*: $\vdash \neg_i(ds\ f) \equiv_i \neg_i(di\ f;skip)$ **by** *auto*
from *2 4* **show** *?thesis* **by** *auto*
qed

lemma *FstImpNotDiChopSkipB*:
$\vdash \triangleright f \supset_i \neg_i(di\ (f;skip))$
proof −
have *1*: $\vdash \triangleright f \supset_i bs\ \neg_i f$
by (*simp add*: *first-d-def*)
hence *2*: $\vdash \triangleright f \supset_i \neg_i(ds\ f)$
using *NotDsEqvBsNot* **by** *auto*
have *3*: $\vdash ds\ f \equiv_i di\ f\ ;\ skip$
using *DsDi* **by** *blast*
have *4*: $\vdash di\ f\ ;\ skip \equiv_i (f;true_i);skip$

by (*simp add*: *di-d-def*)
have *5*: $\vdash (f;true_i);skip \equiv_i f;(true_i;skip)$
using *ChopAssocB* **by** *blast*
have *6*: $\vdash f;(true_i;skip) \equiv_i f;(skip;true_i)$
using *SkipTrueEqvTrueSkip* **using** *TrueChopSkipEqvSkipChopTrue RightChopEqvChop* **by** *blast*
have *7*: $\vdash f;(skip;true_i) \equiv_i (f;skip);true_i$
using *ChopAssoc* **by** *blast*
have *8*: $\vdash (f;skip);true_i \equiv_i di(f;skip)$
by (*simp add*: *di-d-def*)
have *9*: $\vdash \neg_i(ds\ f) \equiv_i \neg_i(di(f;skip))$
using *3 4 5 6 7 8* **by** *auto*
from *2 9* **show** *?thesis* **by** *auto*
qed

lemma *FstImpDiEqv*:
$\vdash \triangleright f \supset_i (di\ f \equiv_i f)$
proof −
have *1*: $\vdash \triangleright f \supset_i \neg_i(di\ f;skip)$ **using** *FstImpNotDiChopSkip* **by** *blast*
have *2*: $\vdash di\ f \supset_i f \vee_i (di\ f;skip)$ **using** *DiEqvOrDiChopSkipB itl-prop*(*31*) **by** *blast*
have *3*: $\vdash \triangleright f \wedge_i di\ f \supset_i (f \vee_i (di\ f;skip)) \wedge_i \neg_i(di\ f;skip)$ **using** *1 2* **by** *auto*
have *4*: $\vdash (f \vee_i (di\ f;skip)) \wedge_i \neg_i(di\ f;skip) \equiv_i f \wedge_i \neg_i(di\ f;skip)$ **by** *auto*
have *5*: $\vdash \triangleright f \wedge_i di\ f \supset_i f \wedge_i \neg_i(di\ f;skip)$ **using** *3 4* **using** *itl-prop*(*31*) *prop02* **by** *blast*
hence *6*: $\vdash \triangleright f \wedge_i di\ f \supset_i f$ **using** *itl-prop*(*32*) **by** *blast*
hence *7*: $\vdash \triangleright f \supset_i (di\ f \supset_i f)$ **using** *FstAndElimL prop13 prop26 prop32* **by** *blast*
have *8*: $\vdash f \supset_i di\ f$ **using** *DiIntro* **by** *auto*
hence *9*: $\vdash \triangleright f \supset_i (f \supset_i (di\ f))$ **by** *auto*
from *7 9* **show** *?thesis* **by** *auto*
qed

lemma *FstAndDiFstAndEqvFstAnd*:
$\vdash \triangleright f \wedge_i di(\triangleright f \wedge_i g) \equiv_i \triangleright f \wedge_i g$
proof −
have *1*: $\vdash \triangleright f \wedge_i di(\triangleright f \wedge_i g) \supset_i \triangleright f$
by *auto*
have *2*: $\vdash \triangleright f \wedge_i di(\triangleright f \wedge_i g) \supset_i di(\triangleright f \wedge_i g)$
by *auto*
have *3*: $\vdash di(\triangleright f \wedge_i g) \equiv_i (\triangleright f \wedge_i g) \vee_i di((\triangleright f \wedge_i g);skip)$
using *DiEqvOrDiChopSkipA* **by** *blast*
have *4*: $\vdash di((\triangleright f \wedge_i g);skip) \equiv_i ((\triangleright f \wedge_i g);skip);true_i$
by (*simp add*: *di-d-def*)
have *5*: $\vdash \triangleright f \wedge_i di(\triangleright f \wedge_i g) \supset_i (\triangleright f \wedge_i g) \vee_i ((\triangleright f \wedge_i g);skip);true_i$
using *2 3 4* **by** *auto*
have *6*: $\vdash \triangleright f \wedge_i g \supset_i f$
using *FstAndElimL* **by** *auto*
hence *7*: $\vdash ((\triangleright f \wedge_i g);skip);true_i \supset_i (f;skip);true_i$
by *auto*
hence *8*: $\vdash ((\triangleright f \wedge_i g);skip);true_i \supset_i di(f;skip)$
by *auto*
have *9*: $\vdash \triangleright f \supset_i \neg_i(di(f;skip))$
using *FstImpNotDiChopSkipB* **by** *blast*

have *10*: $\vdash \rhd f \wedge_i \ di(\rhd f \wedge_i g) \supset_i ((\rhd f \wedge_i g) \vee_i di(f;skip))$
using *5 8 prop35* **by** *blast*
have *11*: $\vdash \rhd f \wedge_i \ di(\rhd f \wedge_i g) \supset_i \neg_i(di(f;skip)) \wedge_i ((\rhd f \wedge_i g) \vee_i di(f;skip))$
using *9 10 1 itl-prop(32) prop02* **by** *blast*
have *12*: $\vdash \neg_i(di(f;skip)) \wedge_i ((\rhd f \wedge_i g) \vee_i di(f;skip)) \equiv_i \neg_i(di(f;skip)) \wedge_i ((\rhd f \wedge_i g))$
by *auto*
have *13*: $\vdash \rhd f \wedge_i \ di(\rhd f \wedge_i g) \supset_i (\rhd f \wedge_i g)$
using *11 12* **by** *auto*
have *14*: $\vdash (\rhd f \wedge_i g) \supset_i \rhd f$
by *auto*
hence *15*: $\vdash (\rhd f \wedge_i g) \supset_i di(\rhd f \wedge_i g \ )$
using *DiIntro* **by** *auto*
have *16*: $\vdash (\rhd f \wedge_i g) \supset_i \rhd f \wedge_i di(\rhd f \wedge_i g)$
using *14 15* **by** *auto*
from *13 16* **show** *?thesis* **using** *itl-prop(31)* **by** *blast*
qed

lemma *FstAndDiImpBsNotAndDi*:
$\vdash (\rhd f \wedge_i di\ g) \supset_i (bs\ \neg_i(di\ f \wedge_i g))$
proof $-$
have *1*: $\vdash (\rhd f \wedge_i di\ g) \wedge_i \neg_i(bs\ \neg_i(di\ f \wedge_i g)) \supset_i ds(di\ f \wedge_i g)$
by (*simp add*: *ds-d-def*)
hence *2*: $\vdash (\rhd f \wedge_i di\ g) \wedge_i \neg_i(bs\ \neg_i(di\ f \wedge_i g)) \supset_i ds(di\ f)$
using *DsAndImp* **by** *auto*
hence *3*: $\vdash (\rhd f \wedge_i di\ g) \wedge_i \neg_i(bs\ \neg_i(di\ f \wedge_i g)) \supset_i di(di\ f);skip$
using *DsDi* **using** *itl-prop(31) prop02* **by** *blast*
hence *4*: $\vdash (\rhd f \wedge_i di\ g) \wedge_i \neg_i(bs\ \neg_i(di\ f \wedge_i g)) \supset_i di\ f;skip$
using *DiEqvDiDi* **using** *LeftChopImpChop itl-prop(31) prop02* **by** *blast*
hence *5*: $\vdash (\rhd f \wedge_i di\ g) \wedge_i \neg_i(bs\ \neg_i(di\ f \wedge_i g)) \supset_i ds\ f$
using *DsDi* **using** *itl-prop(31) prop02* **by** *blast*
hence *6*: $\vdash \ (\rhd f \wedge_i di\ g) \wedge_i \neg_i(bs\ \neg_i(di\ f \wedge_i g)) \supset_i \neg_i (\rhd\ f)$
using *DsImpNotFst* **using** *itl-prop(31) prop02* **by** *blast*
from *6* **show** *?thesis* **by** *auto*
qed

lemma *FstFstOrEqvFstOrL*:
$\vdash \rhd(\rhd f \vee_i g) \equiv_i \rhd(f \vee_i g)$
proof $-$
have *1*: $\vdash \rhd(f \vee_i g) \equiv_i (f \vee_i g) \wedge_i bs\neg_i(f \vee_i g)$
by (*simp add*: *first-d-def*)
have *2*: $\vdash \ \neg_i(f \vee_i g) \equiv_i \ (\neg_i f \wedge_i \neg_i g)$
by *auto*
hence *3*: $\vdash \ bs\neg_i(f \vee_i g) \equiv_i \ bs\ (\neg_i f \wedge_i \neg_i g)$
using *BsEqvRule* **by** *blast*
have *4*: $\vdash bs\ (\neg_i f \wedge_i \neg_i g) \equiv_i bs\ \neg_i\ f \wedge_i bs\ \neg_i\ g$
using *BsAndEqv itl-prop(30)* **by** *blast*
hence *5*: $\vdash (f \vee_i g) \wedge_i bs\neg_i(f \vee_i g) \equiv_i \ (f \vee_i g) \wedge_i bs\ \neg_i\ f \wedge_i bs\ \neg_i\ g$
using *4 3* **by** *simp*
have *6*: $\vdash (f \vee_i g) \wedge_i bs\ \neg_i\ f \wedge_i bs\ \neg_i\ g \equiv_i$
$((f \wedge_i bs\ \neg_i\ f) \vee_i (g \wedge_i bs\ \neg_i\ f)) \wedge_i bs\ \neg_i\ g$

by *auto*

have *7*: $\vdash ((f \wedge_i bs\ \neg_i f) \vee_i (g \wedge_i bs\ \neg_i f)) \wedge_i bs\ \neg_i g \equiv_i$
$(\triangleright f \vee_i (g \wedge_i bs\ \neg_i f)) \wedge_i bs\ \neg_i g$

by (*simp add*: *first-d-def*)

have *8*: $\vdash (\triangleright f \vee_i (g \wedge_i bs\ \neg_i f)) \wedge_i bs\ \neg_i g \equiv_i$
$((\triangleright f \vee_i g) \wedge_i (\triangleright f \vee_i bs \neg_i f)) \wedge_i bs\ \neg_i g$

by *auto*

have *9*: $\vdash ((\triangleright f \vee_i g) \wedge_i (\triangleright f \vee_i bs \neg_i f)) \wedge_i bs\ \neg_i g \equiv_i$
$((\triangleright f \vee_i g) \wedge_i ((f \wedge_i bs\ \neg_i f) \vee_i bs \neg_i f)) \wedge_i bs\ \neg_i g$

by (*simp add*: *first-d-def*)

have *10*: $\vdash ((\triangleright f \vee_i g) \wedge_i ((f \wedge_i bs\ \neg_i f) \vee_i bs \neg_i f)) \wedge_i bs\ \neg_i g \equiv_i$
$(\triangleright f \vee_i g) \wedge_i bs\ \neg_i f \wedge_i bs\ \neg_i g$

by *auto*

have *11*: $\vdash (\triangleright f \vee_i g) \wedge_i bs\ \neg_i f \wedge_i bs\ \neg_i g \equiv_i$
$(\triangleright f \vee_i g) \wedge_i bs(\neg_i(\triangleright f)) \wedge_i bs\ \neg_i g$

using *BsNotFstEqvBsNot* **using** *itl-prop*(*30*) *prop05 prop06* **by** *blast*

have *12*: $\vdash (\triangleright f \vee_i g) \wedge_i bs(\neg_i(\triangleright f)) \wedge_i bs\ \neg_i g \equiv_i$
$(\triangleright f \vee_i g) \wedge_i bs\ (\neg_i(\triangleright f) \wedge_i \neg_i g)$

using *BsAndEqv* **using** *prop05* **by** *blast*

have *13*: $\vdash\ (\neg_i(\triangleright f) \wedge_i \neg_i g) \equiv_i \neg_i(\triangleright f \vee_i g)$

by *auto*

hence *14*: $\vdash bs\ (\neg_i(\triangleright f) \wedge_i \neg_i g) \equiv_i bs\ \neg_i(\triangleright f \vee_i g)$

using *BsEqvRule* **by** *blast*

hence *15*: $\vdash (\triangleright f \vee_i g) \wedge_i bs\ (\neg_i(\triangleright f) \wedge_i \neg_i g) \equiv_i (\triangleright f \vee_i g) \wedge_i bs\ \neg_i(\triangleright f \vee_i g)$

by *auto*

have *16*: $\vdash (\triangleright f \vee_i g) \wedge_i bs\ \neg_i(\triangleright f \vee_i g) \equiv_i \triangleright(\triangleright f \vee_i g)$

by (*simp add*: *first-d-def*)

from *16 15 12 11 10 9 8 7 6 5 1* **show** *?thesis* **by** *auto*

qed

lemma *FstFstOrEqvFstOrR*:
$\vdash \triangleright(f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i g)$

proof –

have *1*: $\vdash (f \vee_i \triangleright g) \equiv_i (\triangleright g \vee_i f)$ **by** *auto*

hence *2*: $\vdash \triangleright\ (f \vee_i \triangleright g) \equiv_i \triangleright(\triangleright g \vee_i f)$ **using** *FstEqvRule* **by** *blast*

have *3*: $\vdash \triangleright(\triangleright g \vee_i f) \equiv_i \triangleright(g \vee_i f)$ **using** *FstFstOrEqvFstOrL* **by** *blast*

have *4*: $\vdash (g \vee_i f) \equiv_i (f \vee_i g)$ **by** *auto*

hence *5*: $\vdash \triangleright(g \vee_i f) \equiv_i \triangleright(f \vee_i g)$ **using** *FstEqvRule* **by** *blast*

from *2 3 5* **show** *?thesis* **by** *auto*

qed

lemma *FstFstOrEqvFstOr*:
$\vdash \triangleright(\triangleright f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i g)$

proof –

have *1*: $\vdash\ \triangleright(\triangleright f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i \triangleright g)$ **using** *FstFstOrEqvFstOrL* **by** *blast*

have *2*: $\vdash \triangleright(f \vee_i \triangleright g) \equiv_i \triangleright(f \vee_i g)$ **using** *FstFstOrEqvFstOrR* **by** *blast*

from *1 2* **show** *?thesis* **by** *auto*

qed

lemma *FstLenEqvLen*:

$\vdash \rhd(\ len(k)) \equiv_i len(k)$
proof −
have *1*: $\vdash \rhd(\ len(k) \wedge_i true_i) \equiv_i len(k) \wedge_i true_i$ **using** *FstLenAndEqvLenAnd* **by** *blast*
have *2*: $\vdash len(k) \wedge_i true_i \equiv_i len(k)$ **by** *auto*
hence *3*: $\vdash \rhd(\ len(k) \wedge_i true_i) \equiv_i \rhd(len(k))$ **using** *FstEqvRule* **by** *blast*
from *1 2 3* **show** *?thesis* **by** *auto*
qed

lemma *FstSkip*:
$\vdash \rhd\ skip \equiv_i skip$
proof −
have *1*: $\vdash skip \equiv_i len(1)$ **using** *LenOneEqvSkip* **using** *itl-prop*(*30*) **by** *blast*
hence *2*: $\vdash \rhd skip \equiv_i \rhd\ (len(1))$ **using** *FstEqvRule* **by** *blast*
have *3*: $\vdash \rhd(\ len(1)) \equiv_i len(1)$ **using** *FstLenEqvLen* **by** *blast*
from *1 2 3* **show** *?thesis* **using** *LenOneEqvSkip prop03* **by** *blast*
qed

lemma *HaltStateEqvFstFinState*:
$\vdash halt\ (init\ w) \equiv_i \rhd\ (fin\ (init\ w))$
proof −
have *1*: $\vdash halt\ (init\ w) \equiv_i \Box(empty \equiv_i (init\ w))$ **by** (*simp add*: *halt-d-def*)
have *2*: $\vdash \Box(empty \equiv_i (init\ w)) \equiv_i \Box((empty \supset_i (init\ w)) \wedge_i ((init\ w) \supset_i empty))$ **by** *auto*
have *3*: $\vdash \Box((empty \supset_i (init\ w)) \wedge_i ((init\ w) \supset_i empty)) \equiv_i$
$\Box((empty \supset_i (init\ w))) \wedge_i \Box((init\ w) \supset_i empty)$ **by** *auto*
have *4*: $\vdash ((init\ w) \supset_i empty) \equiv_i (more \supset_i \neg_i(init\ w))$ **by** *auto*
hence *5*: $\vdash \Box\ ((init\ w) \supset_i empty) \equiv_i \Box\ (more \supset_i \neg_i(init\ w))$ **using** *BoxEqvBox* **by** *blast*
have *6*: $\vdash \Box\ (more \supset_i \neg_i(init\ w)) \equiv_i bs(\neg_i(fin(init\ w)))$ **using** *BoxMoreStateEqvBsFinState* **by** *blast*
have *7*: $\vdash \Box((empty \supset_i (init\ w))) \equiv_i fin(init\ w)$ **by** (*simp add*: *fin-d-def*)
have *8*: $\vdash\ \Box((empty \supset_i (init\ w))) \wedge_i \Box((init\ w) \supset_i empty) \equiv_i$
$fin(init\ w) \wedge_i bs(\neg_i(fin(init\ w)))$ **using** *5 6 7* **by** *auto*
from *1 2 3 8* **show** *?thesis* **by** (*simp add*:*first-d-def*)
qed

lemma *FstLenEqvLenFst*:
$\vdash \rhd(len\ k\ ;\ f) \equiv_i len\ k\ ;\ \rhd\ f$
proof −
have *1*: $\vdash len\ k\ ;\ f \equiv_i \rhd(len\ k)\ ;\ f$ **using** *FstLenEqvLen LeftChopEqvChop* **by** *auto*
have *2*: $\vdash \rhd(len\ k\ ;\ f) \equiv_i \rhd\ (\rhd(len\ k)\ ;\ f)$ **using** *1 FstEqvRule* **by** *blast*
have *3*: $\vdash \rhd\ (\rhd(len\ k)\ ;\ f) \equiv_i \rhd(len\ k)\ ;\ \rhd f$ **using** *FstFstChopEqvFstChopFst* **by** *blast*
have *4*: $\vdash \rhd(len\ k)\ ;\ \rhd f \equiv_i len\ k\ ;\ \rhd f$ **using** *FstLenEqvLen LeftChopEqvChop* **by** *auto*
from *2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *FstNextEqvNextFst*:
$\vdash \rhd(\bigcirc\ f) \equiv_i \bigcirc(\ \rhd\ f)$
proof −
have *1*: $\vdash \rhd(\bigcirc\ f) \equiv_i \rhd(skip\ ;\ f)$ **using** *FstEqvRule* **by** (*simp add*: *next-d-def*)
have *2*: $\vdash skip\ ;\ f \equiv_i \rhd skip\ ;\ f$ **using** *FstSkip* **by** *auto*
have *3*: $\vdash \rhd(skip\ ;\ f) \equiv_i \rhd\ (\rhd skip\ ;\ f)$ **using** *2 FstEqvRule LeftChopEqvChop* **by** *blast*
have *4*: $\vdash \rhd\ (\rhd skip\ ;\ f) \equiv_i \rhd skip\ ;\ \rhd f$ **using** *3 FstFstChopEqvFstChopFst* **by** *blast*

```
  have 5: ⊢ ▷skip ; ▷f ≡_i skip ; ▷f  using 4 FstSkip LeftChopEqvChop by blast
  have 6: ⊢ skip ; ▷f  ≡_i  ○( ▷ f) by (simp add: next-d-def)
  from 1 2 3 4 5 6 show ?thesis by auto
qed

lemma FstDiamondStateEqvHalt:
 ⊢ ▷ (◇ (init w)) ≡_i halt (init w)
proof −
  have 1: ⊢ ◇ (init w) ≡_i ◇ ((init w) ∧_i true_i) by simp
  have 2: ⊢ fin (init w) ; true_i ≡_i ◇ ((init w) ∧_i true_i) using 1 FinChopEqvDiamond by blast
  have 3: ⊢ fin (init w) ; true_i ≡_i di (fin (init w)) using di-d-def by simp
  have 4: ⊢ (◇ (init w)) ≡_i (di (fin (init w))) using 1 2 3 by auto
  have 5: ⊢ ▷ (◇ (init w)) ≡_i ▷ (di (fin (init w))) using 4 FstEqvRule by blast
  hence 6: ⊢ ▷ (◇ (init w)) ≡_i ▷ (fin (init w)) using FstDiEqvFst by auto
  hence 7: ⊢ ▷ (◇ (init w)) ≡_i halt (init w) using HaltStateEqvFstFinState by auto
  from 7 show ?thesis by simp
qed

lemma FstBoxStateEqvStateAndEmpty:
 ⊢ ▷ (□ (init w)) ≡_i (init w) ∧_i empty
proof −
  have 1: ⊢ (init w) ∧_i (□ (init w))⋆ ≡_i □ (init w)
  using BoxCSEqvBox by blast
  have 2: ⊢ □ (init w) ≡_i (init w) ∧_i (□ (init w))⋆
  using 1 by simp
  hence 3: ⊢ □ (init w) ≡_i (init w) ∧_i (□ (init w))⋆
  by blast
  have 4: ⊢ ((init w) ∧_i empty) ; (□ (init w))⋆ ≡_i (init w) ∧_i (□ (init w))⋆
  using StateAndEmptyChop by blast
  have 5: ⊢ (init w) ∧_i (□ (init w))⋆  ≡_i  ((init w) ∧_i empty) ; (□ (init w))⋆
  using 4 by simp
  have 6: ⊢ □ (init w) ≡_i ((init w) ∧_i empty) ; (□ (init w))⋆
  using 3 5 prop03  by blast
  have 7: ⊢ ((init w) ∧_i empty) ; (□ (init w))⋆ ≡_i ▷ (init w) ; (□ (init w))⋆
  using FstState by auto
  have 8: ⊢ □ (init w) ≡_i ▷ (init w) ; (□ (init w))⋆
  using 6 7 prop03 by blast
  have 9: ⊢ ▷ (□ (init w)) ≡_i ▷ (▷ (init w) ; (□ (init w))⋆)
  using 8 FstEqvRule by blast
  have 10: ⊢ ▷ (▷ (init w) ; (□ (init w))⋆) ≡_i ▷ (init w) ;  ▷ ((□ (init w))⋆)
  using FstFstChopEqvFstChopFst by blast
  have 11: ⊢ ▷ (init w) ;  ▷ ((□ (init w))⋆) ≡_i ▷ (init w) ; empty
  using RightChopEqvChop FstCSEqvEmpty by blast
  have 12: ⊢ ▷ (init w) ; empty ≡_i ▷ (init w)
  using RightChopEqvChop ChopEmpty by blast
  have 13: ⊢ ▷ (init w) ≡_i (init w) ∧_i empty
  using FstState by auto
  from 9 10 11 12 13 show ?thesis by auto
qed
```

lemma *FstAndFstStarEqvFst*:
$\vdash \triangleright f \wedge_i (\triangleright f)^\star \equiv_i \triangleright f$
proof –
have *1*: $\vdash (\triangleright f)^\star \equiv_i \mathit{empty} \vee_i (\triangleright f);(\triangleright f)^\star$ **using** *CSEqvOrChopCS* **by** *simp*
have *2*: $\vdash (\triangleright f)^\star \wedge_i \triangleright f \equiv_i (\mathit{empty} \vee_i (\triangleright f);(\triangleright f)^\star) \wedge_i \triangleright f$ **using** *1 prop06* **by** *blast*
have *3*: $\vdash (\mathit{empty} \vee_i (\triangleright f);(\triangleright f)^\star) \wedge_i \triangleright f \equiv_i (\mathit{empty} \wedge_i \triangleright f) \vee_i ((\triangleright f);(\triangleright f)^\star \wedge_i \triangleright f)$ **by** *auto*
have *4*: $\vdash (\triangleright f)^\star \wedge_i \triangleright f \equiv_i (\mathit{empty} \wedge_i \triangleright f) \vee_i ((\triangleright f);(\triangleright f)^\star \wedge_i \triangleright f)$ **using** *2 3* **by** *simp*
have *5*: $\vdash (\triangleright f);(\triangleright f)^\star \wedge_i \triangleright f \equiv_i (\triangleright f);(\triangleright f)^\star \wedge_i \triangleright f;\mathit{empty}$ **using** *ChopEmpty* **by** *auto*
have *6*: $\vdash (\triangleright f);(\triangleright f)^\star \wedge_i \triangleright f;\mathit{empty} \equiv_i (\triangleright f);((\triangleright f)^\star \wedge_i \mathit{empty})$ **using** *LFstAndDistrC* **by** *blast*
have *7*: $\vdash (\triangleright f)^\star \wedge_i \mathit{empty} \equiv_i \mathit{empty}$ **using** *EmptyImpCS* **by** *auto*
have *8*: $\vdash (\triangleright f);((\triangleright f)^\star \wedge_i \mathit{empty}) \equiv_i \triangleright f$ **using** *7 RightChopEqvChop ChopEmpty* **by** *auto*
have *9*: $\vdash (\triangleright f);(\triangleright f)^\star \wedge_i \triangleright f \equiv_i \triangleright f$ **using** *5 6 8* **by** *simp*
have *10*: $\vdash (\triangleright f)^\star \wedge_i \triangleright f \equiv_i (\mathit{empty} \wedge_i \triangleright f) \vee_i \triangleright f$ **using** *4 9* **by** *simp*
have *11*: $\vdash (\mathit{empty} \wedge_i \triangleright f) \vee_i \triangleright f \equiv_i \triangleright f$ **by** *auto*
have *12*: $\vdash (\triangleright f)^\star \wedge_i \triangleright f \equiv_i \triangleright f$ **using** *10 11* **by** *simp*
from *12* **show** *?thesis* **by** *auto*
qed

lemma *DiHaltAndDiHaltAndEqvDiHaltAndAnd*:
$\vdash \mathit{di}(\mathit{halt}\ (\mathit{init}\ w) \wedge_i f) \wedge_i \mathit{di}(\mathit{halt}\ (\mathit{init}\ w) \wedge_i g) \equiv_i \mathit{di}(\mathit{halt}\ (\mathit{init}\ w) \wedge_i f \wedge_i g)$
proof –
have *1*: $\vdash \mathit{di}(\mathit{halt}\ (\mathit{init}\ w) \wedge_i f) \wedge_i \mathit{di}(\mathit{halt}\ (\mathit{init}\ w) \wedge_i g) \equiv_i$
$\mathit{di}(\triangleright(\mathit{fin}\ (\mathit{init}\ w)) \wedge_i f) \wedge_i \mathit{di}\ (\triangleright(\mathit{fin}\ (\mathit{init}\ w)) \wedge_i g)$
using *HaltStateEqvFstFinState* **by** *auto*
have *2*: $\vdash \mathit{di}(\triangleright(\mathit{fin}\ (\mathit{init}\ w)) \wedge_i f) \wedge_i \mathit{di}(\triangleright(\mathit{fin}\ (\mathit{init}\ w)) \wedge_i g) \equiv_i$
$\mathit{di}(\triangleright(\mathit{fin}\ (\mathit{init}\ w)) \wedge_i f \wedge_i g)$
using *LFstAndDistrD* **by** *simp*
have *3*: $\vdash \mathit{di}(\triangleright(\mathit{fin}\ (\mathit{init}\ w)) \wedge_i f \wedge_i g) \equiv_i \mathit{di}(\mathit{halt}\ (\mathit{init}\ w) \wedge_i f \wedge_i g)$
using *HaltStateEqvFstFinState* **by** *auto*
from *1 2 3* **show** *?thesis* **by** *simp*
qed

lemma *HaltStateEqvFstHaltState*:
$\vdash \mathit{halt}(\mathit{init}(w)) \equiv_i \triangleright(\mathit{halt}(\mathit{init}(w)))$
proof –
have *1*: $\vdash \mathit{halt}\ (\mathit{init}\ w) \equiv_i \triangleright\ (\mathit{fin}\ (\mathit{init}\ w))$
using *HaltStateEqvFstFinState* **by** *blast*
have *2*: $\vdash \triangleright\ (\mathit{fin}\ (\mathit{init}\ w)) \equiv_i \triangleright\ (\triangleright\ (\mathit{fin}\ (\mathit{init}\ w)))$
using *FstEqvRule FstFixFst* **using** *itl-prop*(*30*) **by** *blast*
have *3*: $\vdash \triangleright\ (\triangleright\ (\mathit{fin}\ (\mathit{init}\ w))) \equiv_i \triangleright(\mathit{halt}(\mathit{init}(w)))$
using *FstEqvRule HaltStateEqvFstFinState* **using** *itl-prop*(*30*) **by** *blast*
from *1 2 3* **show** *?thesis* **by** *fastforce*
qed

lemma *counter-ex-lhs*:
$\vdash ((\triangleright(\mathit{len}(5)) \wedge_i \triangleright(\mathit{len}(2)))\ ;\ (\mathit{len}(5) \vee_i \mathit{len}(2))) \equiv_i \mathit{false}_i$
proof –
have *0*: $\vdash ((\triangleright(\mathit{len}(5)) \wedge_i \triangleright(\mathit{len}(2)))\ ) \equiv_i$

$(len(5) \wedge_i len(2))$
using *FstLenEqvLen* **by** *auto*
have *1*: $\vdash ((\triangleright(len(5)) \wedge_i \triangleright(len(2)))\ ;\ (len(5) \vee_i len(2))) \equiv_i$
$(len(5) \wedge_i len(2)); (len(5) \vee_i len(2))$
using *0* **using** *LeftChopEqvChop* **by** *blast*
have *2*: $\vdash (len(5) \wedge_i len(2)) \equiv_i false_i$
by (*simp*)
have *3*: $\vdash ((len(5) \wedge_i len(2)); (len(5) \vee_i len(2))) \equiv_i (false_i;(len(5) \vee_i len(2)))$
by (*simp add*: *2 LeftChopEqvChop*)
have *4*: $\vdash (false_i;(len(5) \vee_i len(2))) \equiv_i false_i$
by (*simp*)
from *1 3 4* **show** *?thesis* **by** *fastforce*
qed

lemma *counter-extra*:
assumes $\vdash f \equiv_i g$
$\vdash f1 \equiv_i g1$
shows $\vdash f \vee_i f1 \equiv_i g \vee_i g1$
using *assms*(*1*) *assms*(*2*) **by** *simp*

lemma *counter-ex-rhs*:
$\vdash ((\triangleright\ (len(5))\ ;\ (len(5) \vee_i len(2))) \wedge_i (\triangleright\ (len(2))\ ;\ (len(5) \vee_i len(2)))) \equiv_i len(7)$
proof −
have *1*: $\vdash (\triangleright\ (len(5))\ ;\ (len(5) \vee_i len(2))) \equiv_i$
$len(5);(len(5) \vee_i len(2))$
using *FstLenEqvLen LeftChopEqvChop* **by** *blast*
have *2*: $\vdash (\triangleright\ (len(2))\ ;\ (len(5) \vee_i len(2))) \equiv_i$
$len(2)\ ;(len(5) \vee_i len(2))$
using *FstLenEqvLen LeftChopEqvChop* **by** *blast*
have *3*: $\vdash len(5);(len(5) \vee_i len(2)) \equiv_i$
$((len(5);len(5)) \vee_i (len(5);len(2)))$
using *ChopOrEqv* **by** *blast*
have *4*: $\vdash ((len(5);len(5)) \vee_i (len(5);len(2))) \equiv_i$
$(len(10) \vee_i len(7))$
using *LenEqvLenChopLen*
by (*smt Suc-numeral add-2-eq-Suc′ arith-simps*(*4*) *arith-simps*(*7*) *iff-defs itl-valid numeral-Bit0 numeral-nat*(*3*) *or-defs*)
have *5*: $\vdash len(2)\ ;(len(5) \vee_i len(2)) \equiv_i$
$((len(2);len(5)) \vee_i (len(2);len(2)))$
using *ChopOrEqv* **by** *blast*
have *60*: $\vdash ((len(2);len(5))\ ) \equiv_i$
$(len(7)\ )$
using *LenEqvLenChopLen*
by (*smt Suc-numeral add-numeral-left itl-prop*(*30*) *numeral-One plus-1-eq-Suc semiring-norm*(*2*) *semiring-norm*(*5*) *semiring-norm*(*8*))
have *61*: $\vdash ((len(2);len(2))\ ) \equiv_i$
$(len(4)\ )$
using *LenEqvLenChopLen*
by (*smt Suc-numeral add-numeral-left itl-prop*(*30*) *numeral-One plus-1-eq-Suc semiring-norm*(*2*) *semiring-norm*(*5*) *semiring-norm*(*8*))

have *6*: $\vdash ((len(2);len(5)) \vee_i (len(2);len(2))) \equiv_i$
$(len(7) \vee_i len(4))$
using *60 61 counter-extra* **by** *blast*
have *7*: $\vdash ((len(10) \vee_i len(7)) \wedge_i (len(7) \vee_i len(4))) \equiv_i$
$((len(7) \vee_i len(10)) \wedge_i (len(7) \vee_i len(4)))$
by *fastforce*
have *8*: $\vdash ((len(7) \vee_i len(10)) \wedge_i (len(7) \vee_i len(4))) \equiv_i$
$(len(7) \vee_i (len(10) \wedge_i len(4)))$
by *fastforce*
have *9*: $\vdash (len(10) \wedge_i len(4)) \equiv_i false_i$
by (*simp*)
have *10* : $\vdash (len(7) \vee_i (len(10) \wedge_i len(4))) \equiv_i len(7)$
using *9* **by** *auto*
have *11*: $\vdash ((\triangleright (len(5)) ; (len(5) \vee_i len(2))) \wedge_i (\triangleright (len(2)) ; (len(5) \vee_i len(2)))) \equiv_i$
$(len(5);(len(5) \vee_i len(2)) \wedge_i len(2) ;(len(5) \vee_i len(2)))$
using *1 2* **by** *auto*
have *12*: $\vdash (len(5);(len(5) \vee_i len(2)) \wedge_i len(2) ;(len(5) \vee_i len(2))) \equiv_i$
$(len\ 10 \vee_i len\ 7) \wedge_i (len\ 7 \vee_i len\ 4)$
using *4 6* **by** *auto*
have *13*: $\vdash (len(5);(len(5) \vee_i len(2)) \wedge_i len(2) ;(len(5) \vee_i len(2))) \equiv_i len(7)$
using *12 10 3 5 8 7* **using** *prop03* **by** *blast*
from *11 13* **show** *?thesis* **using** *prop03* **by** *blast*
qed

end

theory *Monitor*
imports *First*

begin

7 Monitors

The RV monitors language is introduced plus the algebraic properties of the monitor operators.

7.1 Syntax

datatype *'a monitor* =
mFIRST-d 'a pitl ((*FIRST* -))
| *mUPTO-d 'a monitor 'a monitor* ((- *UPTO* -) [*84*,*84*] *83*)
| *mTHRU-d 'a monitor 'a monitor* ((- *THRU* -) [*84*,*84*] *83*)
| *mTHEN-d 'a monitor 'a monitor* ((- *THEN* -) [*84*,*84*] *83*)
| *mWITH-d 'a monitor 'a pitl* ((- *WITH* -) [*84*,*84*] *83*)

7.2 Derived Monitors

definition *mHALT-d* ((*HALT* -) [*84*] *83*)
where

HALT w ≡ *FIRST* (*fin* (*init w*))

definition *mLEN-d* ((*LEN* -) [*84*] *83*)
where
LEN k ≡ *FIRST* (*len k*)

definition *mEMPTY-d* (*EMPTY*)
where
EMPTY ≡ *FIRST empty*

definition *mSKIP-d* (*SKIP*)
where
SKIP ≡ *FIRST skip*

definition *mGUARD-d* ((*GUARD* -) [*84*] *83*)
where
GUARD w ≡ *EMPTY WITH* ((*init w*))

definition *mFAIL-d* (*FAIL*)
where
FAIL ≡ *GUARD false*$_i$

primrec *mTIMES-d* :: *'a monitor* ⇒ *nat* ⇒ *'a monitor* ((- *TIMES* -) [*84*,*84*] *83*)
where
mTIMES-0 : *a TIMES 0* = *EMPTY*
| *mTIMES-Suc*: *a TIMES* (*Suc k*) = *a THEN* (*a TIMES k*)

definition *mALWAYS-d* ((- *ALWAYS* -) [*84*,*84*] *83*)
where
a ALWAYS (*w*) ≡ *a WITH* (*bi* (*fin* (*init w*)))

definition *mSOMETIME-d* ((- *SOMETIME* -) [*84*,*84*] *83*)
where
a SOMETIME (*w*) ≡ *a WITH* (*di* (*fin* (*init w*)))

definition *mlimit-d* ((*Limit* -) [*84*] *83*)
where
Limit f ≡ (*bs* (¬$_i$ *f*))

definition *mWITHIN-d* ((- *WITHIN* -) [*84*,*84*] *83*)
where
a WITHIN (*f*) ≡ *a WITH* (*Limit f*)

definition *mUNTIL-d* ((- *UNTIL* -) [*84*,*84*] *83*)
where
w1 UNTIL w2 ≡ (*HALT w2*) *WITH* (*bm w1*)

7.3 Semantics

fun *semantics-monitor* :: *'a monitor* ⇒ *'a pitl* (($\mathcal{M}$ -) [*80*] *80*)

where
$(\mathcal{M}\ (FIRST\ a)) = \triangleright\ a$
$|\ (\mathcal{M}\ (a\ UPTO\ b)) = \triangleright(\ (\mathcal{M}\ a) \vee_i (\mathcal{M}\ b))$
$|\ (\mathcal{M}\ (a\ THRU\ b)) = \triangleright(\ di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))$
$|\ (\mathcal{M}\ (a\ THEN\ b)) = ((\mathcal{M}\ a);(\mathcal{M}\ b))$
$|\ (\mathcal{M}\ (a\ WITH\ f)) = (\ (\mathcal{M}\ a) \wedge_i f\ )$

definition *eq-d* :: *'a monitor* ⇒ *'a monitor* ⇒ *bool* ((- $\simeq$ -) [*84*,*84*] *83*)
where
eq-d a b $\equiv (\vdash (\mathcal{M}\ a) \equiv_i (\mathcal{M}\ b))$

lemma *MonEqRefl*:
$a \simeq a$
by (*simp add*: *eq-d-def*)

lemma *MonEqSym*:
assumes $a \simeq b$
shows $b \simeq a$
using *assms eq-d-def itl-prop*(*30*) **by** *blast*

lemma *MonEqTrans*:
assumes $a \simeq b$
$b \simeq c$
shows $a \simeq c$
using *assms*(*1*) *assms*(*2*) **using** *eq-d-def prop03* **by** *blast*

lemma *MonEq*:
$(a \simeq b) = (\vdash (\mathcal{M}\ a) \equiv_i (\mathcal{M}\ b))$
by (*simp add*: *eq-d-def*)

lemma *MonEqSubstWith*:
assumes $a \simeq b$
shows $(a\ WITH\ f) \simeq (b\ WITH\ f)$
using *assms* **by** (*simp add*: *eq-d-def*)

lemma *MonEqSubstThen*:
assumes $a1 \simeq b1$
$a2 \simeq b2$
shows $(a1\ THEN\ a2) \simeq (b1\ THEN\ b2)$
using *assms*(*1*) *assms*(*2*) **by** (*simp add*: *eq-d-def*)

lemma *MonEqSubstUpto*:
assumes $a1 \simeq b1$
$a2 \simeq b2$
shows $(a1\ UPTO\ a2) \simeq (b1\ UPTO\ b2)$
using *assms*(*1*) *assms*(*2*)
proof −
have *1*: $a1 \simeq b1$ **using** *assms*(*1*) **by** *blast*
have *2*: $a2 \simeq b2$ **using** *assms*(*2*) **by** *blast*

```
 have 3: ((a1 UPTO a2) ≃ (b1 UPTO b2)) =
         (⊢ (𝓜 a1 UPTO a2) ≡ᵢ (𝓜 b1 UPTO b2)) by (simp add: eq-d-def)
 have 4: ⊢  (𝓜 a1 UPTO a2) ≡ᵢ ▷( (𝓜 a1) ∨ᵢ (𝓜 a2)) by simp
 have 5: ⊢ (𝓜 b1 UPTO b2) ≡ᵢ ▷( (𝓜 b1) ∨ᵢ (𝓜 b2)) by simp
 have 6: ⊢ ( (𝓜 a1) ∨ᵢ (𝓜 a2)) ≡ᵢ ( (𝓜 b1) ∨ᵢ (𝓜 b2)) using 1 2 eq-d-def by auto
 have 7: ⊢ ▷( (𝓜 a1) ∨ᵢ (𝓜 a2)) ≡ᵢ ▷( (𝓜 b1) ∨ᵢ (𝓜 b2)) using 6 FstEqvRule by blast
 have 8:  (⊢ (𝓜 a1 UPTO a2) ≡ᵢ (𝓜 b1 UPTO b2)) using 7 6 5 by auto
 from 3 7 show ?thesis by auto
qed

lemma MonEqSubstThru:
 assumes a1 ≃ b1
        a2 ≃ b2
 shows   (a1 THRU a2) ≃ (b1 THRU b2)
using assms(1) assms(2)
proof −
 have 1: a1 ≃ b1
      using assms(1) by blast
 have 2: a2 ≃ b2
      using assms(2) by blast
 have 3: ((a1 THRU a2) ≃ (b1 THRU b2)) =
         (⊢ (𝓜 a1 THRU a2) ≡ᵢ (𝓜 b1 THRU b2))
      by (simp add: eq-d-def)
 have 4: ⊢  (𝓜 a1 THRU a2) ≡ᵢ ▷( di(𝓜 a1) ∧ᵢ di(𝓜 a2))
      by simp
 have 5: ⊢ (𝓜 b1 THRU b2) ≡ᵢ ▷( di(𝓜 b1) ∧ᵢ di(𝓜 b2))
      by simp
 have 6: ⊢ ( (𝓜 a1) ∧ᵢ (𝓜 a2)) ≡ᵢ ( (𝓜 b1) ∧ᵢ (𝓜 b2))
      using 1 2 eq-d-def by auto
 have 7: ⊢ ( di(𝓜 a1) ∧ᵢ di(𝓜 a2)) ≡ᵢ ( di(𝓜 b1) ∧ᵢ di(𝓜 b2))
      using 6 by (meson 1 2 DiEqvDi eq-d-def itl-prop(31) prop22)
 have 8: ⊢ ▷( di(𝓜 a1) ∧ᵢ di(𝓜 a2)) ≡ᵢ ▷( di(𝓜 b1) ∧ᵢ di(𝓜 b2))
      using 7 FstEqvRule by blast
 have 9:  (⊢ (𝓜 a1 THRU a2) ≡ᵢ (𝓜 b1 THRU b2))
      using 8 5 4 by auto
 from 3 9 show ?thesis by auto
qed

definition mAND-d ((- AND -) [84,84] 83)
where
  a AND b ≡ a WITH (𝓜 b)

definition mITERATE-d ((- ITERATE -) [84,84] 83)
where
  a ITERATE b ≡ a WITH (𝓜 b)⋆

definition mSTAR-d ((- STAR -) [84,84] 83)
where
 a STAR f ≡ (FIRST(◇ f)) ITERATE (a)
```

```
definition mREPEAT-d ((- REPEATUNTIL -) [84,84] 83)
where
  a REPEATUNTIL w ≡ ( (HALT w) ITERATE (a WITH (keep(¬ᵢ (init w)))))
```

7.4 Monitor Laws

```
lemma MFixFst:
⊢ (ℳ a) ≡ᵢ ▷ (ℳ a)
proof
 (induct a )
 case (mFIRST-d x)
 then show ?case
  proof −
   have 1: ⊢ (ℳ FIRST x) ≡ᵢ ▷ x  by simp
   have 2: ⊢ ▷ x ≡ᵢ ▷ (▷ x) using FstFixFst by auto
   have 3: ⊢  ▷ (▷ x) ≡ᵢ ▷(ℳ FIRST x) by simp
   from 1 2 3 show ?thesis by auto
  qed
 next
  case (mUPTO-d a1 a2)
  then show ?case
  proof −
  have 1: ⊢ (ℳ (a1 UPTO a2)) ≡ᵢ ▷( (ℳ a1) ∨ᵢ (ℳ a2)) by simp
  have 2: ⊢  ▷( (ℳ a1) ∨ᵢ (ℳ a2)) ≡ᵢ ▷(▷((ℳ a1) ∨ᵢ (ℳ a2))) using FstFixFst by auto
  have 3: ⊢ ▷(▷((ℳ a1) ∨ᵢ (ℳ a2))) ≡ᵢ ▷(ℳ (a1 UPTO a2)) by simp
  from 1 2 3 show ?thesis by auto
 qed
next
  case (mTHRU-d a1 a2)
  then show ?case
  proof −
   have 1: ⊢ (ℳ (a1 THRU a2)) ≡ᵢ ▷( di (ℳ a1) ∧ᵢ di(ℳ a2)) by simp
   have 2: ⊢ ▷( di (ℳ a1) ∧ᵢ di(ℳ a2)) ≡ᵢ ▷(▷(di(ℳ a1) ∧ᵢ di(ℳ a2))) using FstFixFst by auto
   have 3: ⊢ ▷(▷( di (ℳ a1) ∧ᵢ di(ℳ a2))) ≡ᵢ ▷(ℳ (a1 THRU a2)) by simp
   from 1 2 3 show ?thesis by auto
 qed
next
  case (mTHEN-d a1 a2)
  then show ?case
  proof −
   have 1: ⊢ (ℳ (a1 THEN a2)) ≡ᵢ (ℳ a1) ; (ℳ a2)
        by simp
   have 2: ⊢ (ℳ a1) ; (ℳ a2) ≡ᵢ ▷(ℳ a1) ; ▷(ℳ a2)
        using ChopEqvChop mTHEN-d.hyps(1) mTHEN-d.hyps(2) by blast
   have 3: ⊢ ▷(ℳ a1) ; ▷(ℳ a2) ≡ᵢ ▷(▷(ℳ a1) ; (ℳ a2))
        using FstFstChopEqvFstChopFst itl-prop(30) by blast
   have 4: ⊢ ▷(▷(ℳ a1) ; (ℳ a2)) ≡ᵢ ▷((ℳ a1) ; (ℳ a2))
        using FstEqvRule LeftChopEqvChop itl-prop(30) mTHEN-d.hyps(1) by blast
   have 5: ⊢ ▷((ℳ a1) ; (ℳ a2)) ≡ᵢ ▷(ℳ (a1 THEN a2))
        by simp
   from 1 2 3 4 5 show ?thesis by auto
  qed
next
  case (mWITH-d a x2)
  then show ?case
  proof −
   have 1: ⊢ (M (a WITH x2)) ≡_i (M a) ∧_i ( x2)
        by simp
   have 2: ⊢ (M a) ∧_i ( x2) ≡_i ▷(M a) ∧_i ( x2)
        using mWITH-d.hyps by auto
   have 3: ⊢ ▷(M a) ∧_i ( x2) ≡_i ▷(▷(M a) ∧_i ( x2))
        using FstFstAndEqvFstAnd itl-prop(30) by blast
   have 4: ⊢ ▷(▷(M a) ∧_i ( x2)) ≡_i ▷((M a) ∧_i ( x2))
        using 2 FstEqvRule itl-prop(30) by blast
   have 5: ⊢ ▷((M a) ∧_i ( x2)) ≡_i ▷(M (a WITH x2))
        by simp
   from 1 2 3 4 5 show ?thesis by auto
 qed
qed

lemma MGuardFalseEqvFalse:
 ⊢ M (GUARD false_i) ≡_i false_i
proof −
  have 1: ⊢ M(GUARD false_i) ≡_i M(EMPTY WITH (init false_i)) by (simp add: mGUARD-d-def)
  have 2: ⊢ M(EMPTY WITH (init false_i)) ≡_i M(EMPTY) ∧_i (init false_i) by simp
  have 3: ⊢ false_i ≡_i (init false_i) by simp
  have 4: ⊢ M(EMPTY) ∧_i (init false_i) ≡_i M(EMPTY) ∧_i false_i using 3 by simp
  have 5: ⊢ M(EMPTY) ∧_i false_i ≡_i false_i by simp
  have 6: ⊢M(EMPTY) ∧_i (init false_i) ≡_i false_i using 4 5 by simp
  have 7: ⊢M(EMPTY WITH (init false_i)) ≡_i false_i using 2 6 by simp
  have 8: ⊢M(GUARD false_i) ≡_i false_i using 1 7 by simp
  from 8 show ?thesis by auto
qed

lemma MFirstFalseEqvFalse:
 ⊢ M (FIRST false_i) ≡_i false_i
proof −
  have 1: ⊢ M(FIRST false_i) ≡_i ▷ false_i by simp
  have 2:  ⊢ M(FIRST false_i) ≡_i false_i using FstFalse by simp
 from 2 show ?thesis by auto
qed

lemma MFailAlt:
 ⊢ M FAIL ≡_i false_i
proof −
  have 1: ⊢ M FAIL ≡_i M (GUARD (false_i))  by (simp add: mFAIL-d-def)
  have 2: ⊢ M (GUARD (false_i)) ≡_i false_i using MGuardFalseEqvFalse by auto
 from 1 2 show ?thesis by auto
qed
```

```
lemma MFailEqvFirstFalseWithinEmpty:
  (FAIL) ≃ ((FIRST false_i) WITHIN ( empty ))
proof -
  have 1: ⊢ ℳ( (FIRST false_i) WITHIN ( empty )) ≡_i ℳ ((FIRST false_i) WITH (Limit empty) )
      by (simp add: mWITHIN-d-def)
  have 2: ⊢ ℳ ((FIRST false_i) WITH (Limit empty) ) ≡_i ℳ (FIRST false_i) ∧_i (Limit empty )
      by simp
  have 3: ⊢ ℳ ((FIRST false_i) WITH (Limit empty) ) ≡_i false_i
      using MFirstFalseEqvFalse by auto
  have 4: ⊢ ℳ( (FIRST false_i) WITHIN ( empty )) ≡_i false_i
      using 1 3 by auto
  have 5: ⊢ (ℳ FAIL) ≡_i false_i
      using MFailAlt by simp
 from 4 5 show ?thesis by (metis eq-d-def itl-prop(30) prop21)
 qed

lemma MEmptyAlt:
 ⊢ (ℳ EMPTY) ≡_i empty
proof -
  have 1: ⊢ (ℳ EMPTY) ≡_i (ℳ (FIRST empty)) by (simp add: mEMPTY-d-def)
  have 2: ⊢ (ℳ (FIRST empty)) ≡_i ▷ empty by simp
  have 3: ⊢ ▷ empty ≡_i empty using FstEmpty by auto
 from 1 2 3 show ?thesis by auto
qed

lemma MSkipAlt:
 ⊢ ℳ SKIP ≡_i skip
proof -
  have 1: ⊢ ℳ SKIP ≡_i ℳ(FIRST skip) by (simp add: mSKIP-d-def)
  have 2: ⊢ ℳ(FIRST skip) ≡_i ▷ skip by simp
  have 3: ⊢ ▷ skip ≡_i skip using FstSkip by simp
 from 1 2 3 show ?thesis by auto
qed

lemma MGuardAlt:
 ⊢ ℳ (GUARD(w)) ≡_i empty ∧_i init w
proof -
  have 1: ⊢ ℳ (GUARD(w)) ≡_i ℳ(EMPTY WITH ( (init w))) by (simp add:mGUARD-d-def)
  have 2: ⊢ ℳ(EMPTY WITH ( (init w))) ≡_i (ℳ EMPTY) ∧_i ( init w) by simp
  have 3: ⊢ (ℳ EMPTY) ∧_i ( init w) ≡_i empty ∧_i ( init w) using MEmptyAlt prop06 by blast
  have 4: ⊢  empty ∧_i ( init w) ≡_i empty ∧_i init w by simp
 from 1 2 3 4 show ?thesis by auto
qed

lemma MLengthAlt:
 ⊢ℳ(LEN(k)) ≡_i len(k)
proof -
  have 1: ⊢ ℳ(LEN(k)) ≡_i ℳ(FIRST(len(k))) by (simp add:mLEN-d-def)
  have 2: ⊢ ℳ(FIRST(len(k))) ≡_i ▷(len(k)) by simp
  have 3: ⊢ ▷(len(k)) ≡_i len(k) using FstLenEqvLen by blast
 from 1 2 3 show ?thesis by auto
qed

lemma MAlwaysAlt:
 ⊢ 𝓜(a ALWAYS w) ≡_i 𝓜(a) ∧_i □ (init w)
proof −
  have 1: ⊢ 𝓜(a ALWAYS w) ≡_i 𝓜(a WITH (bi (fin (init w))))
      by (simp add: mALWAYS-d-def)
  have 2: ⊢ 𝓜(a WITH (bi (fin (init w)))) ≡_i 𝓜(a) ∧_i (bi (fin (init w)))
      by simp
  have 3: ⊢ 𝓜(a) ∧_i (bi (fin (init w))) ≡_i 𝓜(a) ∧_i □ (init w)
      using BoxStateEqvBiFinState by auto
  from 1 2 3 show ?thesis by simp
qed

lemma MSometimeAlt:
⊢ 𝓜(a SOMETIME w) ≡_i 𝓜(a) ∧_i ◇ (init w)
proof −
  have 1: ⊢ 𝓜(a SOMETIME w) ≡_i 𝓜(a WITH (di (fin (init w))))
      by (simp add: mSOMETIME-d-def)
  have 2: ⊢𝓜(a WITH (di (fin (init w)))) ≡_i 𝓜(a) ∧_i (di (fin (init w)))
      by simp
  have 3: ⊢𝓜(a WITH (di (fin (init w)))) ≡_i 𝓜(a) ∧_i  ◇ (init w)
      using DiamondStateEqvDiFinState by auto
  from 1 2 3 show ?thesis by simp
qed

lemma MWithinAlt:
⊢ 𝓜(a WITHIN f) ≡_i 𝓜(a) ∧_i (bs (¬_i f))
proof −
  have 1: ⊢ 𝓜(a WITHIN f) ≡_i 𝓜(a WITH (bs (¬_i f)))
      by (simp add: mWITHIN-d-def mlimit-d-def)
  have 2: ⊢ 𝓜(a WITH (bs (¬_i f))) ≡_i 𝓜(a) ∧_i (bs (¬_i f))
      by simp
  from 1 2 show ?thesis by simp
qed

lemma MTimesAlt:
⊢ 𝓜(a TIMES k) ≡_i power (𝓜 a) k
proof
 (induct k)
 case 0
 then show ?case
  proof −
   have 1: ⊢ 𝓜  a TIMES 0 ≡_i 𝓜 EMPTY by simp
   have 2: ⊢ 𝓜 EMPTY ≡_i empty  using MEmptyAlt by simp
   have 3: ⊢ empty ≡_i power (𝓜 a) 0 by simp
   from 1 2 3 show ?thesis by auto
  qed
 next
```

case (*Suc k*)

then show *?case*

proof −

have *1*: $\vdash \mathcal{M}\ \ a\ TIMES\ Suc\ k \equiv_i \mathcal{M}(a\ THEN\ (a\ TIMES\ k)\ )$

by *simp*

have *2*: $\vdash \mathcal{M}(a\ THEN\ (a\ TIMES\ k)\ ) \equiv_i (\mathcal{M}\ a);(\mathcal{M}(a\ TIMES\ k))$

by *simp*

have *3*: $\vdash (\mathcal{M}\ a);(\mathcal{M}(a\ TIMES\ k)) \equiv_i (\mathcal{M}\ a);(power\ (\mathcal{M}\ a)\ k)$

using *RightChopEqvChop Suc.hyps* **by** *blast*

have *4*: $\vdash (\mathcal{M}\ a);(power\ (\mathcal{M}\ a)\ k) \equiv_i power\ (\mathcal{M}\ a)\ (Suc\ k)$

by *simp*

from *1 2 3 4* **show** *?thesis* **by** *auto*

qed

qed

lemma *MUptoAlt*:

$\vdash \mathcal{M}(a\ UPTO\ b) \equiv_i (\ (\mathcal{M}\ a) \wedge_i bi\ \neg_i(\mathcal{M}\ b)) \vee_i ((\mathcal{M}\ b) \wedge_i bi\ \neg_i(\mathcal{M}\ a)) \vee_i ((\mathcal{M}\ a) \wedge_i (\mathcal{M}\ b))$

proof −

have *1*: $\vdash \mathcal{M}(a\ UPTO\ b) \equiv_i \triangleright((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b))$

by *simp*

have *2*: $\vdash \triangleright((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b)) \equiv_i (\triangleright(\mathcal{M}\ a) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ b))) \vee_i (\triangleright(\mathcal{M}\ b) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ a)))$

using *FstWithOrEqv* **by** *blast*

have *3*: $\vdash (\triangleright(\mathcal{M}\ a) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ b))) \vee_i (\triangleright(\mathcal{M}\ b) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ a))) \equiv_i$

$((\mathcal{M}\ a) \wedge_i ((\mathcal{M}\ b) \vee_i \neg_i(\mathcal{M}\ b)) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ b))) \vee_i$

$((\mathcal{M}\ b) \wedge_i ((\mathcal{M}\ a) \vee_i \neg_i(\mathcal{M}\ a)) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ a)))$

using *MFixFst* **by** *auto*

have *4*: $\vdash ((\mathcal{M}\ a) \wedge_i ((\mathcal{M}\ b) \vee_i \neg_i(\mathcal{M}\ b)) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ b))) \vee_i$

$((\mathcal{M}\ b) \wedge_i ((\mathcal{M}\ a) \vee_i \neg_i(\mathcal{M}\ a)) \wedge_i\ (bs\ \neg_i(\mathcal{M}\ a))) \equiv_i$

$((\mathcal{M}\ a) \wedge_i (\ ((\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)) \vee_i (\neg_i(\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)))\ ) \vee_i$

$((\mathcal{M}\ b) \wedge_i (\ ((\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)) \vee_i (\neg_i(\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)))\ )$

by *auto*

have *5*: $\vdash ((\mathcal{M}\ a) \wedge_i (\ ((\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)) \vee_i (\neg_i(\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)))\ ) \vee_i$

$((\mathcal{M}\ b) \wedge_i (\ ((\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)) \vee_i (\neg_i(\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)))\ ) \equiv_i$

$((\mathcal{M}\ a) \wedge_i (\ (\triangleright(\mathcal{M}\ b)) \vee_i (\neg_i(\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)))\ ) \vee_i$

$((\mathcal{M}\ b) \wedge_i (\ (\triangleright(\mathcal{M}\ a)) \vee_i (\neg_i(\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)))\ )$

by (*simp add*: *first-d-def*)

have *6*: $\vdash ((\mathcal{M}\ a) \wedge_i (\ (\triangleright(\mathcal{M}\ b)) \vee_i (\neg_i(\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)))\ ) \vee_i$

$((\mathcal{M}\ b) \wedge_i (\ (\triangleright(\mathcal{M}\ a)) \vee_i (\neg_i(\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)))\ ) \equiv_i$

$((\mathcal{M}\ a) \wedge_i (\ ((\mathcal{M}\ b)) \vee_i (\neg_i(\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)))\ ) \vee_i$

$((\mathcal{M}\ b) \wedge_i (\ ((\mathcal{M}\ a)) \vee_i (\neg_i(\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)))\ )$

using *MFixFst* **by** *auto*

have *7*: $\vdash (\neg_i(\mathcal{M}\ b) \wedge_i bs\neg_i(\mathcal{M}\ b)) \equiv_i bi(\neg_i(\mathcal{M}\ b))$

using *AndBsEqvBi* **by** *blast*

have *8*: $\vdash (\neg_i(\mathcal{M}\ a) \wedge_i bs\neg_i(\mathcal{M}\ a)) \equiv_i bi(\neg_i(\mathcal{M}\ a))$

using *AndBsEqvBi* **by** *blast*

have *9*: $\vdash ((\mathcal{M}\ a) \wedge_i (\ ((\mathcal{M}\ b)) \vee_i ((\neg_i(\mathcal{M}\ b)) \wedge_i bs(\neg_i(\mathcal{M}\ b))))\ ) \vee_i$

$((\mathcal{M}\ b) \wedge_i (\ ((\mathcal{M}\ a)) \vee_i ((\neg_i(\mathcal{M}\ a)) \wedge_i bs(\neg_i(\mathcal{M}\ a))))\ ) \equiv_i$

$((\mathcal{M}\ a) \wedge_i (\ ((\mathcal{M}\ b)) \vee_i (\ bi(\neg_i(\mathcal{M}\ b))))\ ) \vee_i$

$((\mathcal{M}\ b) \wedge_i (\ ((\mathcal{M}\ a)) \vee_i (\ bi(\neg_i(\mathcal{M}\ a))))\ )$

using *7 8* **by** *auto*

have *10*: $\vdash((\mathcal{M}\ a) \wedge_i (\ ((\mathcal{M}\ b)) \vee_i (\ bi(\neg_i(\mathcal{M}\ b))))\ ) \vee_i$
$((\mathcal{M}\ b) \wedge_i (\ ((\mathcal{M}\ a)) \vee_i (\ bi(\neg_i(\mathcal{M}\ a))))\ ) \equiv_i$
$((\ (\mathcal{M}\ a) \wedge_i (\mathcal{M}\ b)) \vee_i (\ (\mathcal{M}\ a) \wedge_i bi(\neg_i(\mathcal{M}\ b)))) \vee_i$
$((\ (\mathcal{M}\ b) \wedge_i (\mathcal{M}\ a)) \vee_i (\ (\mathcal{M}\ b) \wedge_i bi(\neg_i(\mathcal{M}\ a))))$
by *auto*
have *11*: $\vdash ((\ (\mathcal{M}\ a) \wedge_i (\mathcal{M}\ b)) \vee_i (\ (\mathcal{M}\ a) \wedge_i bi(\neg_i(\mathcal{M}\ b)))) \vee_i$
$((\ (\mathcal{M}\ b) \wedge_i (\mathcal{M}\ a)) \vee_i (\ (\mathcal{M}\ b) \wedge_i bi(\neg_i(\mathcal{M}\ a)))) \equiv_i$
$(\ (\mathcal{M}\ a) \wedge_i bi\ \neg_i(\mathcal{M}\ b)) \vee_i ((\mathcal{M}\ b) \wedge_i bi\ \neg_i(\mathcal{M}\ a)) \vee_i ((\mathcal{M}\ a) \wedge_i (\mathcal{M}\ b))$
by *auto*
from *1 2 3 4 5 6 9 10 11* **show** *?thesis* **by** *auto*
qed

lemma *MThruAlt*:
$\vdash \mathcal{M}(a\ \mathit{THRU}\ b) \equiv_i ((\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i ((\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ a))$
proof −
have *1*: $\vdash \mathcal{M}(a\ \mathit{THRU}\ b) \equiv_i \rhd(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))$
by *simp*
have *2*: $\vdash \rhd(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \equiv_i (\rhd(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i (\rhd(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ a))$
using *FstDiAndDiEqv* **by** *auto*
have *3*: $\vdash (\rhd(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i (\rhd(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ a)) \equiv_i$
$((\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i ((\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ a))$
using *MFixFst* **by** *auto*
from *1 2 3* **show** *?thesis* **by** *auto*
qed

lemma *MHaltAlt*:
$\vdash \mathcal{M}(\mathit{HALT}\ w) \equiv_i halt(init\ w)$
proof −
have *1*: $\vdash \mathcal{M}(\mathit{HALT}\ w) \equiv_i \mathcal{M}(\mathit{FIRST}\ (fin\ (init\ \ w)))$ **by** (*simp add*: *mHALT-d-def*)
have *2*: $\vdash \mathcal{M}(\mathit{FIRST}\ (fin\ (init\ \ w))) \equiv_i \rhd\ (fin\ (init\ \ w))$ **by** *simp*
have *3*: $\vdash \rhd\ (fin\ (init\ \ w)) \equiv_i halt(init\ w)$ **using** *HaltStateEqvFstFinState* **by** *auto*
from *1 2 3* **show** *?thesis* **by** *simp*
qed

lemma *MFailUpto*:
$(\ (\mathit{FAIL}\ \mathit{UPTO}\ a)) \simeq (a)$
proof −
have *1*: $\vdash (\mathcal{M}\ (\mathit{FAIL}\ \mathit{UPTO}\ a)) \equiv_i \rhd((\mathcal{M}\ \mathit{FAIL}) \vee_i (\mathcal{M}\ a))$ **by** *simp*
have *2*: $\vdash \mathcal{M}\ \mathit{FAIL} \vee_i \mathcal{M}\ a \equiv_i false_i \vee_i \mathcal{M}\ a$ **using** *MFailAlt* **by** *auto*
have *3*: $\vdash \rhd((\mathcal{M}\ \mathit{FAIL}) \vee_i (\mathcal{M}\ a)) \equiv_i \rhd(false_i \vee_i (\mathcal{M}\ a))$ **using** *2 FstEqvRule* **by** *blast*
have *4*: $\vdash false_i \vee_i (\mathcal{M}\ a) \equiv_i \mathcal{M}\ a$ **by** *simp*
have *5*: $\vdash \rhd(false_i \vee_i (\mathcal{M}\ a)) \equiv_i\ \rhd(\mathcal{M}\ a)$ **using** *4 FstEqvRule* **by** *blast*
have *6*: $\vdash \rhd(\mathcal{M}\ a) \equiv_i \mathcal{M}\ a$ **using** *MFixFst* **by** *auto*
from *1 2 3 4 5 6* **show** *?thesis* **by** (*simp add*: *eq-d-def*)
qed

lemma *MFailThru*:
$(\mathit{FAIL}\ \mathit{THRU}\ a) \simeq\ \mathit{FAIL}$
proof −
have *1*: $\vdash \mathcal{M}(\mathit{FAIL}\ \mathit{THRU}\ a) \equiv_i \rhd(di\ (\mathcal{M}\ \mathit{FAIL}) \wedge_i di(\mathcal{M}\ a))$

by *simp*
have *2*: $\vdash \triangleright(di\ (\mathcal{M}\ FAIL) \wedge_i di(\mathcal{M}\ a)) \equiv_i \triangleright(di\ (false_i) \wedge_i di(\mathcal{M}\ a))$
using *MFailAlt DiEqvDi FstEqvRule prop06* **by** *blast*
have *3*: $\vdash di\ false_i \equiv_i false_i$
by *simp*
hence *4*: $\vdash \triangleright(di\ (false_i) \wedge_i di(\mathcal{M}\ a)) \equiv_i \triangleright(\ (false_i) \wedge_i di(\mathcal{M}\ a))$
using *FstEqvRule prop06* **by** *blast*
have *5*: $\vdash \triangleright(\ (false_i) \wedge_i di(\mathcal{M}\ a)) \equiv_i \triangleright false_i$
using *FstEqvRule itl-prop*(*19*) **by** *blast*
have *6*: $\vdash \triangleright false_i \equiv_i false_i$ **using** *FstFalse*
by *auto*
have *7*: $\vdash false_i \equiv_i \mathcal{M}\ FAIL$
using *MFailAlt* **by** *auto*
from *1 2 4 5 6 7* **show** *?thesis* **by** (*metis eq-d-def prop03*)
qed

lemma *MFailAnd*:
$(FAIL\ AND\ a) \simeq\ FAIL$
proof −
have *1*: $\vdash \mathcal{M}\ (FAIL\ AND\ a) \equiv_i (\mathcal{M}\ FAIL) \wedge_i (\mathcal{M}\ a)$ **by** (*simp add*: *mAND-d-def*)
have *2*: $\vdash (\mathcal{M}\ FAIL) \wedge_i (\mathcal{M}\ a) \equiv_i false_i \wedge_i (\mathcal{M}\ a)$ **using** *MFailAlt* **by** *auto*
have *3*: $\vdash false_i \wedge_i (\mathcal{M}\ a) \equiv_i false_i$ **by** *auto*
have *4*: $\vdash \mathcal{M}\ (FAIL\ AND\ a) \equiv_i false_i$ **using** *1 2 3* **by** *auto*
have *5*: $\vdash false_i \equiv_i \mathcal{M}\ FAIL$ **using** *MFailAlt* **by** *auto*
from *1 2 3 4 5* **show** *?thesis* **by** (*metis eq-d-def itl-prop*(*30*) *prop21*)
qed

lemma *MThenFail*:
$(a\ THEN\ FAIL) \simeq\ FAIL$
proof −
have *1*: $\vdash \mathcal{M}\ (a\ THEN\ FAIL) \equiv_i (\mathcal{M}\ a);(\mathcal{M}\ FAIL)$ **by** *simp*
have *2*: $\vdash (\mathcal{M}\ a);(\mathcal{M}\ FAIL) \equiv_i (\mathcal{M}\ a);false_i$ **using** *MFailAlt* **by** *auto*
have *3*: $\vdash (\mathcal{M}\ a);false_i \equiv_i false_i$ **by** *auto*
have *4*: $\vdash false_i \equiv_i \mathcal{M}\ FAIL$ **using** *MFailAlt* **by** *auto*
from *1 2 3 4* **show** *?thesis* **by** (*metis eq-d-def itl-prop*(*30*) *prop21*)
qed

lemma *MFailThen*:
$(FAIL\ THEN\ a) \simeq FAIL$
proof −
have *1*: $\vdash \mathcal{M}\ (FAIL\ THEN\ a) \equiv_i (\mathcal{M}\ FAIL);(\mathcal{M}\ a)$ **by** *simp*
have *2*: $\vdash (\mathcal{M}\ FAIL);(\mathcal{M}\ a) \equiv_i false_i;(\mathcal{M}\ a)$ **using** *MFailAlt* **by** *auto*
have *3*: $\vdash false_i;(\mathcal{M}\ a) \equiv_i false_i$ **by** *auto*
have *4*: $\vdash false_i \equiv_i \mathcal{M}\ FAIL$ **using** *MFailAlt* **by** *auto*
from *1 2 3 4* **show** *?thesis* **by** (*metis eq-d-def itl-prop*(*30*) *prop21*)
qed

lemma *MFailWith*:
$(FAIL\ WITH\ f) \simeq\ FAIL$
proof −

have *1*: $\vdash \mathcal{M}$ (*FAIL WITH f*) $\equiv_i$ ($\mathcal{M}$ *FAIL*) $\wedge_i$ *f* **by** *simp*
have *2*: $\vdash$ ($\mathcal{M}$ *FAIL*) $\wedge_i$ *f* $\equiv_i$ *false*$_i$ $\wedge_i$ *f* **using** *MFailAlt* **by** *auto*
have *3*: $\vdash$ *false*$_i$ $\wedge_i$ *f* $\equiv_i$ *false*$_i$ **by** *simp*
have *4*: $\vdash$ *false*$_i$ $\equiv_i$ $\mathcal{M}$ *FAIL* **using** *MFailAlt* **by** *auto*
from *1 2 3 4* **show** *?thesis* **by** (*metis eq-d-def itl-prop*(*30*) *prop21*)
qed

lemma *MWithFalse*:
(*a WITH* ((*false*$_i$))) $\simeq$ *FAIL*
proof −
have *1*: $\vdash \mathcal{M}$ (*a WITH* (*false*$_i$)) $\equiv_i$ (($\mathcal{M}$ *a*) $\wedge_i$ *false*$_i$) **by** *simp*
have *2*: $\vdash$ (($\mathcal{M}$ *a*) $\wedge_i$ *false*$_i$) $\equiv_i$ $\mathcal{M}$ *FAIL* **using** *MFailAlt* **by** *auto*
from *1 2* **show** *?thesis* **by** (*simp add*: *MonEq*)
qed

lemma *MWithTrue*:
(*a WITH* ((*true*$_i$))) $\simeq$ *a*
proof −
have *1*: $\vdash \mathcal{M}$ (*a WITH true*$_i$) $\equiv_i$ (($\mathcal{M}$ *a*) $\wedge_i$ *true*$_i$) **by** (*simp*)
have *2*: $\vdash$ (($\mathcal{M}$ *a*) $\wedge_i$ *true*$_i$) $\equiv_i$ $\mathcal{M}$ *a* **by** *simp*
from *1 2* **show** *?thesis* **by** (*simp add*: *MonEq*)
qed

lemma *MEmptyUpto*:
(*EMPTY UPTO a*) $\simeq$ *EMPTY*
proof −
have *1*: $\vdash \mathcal{M}$ (*EMPTY UPTO a*) $\equiv_i$ $\rhd$(($\mathcal{M}$ *EMPTY*) $\vee_i$ ($\mathcal{M}$ *a*)) **by** *simp*
have *2*: $\vdash$ ($\mathcal{M}$ *EMPTY*) $\equiv_i$ *empty* **using** *MEmptyAlt* **by** *auto*
hence *3*: $\vdash$ ($\mathcal{M}$ *EMPTY*) $\vee_i$ ($\mathcal{M}$ *a*) $\equiv_i$ *empty* $\vee_i$ ($\mathcal{M}$ *a*) **by** *auto*
hence *4*: $\vdash$ $\rhd$(($\mathcal{M}$ *EMPTY*) $\vee_i$ ($\mathcal{M}$ *a*)) $\equiv_i$ $\rhd$(*empty* $\vee_i$ ($\mathcal{M}$ *a*)) **using** *FstEqvRule* **by** *blast*
have *5*: $\vdash$ $\rhd$(*empty* $\vee_i$ ($\mathcal{M}$ *a*)) $\equiv_i$ *empty* **using** *FstEmptyOrEqvEmpty* **by** *blast*
have *6*: $\vdash$ *empty* $\equiv_i$ ($\mathcal{M}$ *EMPTY*) **using** *MEmptyAlt* **by** *auto*
from *1 4 5 6* **show** *?thesis* **by** (*simp add*: *eq-d-def*)
qed

lemma *MEmptyThru*:
(*EMPTY THRU a*) $\simeq$ (*a*)
proof −
have *1*: $\vdash \mathcal{M}$ (*EMPTY THRU a*) $\equiv_i$ $\rhd$(*di*($\mathcal{M}$ *EMPTY*) $\wedge_i$ *di*($\mathcal{M}$ *a*)) **by** *simp*
have *2*: $\vdash$ *di*($\mathcal{M}$ *EMPTY*) $\equiv_i$ *di empty* **using** *MEmptyAlt DiEqvDi* **by** *blast*
hence *3*: $\vdash$ *di*($\mathcal{M}$ *EMPTY*) $\wedge_i$ *di*($\mathcal{M}$ *a*) $\equiv_i$ *di empty* $\wedge_i$ *di*($\mathcal{M}$ *a*) **by** *auto*
hence *4*: $\vdash$ *di empty* $\wedge_i$ *di*($\mathcal{M}$ *a*) $\equiv_i$ *di*($\mathcal{M}$ *a*) **by** *auto*
have *5*: $\vdash$ *di*($\mathcal{M}$ *EMPTY*) $\wedge_i$ *di*($\mathcal{M}$ *a*) $\equiv_i$ *di*($\mathcal{M}$ *a*) **using** *3 4* **by** *auto*
hence *6*: $\vdash$ $\rhd$(*di*($\mathcal{M}$ *EMPTY*) $\wedge_i$ *di*($\mathcal{M}$ *a*)) $\equiv_i$ $\rhd$(*di*($\mathcal{M}$ *a*)) **using** *FstEqvRule* **by** *blast*
have *7*: $\vdash$ $\rhd$(*di*($\mathcal{M}$ *a*)) $\equiv_i$ $\rhd$($\mathcal{M}$ *a*) **using** *FstDiEqvFst* **by** *blast*
have *8*: $\vdash$ $\rhd$($\mathcal{M}$ *a*) $\equiv_i$ ($\mathcal{M}$ *a*) **using** *MFixFst* **by** *auto*
from *1 6 7 8* **show** *?thesis* **by** (*simp add*: *eq-d-def*)
qed

lemma *MThenEmpty*:

```
  (a THEN EMPTY) ≃ (a)
proof −
  have 1: ⊢ ℳ (a THEN EMPTY) ≡ᵢ (ℳ a);(ℳ EMPTY) by simp
  have 2: ⊢ (ℳ a);(ℳ EMPTY) ≡ᵢ (ℳ a); empty using MEmptyAlt by auto
  have 3: ⊢ (ℳ a); empty ≡ᵢ (ℳ a) using ChopEmpty by auto
 from 1 2 3 show ?thesis by (simp add: eq-d-def)
qed

lemma MEmptyThen:
  (EMPTY THEN a) ≃ (a)
proof −
  have 1: ⊢ ℳ (EMPTY THEN a) ≡ᵢ (ℳ EMPTY);(ℳ a) by simp
  have 2: ⊢ (ℳ EMPTY);(ℳ a) ≡ᵢ empty; (ℳ a) using MEmptyAlt by auto
  have 3: ⊢ empty;(ℳ a) ≡ᵢ (ℳ a) using ChopEmpty by auto
 from 1 2 3 show ?thesis by (simp add: eq-d-def)
qed

lemma MEmptyIterate:
  (EMPTY ITERATE b) ≃ (EMPTY)
proof −
  have 1: ⊢ ℳ (EMPTY ITERATE b) ≡ᵢ ℳ (EMPTY WITH (ℳ b)⋆)
       by (simp add: mITERATE-d-def)
  have 2: ⊢ ℳ (EMPTY WITH (ℳ b)⋆) ≡ᵢ ℳ EMPTY ∧ᵢ (ℳ b)⋆
       by simp
  have 3: ⊢ ℳ EMPTY ∧ᵢ (ℳ b)⋆  ≡ᵢ empty ∧ᵢ (ℳ b)⋆
       using MEmptyAlt by auto
  have 4: ⊢ empty ∧ᵢ (ℳ b)⋆  ≡ᵢ empty ∧ᵢ (empty ∨ᵢ (((ℳ b) ∧ᵢ more);(ℳ b)⋆))
       using ChopstarEqv by auto
  have 5: ⊢ empty ∧ᵢ (empty ∨ᵢ (((ℳ b) ∧ᵢ more);(ℳ b)⋆)) ≡ᵢ empty
       by auto
  have 6: ⊢ ℳ (EMPTY ITERATE b) ≡ᵢ ℳ EMPTY
       using 1 2 3 4 5 MEmptyAlt by auto
 from 6 show ?thesis by (metis eq-d-def)
qed

lemma MIterateIdemp:
  (a ITERATE a) ≃  a
proof −
  have 1: ⊢ ℳ (a ITERATE a) ≡ᵢ ℳ (a WITH (ℳ a)⋆) by (simp add: mITERATE-d-def)
  have 2: ⊢ ℳ (a WITH (ℳ a)⋆) ≡ᵢ ℳ a ∧ᵢ (ℳ a)⋆  by simp
  have 3: ⊢ ℳ a ∧ᵢ (ℳ a)⋆ ≡ᵢ ▷(ℳ a) ∧ᵢ (▷(ℳ a))⋆  using MFixFst by auto
  have 4: ⊢ ▷(ℳ a) ∧ᵢ (▷(ℳ a))⋆ ≡ᵢ ▷(ℳ a) using FstAndFstStarEqvFst by simp
  have 5: ⊢ ▷(ℳ a) ≡ᵢ ℳ a using MFixFst by auto
  from 1 2 3 4 5 show ?thesis using prop03 by (metis eq-d-def)
qed

lemma MUptoIdemp:
  (a UPTO a) ≃  a
proof −
  have 1: ⊢ ℳ (a UPTO a) ≡ᵢ ▷((ℳ a) ∨ᵢ (ℳ a)) by simp
  have 2: ⊢ ▷((𝓜 a) ∨ᵢ (𝓜 a)) ≡ᵢ ▷(𝓜 a) using FstEqvRule itl-prop(27) by blast
  have 3: ⊢ ▷(𝓜 a) ≡ᵢ (𝓜 a) using MFixFst by auto
 from 1 2 3 show ?thesis by (simp add: eq-d-def)
qed

lemma MThruIdemp:
  ( a THRU a) ≃ (a)
proof −
  have 1: ⊢ 𝓜 ( a THRU a) ≡ᵢ ▷( di(𝓜 a) ∧ᵢ di(𝓜 a)) by simp
  have 2: ⊢ ▷( di(𝓜 a) ∧ᵢ di(𝓜 a)) ≡ᵢ ▷(di (𝓜 a)) using FstEqvRule itl-prop(20) by blast
  have 3: ⊢ ▷(di (𝓜 a)) ≡ᵢ ▷(𝓜 a) using FstDiEqvFst by blast
  have 4: ⊢ ▷(𝓜 a) ≡ᵢ (𝓜 a) using MFixFst by auto
 from 1 2 3 4 show ?thesis by (simp add: eq-d-def)
qed

lemma MAndIdemp:
 (a AND a) ≃ (a)
proof −
  have 1: ⊢ 𝓜(a AND a) ≡ᵢ (𝓜 a) ∧ᵢ (𝓜 a) by (simp add: mAND-d-def)
  have 2: ⊢ (𝓜 a) ∧ᵢ (𝓜 a) ≡ᵢ (𝓜 a) by auto
 from 1 2 show ?thesis by (simp add: eq-d-def)
qed

lemma MWithIdemp:
  ( (a WITH f)  WITH f) ≃ (a WITH f)
proof −
  have 1: ⊢ 𝓜 ( (a WITH f)  WITH f) ≡ᵢ ((𝓜 a) ∧ᵢ ( f)) ∧ᵢ ( f) by simp
  have 2: ⊢ ((𝓜 a) ∧ᵢ ( f)) ∧ᵢ ( f) ≡ᵢ (𝓜 a) ∧ᵢ ( f) by auto
  have 3: ⊢ (𝓜 a) ∧ᵢ ( f) ≡ᵢ 𝓜(a WITH f) by simp
 from 1 2 3 show ?thesis by (simp add: eq-d-def)
qed

lemma MUptoCommut:
 (a UPTO b) ≃ (b UPTO a)
proof −
  have 1: ⊢ 𝓜(a UPTO b) ≡ᵢ ▷((𝓜 a) ∨ᵢ (𝓜 b)) by simp
  have 2: ⊢ ((𝓜 a) ∨ᵢ (𝓜 b)) ≡ᵢ ((𝓜 b) ∨ᵢ (𝓜 a)) by auto
 hence 3: ⊢ ▷((𝓜 a) ∨ᵢ (𝓜 b)) ≡ᵢ ▷((𝓜 b) ∨ᵢ (𝓜 a))  using FstEqvRule by blast
  have 4: ⊢ ▷((𝓜 b) ∨ᵢ (𝓜 a)) ≡ᵢ 𝓜(b UPTO a) by simp
 from 1 3 4 show ?thesis by (simp add: eq-d-def)
qed

lemma MThruCommut:
 (a THRU b) ≃ (b THRU a)
proof −
  have 1: ⊢ 𝓜(a THRU b) ≡ᵢ ▷(di(𝓜 a) ∧ᵢ di(𝓜 b)) by simp
  have 2: ⊢ (di(𝓜 a) ∧ᵢ di(𝓜 b)) ≡ᵢ (di(𝓜 b) ∧ᵢ di(𝓜 a))  by auto
 hence 3: ⊢ ▷(di(𝓜 a) ∧ᵢ di(𝓜 b)) ≡ᵢ ▷(di(𝓜 b) ∧ᵢ di(𝓜 a)) using FstEqvRule by blast
  have 4: ⊢ ▷(di(𝓜 b) ∧ᵢ di(𝓜 a)) ≡ᵢ 𝓜(b THRU a) by simp
 from 1 3 4 show ?thesis by (simp add: eq-d-def)
```

qed

lemma *MAndCommut*:
$(a\ AND\ b) \simeq (b\ AND\ a)$
proof $-$
have *1*: $\vdash \mathcal{M}(a\ AND\ b) \equiv_i (\mathcal{M}\ a) \wedge_i (\mathcal{M}\ b)$ **by** (*simp add*: *mAND-d-def*)
have *2*: $\vdash (\mathcal{M}\ a) \wedge_i (\mathcal{M}\ b) \equiv_i (\mathcal{M}\ b) \wedge_i (\mathcal{M}\ a)$ **by** *auto*
have *3*: $\vdash (\mathcal{M}\ b) \wedge_i (\mathcal{M}\ a) \equiv_i \mathcal{M}(b\ AND\ a)$ **by** (*simp add*: *mAND-d-def*)
from *1 2 3* **show** *?thesis* **by** (*simp add*: *eq-d-def*)
qed

lemma *MWithCommut*:
$((a\ WITH\ f)\ WITH\ g) \simeq ((a\ WITH\ g)\ WITH\ f)$
proof $-$
have *1*: $\vdash \mathcal{M}((a\ WITH\ f)\ WITH\ g) \equiv_i (\mathcal{M}\ a) \wedge_i (f) \wedge_i (g)$ **by** *simp*
have *2*: $\vdash (\mathcal{M}\ a) \wedge_i (f) \wedge_i (g) \equiv_i \mathcal{M}((a\ WITH\ g)\ WITH\ f)$ **by** *auto*
from *1 2* **show** *?thesis* **by** (*simp add*: *eq-d-def*)
qed

lemma *MWithAbsorp*:
$((a\ WITH\ f)\ WITH\ g) \simeq (a\ WITH\ (f \wedge_i g))$
proof $-$
have *1*: $\vdash \mathcal{M}((a\ WITH\ f)\ WITH\ g) \equiv_i (\mathcal{M}\ a) \wedge_i (f) \wedge_i (g)$ **by** *simp*
have *2*: $\vdash (\mathcal{M}\ a) \wedge_i (f) \wedge_i (g) \equiv_i (\mathcal{M}\ a) \wedge_i (f \wedge_i g)$ **by** *auto*
from *1 2* **show** *?thesis* **by** (*simp add*: *MonEq*)
qed

lemma *MUptoAssoc*:
$((a\ UPTO\ b)\ UPTO\ c) \simeq (a\ UPTO\ (b\ UPTO\ c))$
proof $-$
have *1*: $\vdash \mathcal{M}((a\ UPTO\ b)\ UPTO\ c) \equiv_i \rhd(\mathcal{M}(a\ UPTO\ b) \vee_i (\mathcal{M}\ c))$
by *simp*
have *2*: $\vdash \rhd(\mathcal{M}(a\ UPTO\ b) \vee_i (\mathcal{M}\ c)) \equiv_i \rhd(\rhd((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b)) \vee_i (\mathcal{M}\ c))$
by *simp*
have *3*: $\vdash \rhd(\rhd((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b)) \vee_i (\mathcal{M}\ c)) \equiv_i \rhd(((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b)) \vee_i (\mathcal{M}\ c))$
using *FstFstOrEqvFstOrL* **by** *blast*
have *4*: $\vdash (((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b)) \vee_i (\mathcal{M}\ c)) \equiv_i ((\mathcal{M}\ a) \vee_i ((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))$
by *auto*
hence *5*: $\vdash \rhd(((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b)) \vee_i (\mathcal{M}\ c)) \equiv_i \rhd((\mathcal{M}\ a) \vee_i ((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))$
using *FstEqvRule* **by** *blast*
have *6*: $\vdash \rhd((\mathcal{M}\ a) \vee_i ((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))) \equiv_i \rhd((\mathcal{M}\ a) \vee_i \rhd((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))$
using *FstFstOrEqvFstOrR* *itl-prop*(*30*) **by** *blast*
have *7*: $\vdash \rhd((\mathcal{M}\ a) \vee_i \rhd((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))) \equiv_i \rhd((\mathcal{M}\ a) \vee_i \mathcal{M}(b\ UPTO\ c))$
by *simp*
have *8*: $\vdash \rhd((\mathcal{M}\ a) \vee_i \mathcal{M}(b\ UPTO\ c)) \equiv_i \mathcal{M}(a\ UPTO\ (b\ UPTO\ c))$
by *simp*
from *1 2 3 5 6 7 8* **show** *?thesis* **by** (*simp add*: *eq-d-def*)
qed

lemma *MThruAssoc*:

```
 ((a THRU b) THRU c) ≃ (a THRU (b THRU c))
proof –
  have 1: ⊢ 𝓜((a THRU b) THRU c) ≡i ▷(di(▷(di(𝓜 a) ∧i di(𝓜 b))) ∧i di(𝓜 c))
       by simp
  have 2: ⊢ di(▷(di(𝓜 a) ∧i di(𝓜 b))) ≡i di((di(𝓜 a) ∧i di(𝓜 b)))
       using DiEqvDiFst itl-prop(30) by blast
  have 3: ⊢ di((di(𝓜 a) ∧i di(𝓜 b))) ≡i di(𝓜 a) ∧i di(𝓜 b)
       using DiDiAndEqvDi by blast
  have 4: ⊢ di(▷(di(𝓜 a) ∧i di(𝓜 b))) ≡i di(𝓜 a) ∧i di(𝓜 b)
       using 2 3 by auto
  hence 5: ⊢ di(▷(di(𝓜 a) ∧i di(𝓜 b)))  ∧i di(𝓜 c) ≡i di(𝓜 a) ∧i di(𝓜 b)  ∧i di(𝓜 c)
       by auto
  have 6: ⊢ di(𝓜 b)  ∧i di(𝓜 c) ≡i di (di(𝓜 b)  ∧i di(𝓜 c))
       using DiDiAndEqvDi itl-prop(30) by blast
  have 7: ⊢ di (di(𝓜 b)  ∧i di(𝓜 c)) ≡i di (▷(di(𝓜 b)  ∧i di(𝓜 c)))
       using DiEqvDiFst by blast
  have 8: ⊢ di(𝓜 b)  ∧i di(𝓜 c) ≡i di (▷(di(𝓜 b)  ∧i di(𝓜 c)))
       using 6 7 using prop03 by blast
  hence 9: ⊢ di(𝓜 a) ∧i di(𝓜 b)  ∧i di(𝓜 c) ≡i di(𝓜 a) ∧i di (▷(di(𝓜 b)  ∧i di(𝓜 c)))
       by auto
  have 10: ⊢ di(▷(di(𝓜 a) ∧i di(𝓜 b)))  ∧i di(𝓜 c) ≡i
            di(𝓜 a) ∧i di (▷(di(𝓜 b)  ∧i di(𝓜 c)))
       using 5 9 by auto
  hence 11: ⊢ ▷(di(▷(di(𝓜 a) ∧i di(𝓜 b)))  ∧i di(𝓜 c)) ≡i
            ▷(di(𝓜 a) ∧i di (▷(di(𝓜 b)  ∧i di(𝓜 c))))
       using FstEqvRule by blast
  have 12: ⊢ ▷(di(𝓜 a) ∧i di (▷(di(𝓜 b)  ∧i di(𝓜 c)))) ≡i 𝓜(a THRU (b THRU c))
       by simp
 from 1 11 12 show ?thesis by (simp add: eq-d-def)
qed

lemma MAndAssoc:
 ((a AND b) AND c) ≃ (a AND (b AND c))
proof –
  have 1: ⊢ 𝓜((a AND b) AND c) ≡i (𝓜 a) ∧i (𝓜 b) ∧i (𝓜 c) by (simp add: mAND-d-def)
  have 2: ⊢ (𝓜 a) ∧i (𝓜 b) ∧i (𝓜 c) ≡i 𝓜(a AND (b AND c)) by (simp add: mAND-d-def)
 from 1 2 show ?thesis by (simp add: eq-d-def)
qed

lemma MThenAssoc:
 ((a THEN b) THEN c) ≃ (a THEN (b THEN c))
proof –
  have 1: ⊢ 𝓜((a THEN b) THEN c) ≡i ((𝓜 a);(𝓜 b));(𝓜 c)  by simp
  have 2: ⊢ ((𝓜 a);(𝓜 b));(𝓜 c) ≡i (𝓜 a);((𝓜 b);(𝓜 c)) using ChopAssocB by blast
  have 3: ⊢ (𝓜 a);((𝓜 b);(𝓜 c)) ≡i 𝓜(a THEN (b THEN c)) by simp
 from 1 2 3 show ?thesis by (simp add: eq-d-def)
qed

lemma MUptoThruAbsorp:
 (a UPTO (a THRU b)) ≃  a
```

```
proof −
  have 1: ⊢ 𝓜(a UPTO (a THRU b)) ≡ᵢ ▷((𝓜 a) ∨ᵢ ▷(di(𝓜 a) ∧ᵢ di(𝓜 b) ))
      by simp
  have 2: ⊢ ▷((𝓜 a) ∨ᵢ ▷(di(𝓜 a) ∧ᵢ di(𝓜 b) )) ≡ᵢ
           ▷((𝓜 a) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b) ))
      using FstFstOrEqvFstOrR by auto
  have 3: ⊢  ((𝓜 a) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b) )) ≡ᵢ
            (((𝓜 a) ∨ᵢ di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)))
      by auto
  have 4: ⊢ (((𝓜 a) ∨ᵢ di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b))) ≡ᵢ
           ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)))
      using OrDiEqvDi by auto
  have 5: ⊢ ((𝓜 a) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b) )) ≡ᵢ
           ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)))
      using 3 4 by auto
  hence 6: ⊢ ▷ ((𝓜 a) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b) )) ≡ᵢ
            ▷ ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)))
      using FstEqvRule by blast
  have 7: ⊢ ▷ ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b))) ≡ᵢ
           (di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)) ∧ᵢ
           bs ¬ᵢ( (di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)))
      by (simp add: first-d-def)
  have 8: ⊢ (di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)) ≡ᵢ
           (di(𝓜 a) ∧ᵢ (𝓜 a)) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b))
      by auto
  hence 9: ⊢ ¬ᵢ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b))) ≡ᵢ
           ¬ᵢ((di(𝓜 a) ∧ᵢ (𝓜 a)) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b)))
      using prop01 by blast
  have 10: ⊢ ¬ᵢ((di(𝓜 a) ∧ᵢ (𝓜 a)) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b))) ≡ᵢ
            ¬ᵢ(((𝓜 a)) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b)))
      using AndDiEqv by auto
  have 11: ⊢ ¬ᵢ(((𝓜 a)) ∨ᵢ (di(𝓜 a) ∧ᵢ di(𝓜 b))) ≡ᵢ
            ¬ᵢ(𝓜 a) ∧ᵢ ¬ᵢ(di(𝓜 a) ∧ᵢ di(𝓜 b))
      by auto
  have 12: ⊢ ¬ᵢ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b))) ≡ᵢ
            ¬ᵢ(𝓜 a) ∧ᵢ ¬ᵢ(di(𝓜 a) ∧ᵢ di(𝓜 b))
      using 9 10 11 by auto
  hence 13: ⊢ bs ¬ᵢ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b))) ≡ᵢ
            bs (¬ᵢ(𝓜 a) ∧ᵢ ¬ᵢ(di(𝓜 a) ∧ᵢ di(𝓜 b)))
      using BsEqvRule by blast
  have 14: ⊢ bs ((¬ᵢ(𝓜 a)) ∧ᵢ ¬ᵢ(di(𝓜 a) ∧ᵢ di(𝓜 b))) ≡ᵢ
           bs ((¬ᵢ(𝓜 a))) ∧ᵢ bs(¬ᵢ(di(𝓜 a) ∧ᵢ di(𝓜 b)))
      using BsAndEqv using itl-prop(30) by blast
  have 141: ⊢ bs ¬ᵢ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b))) ≡ᵢ
            bs ((¬ᵢ(𝓜 a))) ∧ᵢ bs(¬ᵢ(di(𝓜 a) ∧ᵢ di(𝓜 b)))
      using 13 14 by auto
  hence 15: ⊢ (di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)) ∧ᵢ
            bs ¬ᵢ((di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b))) ≡ᵢ
            (di(𝓜 a)) ∧ᵢ ((𝓜 a) ∨ᵢ di(𝓜 b)) ∧ᵢ
            bs ((¬ᵢ(𝓜 a))) ∧ᵢ bs(¬ᵢ(di(𝓜 a) ∧ᵢ di(𝓜 b)))
```

by *auto*
have *16*: $\vdash (di(\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$
$bs\ ((\neg_i(\mathcal{M}\ a))) \wedge_i bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i$
$(bs\ ((\neg_i(\mathcal{M}\ a))) \wedge_i di(\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$
$bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)))$
by *auto*
have *17*: $\vdash (bs\ ((\neg_i(\mathcal{M}\ a))) \wedge_i di(\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$
$bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i$
$(\rhd(\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$
$bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)))$
using *FstEqvBsNotAndDi itl-prop*(*30*) *prop06* **by** *blast*
have *18*: $\vdash (\rhd(\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$
$bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i$
$((\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$
$bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)))$
using *MFixFst itl-prop*(*30*) *prop06* **by** *blast*
have *19*: $\vdash ((\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$
$bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i$
$((\mathcal{M}\ a)) \wedge_i\ \ bs(\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)))$
by *auto*
have *20*: $\vdash\ \ (\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i (\neg_i(di(\mathcal{M}\ a)) \vee_i \neg_i(di(\mathcal{M}\ b))\ )$
by *auto*
have *21*: $\vdash (\neg_i(di(\mathcal{M}\ a)) \vee_i \neg_i(di(\mathcal{M}\ b))\ ) \equiv_i ((bi\ \neg_i(\mathcal{M}\ a)) \vee_i (bi\neg_i(\mathcal{M}\ b))\ )$
by *auto*
have *22*: $\vdash (\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i ((bi\ \neg_i(\mathcal{M}\ a)) \vee_i (bi\neg_i(\mathcal{M}\ b)))$
using *20 21* **by** *auto*
hence *23*: $\vdash bs\ (\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i bs\ ((bi\ \neg_i(\mathcal{M}\ a)) \vee_i (bi\neg_i(\mathcal{M}\ b)))$
using *BsEqvRule* **by** *blast*
have *24*: $\vdash bs\ ((bi\ \neg_i(\mathcal{M}\ a)) \vee_i (bi\neg_i(\mathcal{M}\ b))) \equiv_i bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b))$
using *BsOrBsEqvBsBiOrBi itl-prop*(*30*) **by** *blast*
have *25*: $\vdash bs\ (\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b))$
using *23 24* **by** *auto*
hence *26*: $\vdash (\mathcal{M}\ a) \wedge_i bs\ (\neg_i(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b))) \equiv_i$
$(\mathcal{M}\ a) \wedge_i (bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b)))$
by *auto*
have *27*: $\vdash(\mathcal{M}\ a) \wedge_i (bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b))) \equiv_i$
$\rhd(\mathcal{M}\ a) \wedge_i (bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b)))$
using *MFixFst prop06* **by** *blast*
have *28*: $\vdash \rhd(\mathcal{M}\ a) \wedge_i (bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b))) \equiv_i$
$(\mathcal{M}\ a) \wedge_i bs\ \neg_i(\mathcal{M}\ a) \wedge_i (bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b)))$
by (*simp add*: *first-d-def*)
have *29*: $\vdash (\mathcal{M}\ a) \wedge_i bs\ \neg_i(\mathcal{M}\ a) \wedge_i (bs\ (\neg_i(\mathcal{M}\ a)) \vee_i bs\ (\neg_i(\mathcal{M}\ b))) \equiv_i$
$(\mathcal{M}\ a) \wedge_i bs\ \neg_i(\mathcal{M}\ a)$
by *auto*
have *30*: $\vdash (\mathcal{M}\ a) \wedge_i bs\ \neg_i(\mathcal{M}\ a) \equiv_i \rhd(\mathcal{M}\ a)$
by (*simp add*: *first-d-def*)
have *31*: $\vdash \rhd(\mathcal{M}\ a) \equiv_i (\mathcal{M}\ a)$
using *MFixFst* **by** *auto*
have *32*: $\vdash \mathcal{M}(a\ UPTO\ (a\ THRU\ b)) \equiv_i$
$(di(\mathcal{M}\ a)) \wedge_i ((\mathcal{M}\ a) \vee_i di(\mathcal{M}\ b)) \wedge_i$

```
          bs ¬ᵢ( (di(ℳ a)) ∧ᵢ ((ℳ a) ∨ᵢ di(ℳ b)))
      using 1 2 6 7 by auto
  have 33: ⊢ (di(ℳ a)) ∧ᵢ ((ℳ a) ∨ᵢ di(ℳ b)) ∧ᵢ
          bs ¬ᵢ( (di(ℳ a)) ∧ᵢ ((ℳ a) ∨ᵢ di(ℳ b))) ≡ᵢ
           ((ℳ a)) ∧ᵢ  bs(¬ᵢ(di(ℳ a) ∧ᵢ di(ℳ b)))
      using 15 16 17 18 19 by auto
  have 34: ⊢ ((ℳ a)) ∧ᵢ  bs(¬ᵢ(di(ℳ a) ∧ᵢ di(ℳ b))) ≡ᵢ (ℳ a)
      using 26 27 28 29 30 31 using prop03 by blast
 from 32 33 34 show ?thesis by (simp add: eq-d-def)
qed

lemma MThruUptoAbsorp:
 (a THRU (a UPTO b)) ≃ (a)
proof −
  have 1: ⊢ ℳ(a THRU (a UPTO b)) ≡ᵢ ▷(di(ℳ a) ∧ᵢ di(▷((ℳ a) ∨ᵢ (ℳ b))))
      by simp
  have 2: ⊢  ▷(di(ℳ a) ∧ᵢ di(▷((ℳ a) ∨ᵢ (ℳ b)))) ≡ᵢ
            ▷(di(ℳ a) ∧ᵢ di(((ℳ a) ∨ᵢ (ℳ b))))
      using DiEqvDiFst using FstEqvRule itl-prop(30) prop05 by blast
  have 3: ⊢ ▷(di(ℳ a) ∧ᵢ di(((ℳ a) ∨ᵢ (ℳ b)))) ≡ᵢ
          ▷(di(ℳ a) ∧ᵢ (di(ℳ a) ∨ᵢ di(ℳ b)))
      using DiOrEqv using FstEqvRule prop05 by blast
  have 4: ⊢ (di(ℳ a) ∧ᵢ (di(ℳ a) ∨ᵢ di(ℳ b))) ≡ᵢ (di(ℳ a))
       by auto
  hence 5: ⊢ ▷(di(ℳ a) ∧ᵢ (di(ℳ a) ∨ᵢ di(ℳ b))) ≡ᵢ ▷(di(ℳ a))
      using FstEqvRule by blast
  have 6: ⊢ ▷(di(ℳ a)) ≡ᵢ ▷(ℳ a)
      using FstDiEqvFst by blast
  have 7: ⊢ ▷(ℳ a) ≡ᵢ (ℳ a)
      using MFixFst by auto
 from 1 2 3 5 6 7 show ?thesis by (simp add: eq-d-def)
qed

lemma MUptoThruDistrib:
 (a UPTO (b THRU c)) ≃ ((a UPTO b) THRU (a UPTO c))
proof −
  have 1: ⊢ ℳ((a UPTO b) THRU (a UPTO c)) ≡ᵢ
          ▷( di(▷((ℳ a) ∨ᵢ (ℳ b))) ∧ᵢ di(▷((ℳ a) ∨ᵢ (ℳ c))) )
      by simp
  have 2: ⊢ ( di(▷((ℳ a) ∨ᵢ (ℳ b))) ∧ᵢ di(▷((ℳ a) ∨ᵢ (ℳ c))) ) ≡ᵢ
          ( di(((ℳ a) ∨ᵢ (ℳ b))) ∧ᵢ di(((ℳ a) ∨ᵢ (ℳ c))) )
      using DiEqvDiFst by (metis itl-prop(31) prop22)
  have 3: ⊢ ( di(((ℳ a) ∨ᵢ (ℳ b))) ∧ᵢ di(((ℳ a) ∨ᵢ (ℳ c))) ) ≡ᵢ
          ( di(ℳ a) ∨ᵢ di(ℳ b)) ∧ᵢ (di(ℳ a) ∨ᵢ di(ℳ c))
      using DiOrEqv by auto
  have 4: ⊢ ( di(ℳ a) ∨ᵢ di(ℳ b)) ∧ᵢ (di(ℳ a) ∨ᵢ di(ℳ c)) ≡ᵢ
          di(ℳ a) ∨ᵢ (di(ℳ b) ∧ᵢ di(ℳ c))
      by auto
  have 5: ⊢ ( di(▷((ℳ a) ∨ᵢ (ℳ b))) ∧ᵢ di(▷((ℳ a) ∨ᵢ (ℳ c))) ) ≡ᵢ
          di(ℳ a) ∨ᵢ (di(ℳ b) ∧ᵢ di(ℳ c))
```

using *2 3 4* **by** *auto*

hence *6*: $\vdash \rhd(\ di(\rhd((\mathcal{M}\ a) \vee_i (\mathcal{M}\ b))) \wedge_i di(\rhd((\mathcal{M}\ a) \vee_i (\mathcal{M}\ c)))\ ) \equiv_i$
$\rhd(\ di(\mathcal{M}\ a) \vee_i (di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))\ )$
using *FstEqvRule* **by** *blast*

have *7*: $\vdash \rhd(\ di(\mathcal{M}\ a) \vee_i (di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))\ ) \equiv_i$
$\rhd(\ \rhd(di(\mathcal{M}\ a)) \vee_i \rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))\ )$
using *FstFstOrEqvFstOr* **by** *auto*

have *8*: $\vdash \rhd(di(\mathcal{M}\ a)) \equiv_i \rhd((\mathcal{M}\ a))$
using *FstDiEqvFst* **by** *blast*

have *9*: $\vdash \rhd((\mathcal{M}\ a)) \equiv_i (\mathcal{M}\ a)$
using *MFixFst* **by** *auto*

have *10*: $\vdash \rhd(di(\mathcal{M}\ a)) \equiv_i (\mathcal{M}\ a)$
using *8 9* **by** *auto*

hence *11*: $\vdash \rhd(di(\mathcal{M}\ a)) \vee_i \rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)) \equiv_i$
$(\mathcal{M}\ a) \vee_i \rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))$
by *auto*

hence *12*: $\vdash \rhd(\rhd(di(\mathcal{M}\ a)) \vee_i \rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))) \equiv_i$
$\rhd((\mathcal{M}\ a) \vee_i \rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)))$
using *FstEqvRule* **by** *blast*

have *13*: $\vdash \rhd((\mathcal{M}\ a) \vee_i \rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))) \equiv_i \mathcal{M}(a\ UPTO\ (b\ THRU\ c))$
by *simp*

from *1 6 7 12 13* **show** *?thesis* **by** (*simp add*: *eq-d-def*)

qed

lemma *MThruUptoDistrib*:
$(a\ THRU\ (b\ UPTO\ c)) \simeq ((a\ THRU\ b)\ UPTO\ (a\ THRU\ c))$

proof −

have *1*: $\vdash \mathcal{M}((a\ THRU\ b)\ UPTO\ (a\ THRU\ c)) \equiv_i$
$\rhd(\rhd(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i \rhd(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ c)))$ **by** *simp*

have *2*: $\vdash \rhd(\rhd(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i \rhd(di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ c))) \equiv_i$
$\rhd((di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i (di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ c)))$ **using** *FstFstOrEqvFstOr* **by** *auto*

have *3*: $\vdash ((di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i (di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ c))) \equiv_i$
$(di(\mathcal{M}\ a) \wedge_i (di(\mathcal{M}\ b) \vee_i di(\mathcal{M}\ c)))$ **by** *auto*

have *4*: $\vdash (di(\mathcal{M}\ a) \wedge_i (di(\mathcal{M}\ b) \vee_i di(\mathcal{M}\ c))) \equiv_i$
$(di(\mathcal{M}\ a) \wedge_i di(\ (\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))$ **using** *DiOrEqv* **by** *auto*

have *5*: $\vdash (di(\mathcal{M}\ a) \wedge_i di(\ (\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))) \equiv_i$
$(di(\mathcal{M}\ a) \wedge_i di(\rhd(\ (\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))))$ **using** *DiEqvDiFst prop05* **by** *blast*

have *6*: $\vdash ((di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i (di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ c))) \equiv_i$
$(di(\mathcal{M}\ a) \wedge_i di(\rhd(\ (\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))))$ **using** *3 4 5* **by** *auto*

hence *7*: $\vdash \rhd((di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ b)) \vee_i (di(\mathcal{M}\ a) \wedge_i di(\mathcal{M}\ c))) \equiv_i$
$\rhd(di(\mathcal{M}\ a) \wedge_i di(\rhd(\ (\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))))$ **using** *FstEqvRule* **by** *blast*

have *8*: $\vdash \rhd(di(\mathcal{M}\ a) \wedge_i di(\rhd(\ (\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))) \equiv_i$
$\mathcal{M}(a\ THRU\ (b\ UPTO\ c))$ **by** *simp*

from *1 2 7 8* **show** *?thesis* **by** (*simp add*: *eq-d-def*)

qed

lemma *MThruUptoRDistrib*:
$((a\ THRU\ b)\ UPTO\ c) \simeq ((a\ UPTO\ c)\ THRU\ (b\ UPTO\ c))$

proof −

have *1*: $((a\ THRU\ b)\ UPTO\ c) \simeq (c\ UPTO\ (a\ THRU\ b))$

using *MUptoCommut* **by** *auto*
have *2*: (*c UPTO* (*a THRU b*)) ≃ ((*c UPTO a*) *THRU* (*c UPTO b*))
using *MUptoThruDistrib* **by** *auto*
have *3*: (*c UPTO a*) ≃ (*a UPTO c*)
using *MUptoCommut* **by** *auto*
have *4*: (*c UPTO b*) ≃ (*b UPTO c*)
using *MUptoCommut* **by** *auto*
have *5*: ((*c UPTO a*) *THRU* (*c UPTO b*)) ≃ ((*a UPTO c*) *THRU* (*c UPTO b*))
using *3* **by** (*simp add*: *MonEqRefl MonEqSubstThru*)
have *6*: ((*a UPTO c*) *THRU* (*c UPTO b*)) ≃ ((*c UPTO b*) *THRU* (*a UPTO c*))
using *MThruCommut* **by** *auto*
have *7*: ((*c UPTO b*) *THRU* (*a UPTO c*)) ≃ ((*b UPTO c*) *THRU* (*a UPTO c*))
using *4* **by** (*simp add*: *MonEqRefl MonEqSubstThru*)
from *1 2 5 6 7* **show** *?thesis* **using** *MThruCommut* **by** (*metis MonEqTrans*)
qed

lemma *MUptoThruRDistrib*:
((*a UPTO b*) *THRU c*) ≃ ((*a THRU c*) *UPTO* (*b THRU c*))
proof −
have *1*: ((*a UPTO b*) *THRU c*) ≃ (*c THRU* (*a UPTO b*))
using *MThruCommut* **by** *auto*
have *2*: (*c THRU* (*a UPTO b*)) ≃ ((*c THRU a*) *UPTO* (*c THRU b*))
using *MThruUptoDistrib* **by** *auto*
have *3*: (*c THRU a*) ≃ (*a THRU c*)
using *MThruCommut* **by** *auto*
have *4*: (*c THRU b*) ≃ (*b THRU c*)
using *MThruCommut* **by** *auto*
have *5*: ((*c THRU a*) *UPTO* (*c THRU b*)) ≃ ((*a THRU c*) *UPTO* (*c THRU b*))
using *3* **by** (*simp add*: *MonEqRefl MonEqSubstUpto*)
have *6*: ((*a THRU c*) *UPTO* (*c THRU b*)) ≃ ((*c THRU b*) *UPTO* (*a THRU c*))
using *MUptoCommut* **by** *auto*
have *7*: ((*c THRU b*) *UPTO* (*a THRU c*)) ≃ ((*b THRU c*) *UPTO* (*a THRU c*))
using *4* **by** (*simp add*: *MonEqRefl MonEqSubstUpto*)
from *1 2 5 6 7* **show** *?thesis* **using** *MUptoCommut* **by** (*metis MonEqTrans*)
qed

lemma *MWithAndDistrib*:
((*a AND b*) *WITH f*) ≃ ((*a WITH f*) *AND* (*b WITH f*))
proof −
have *1*: $\vdash \mathcal{M}((a\ AND\ b)\ WITH\ f) \equiv_i \mathcal{M}(a\ AND\ b) \wedge_i f$
by *simp*
have *2*: $\vdash \mathcal{M}(a\ AND\ b) \equiv_i \mathcal{M}(a\ WITH\ (\mathcal{M}\ b))$
by (*simp add*: *mAND-d-def*)
have *3*: $\vdash \mathcal{M}(a\ AND\ b) \wedge_i f \equiv_i \mathcal{M}(a\ WITH\ (\mathcal{M}\ b)) \wedge_i f$
using *2 prop06* **by** *simp*
have *4*: $\vdash \mathcal{M}(a\ WITH\ (\mathcal{M}\ b)) \wedge_i f \equiv_i \mathcal{M}(a) \wedge_i \mathcal{M}(b) \wedge_i f$
by *simp*
have *5*: $\vdash \mathcal{M}(a) \wedge_i \mathcal{M}(b) \wedge_i f \equiv_i (\mathcal{M}(a) \wedge_i f) \wedge_i (\mathcal{M}(b) \wedge_i f)$
by *auto*

```
  have 6: $\vdash (\mathcal{M}(a) \wedge_i f) \wedge_i (\mathcal{M}(b) \wedge_i f) \equiv_i \mathcal{M}(a\ WITH\ f) \wedge_i \mathcal{M}(b\ WITH\ f)$
      by simp
  have 7: $\vdash \mathcal{M}(a\ WITH\ f) \wedge_i \mathcal{M}(b\ WITH\ f) \equiv_i \mathcal{M}((a\ WITH\ f)\ WITH\ (\mathcal{M}(b\ WITH\ f)))$
      by simp
  have 8: $\vdash \mathcal{M}((a\ WITH\ f)\ WITH\ (\mathcal{M}(b\ WITH\ f))) \equiv_i \mathcal{M}((a\ WITH\ f)\ AND\ (b\ WITH\ f))$
      by (simp add: mAND-d-def)
  from 1 2 3 4 5 6 7 8 show ?thesis by (simp add: eq-d-def)
qed


lemma MHaltWithAndDistrib:
 $(((HALT\ w)\ WITH\ f)\ AND\ ((HALT\ w)\ WITH\ g)) \simeq ((HALT\ w)\ WITH\ (f \wedge_i g))$
proof −
  have 1: $\vdash \mathcal{M}(((HALT\ w)\ WITH\ f)\ AND\ ((HALT\ w)\ WITH\ g)) \equiv_i$
          $\mathcal{M}(((HALT\ w)\ WITH\ f)\ WITH\ (\mathcal{M}((HALT\ w)\ WITH\ g)))$
      by (simp add: mAND-d-def)
  have 2: $\vdash \mathcal{M}(((HALT\ w)\ WITH\ f)\ WITH\ (\mathcal{M}((HALT\ w)\ WITH\ g))) \equiv_i$
          $\mathcal{M}(HALT\ w) \wedge_i f \wedge_i \mathcal{M}(HALT\ w) \wedge_i g$
      by (simp add: mHALT-d-def)
  have 3: $\vdash \mathcal{M}(HALT\ w) \wedge_i f \wedge_i \mathcal{M}(HALT\ w) \wedge_i g \equiv_i \mathcal{M}(HALT\ w) \wedge_i f \wedge_i g$
      by auto
  from 1 2 3 show ?thesis by (simp add: eq-d-def)
qed

lemma MHaltWithUptoHaltWithEqvHaltWithOr:
 $(((HALT\ w)\ WITH\ f)\ UPTO\ ((HALT\ w)\ WITH\ g)) \simeq ((HALT\ w)\ WITH\ (f \vee_i g))$
proof −
  have 1: $\vdash \mathcal{M}(((HALT\ w)\ WITH\ f)\ UPTO\ ((HALT\ w)\ WITH\ g)) \equiv_i$
          $\rhd(\mathcal{M}((HALT\ w)\ WITH\ f) \vee_i \mathcal{M}((HALT\ w)\ WITH\ g))$
      by simp
  have 2: $\vdash \rhd(\mathcal{M}((HALT\ w)\ WITH\ f) \vee_i \mathcal{M}((HALT\ w)\ WITH\ g)) \equiv_i$
          $\rhd((\mathcal{M}(HALT\ w) \wedge_i f) \vee_i (\mathcal{M}(HALT\ w) \wedge_i g))$
      by simp
  have 3: $\vdash (\mathcal{M}(HALT\ w) \wedge_i f) \vee_i (\mathcal{M}(HALT\ w) \wedge_i g) \equiv_i (\mathcal{M}(HALT\ w) \wedge_i (f \vee_i g))$
      by auto
  have 4: $\vdash \rhd((\mathcal{M}(HALT\ w) \wedge_i f) \vee_i (\mathcal{M}(HALT\ w) \wedge_i g)) \equiv_i \rhd(\mathcal{M}(HALT\ w) \wedge_i (f \vee_i g))$
      using 3 FstEqvRule by blast
  have 5: $\vdash \rhd(\mathcal{M}(HALT\ w) \wedge_i (f \vee_i g)) \equiv_i \rhd(\mathcal{M}((HALT\ w)\ WITH\ (f \vee_i g)))$
      by simp
  have 6: $\vdash (\mathcal{M}((HALT\ w)\ WITH\ (f \vee_i g))) \equiv_i \rhd(\mathcal{M}((HALT\ w)\ WITH\ (f \vee_i g)))$
      using MFixFst by blast
  from 1 2 3 4 5 6 show ?thesis by (simp add: eq-d-def)
qed

lemma MHaltWithThruHaltWithEqvHaltWithAndHaltWith:
 $(((HALT\ w)\ WITH\ f)\ THRU\ ((HALT\ w)\ WITH\ g)) \simeq (((HALT\ w)\ WITH\ f)\ AND\ ((HALT\ w)\ WITH\ g))$
proof −
  have 1: $\vdash \mathcal{M}(((HALT\ w)\ WITH\ f)\ THRU\ ((HALT\ w)\ WITH\ g)) \equiv_i$
          $\rhd(\ di(\mathcal{M}(HALT\ w) \wedge_i f) \wedge_i di(\mathcal{M}(HALT\ w) \wedge_i g)\ )$
      by simp
```

have *2*: $\vdash di(\mathcal{M}(HALT\ w) \wedge_i f) \wedge_i di(\mathcal{M}(HALT\ w) \wedge_i g) \equiv_i$
$di(halt(init\ w) \wedge_i f) \wedge_i di(halt(init\ w) \wedge_i g)$
using *MHaltAlt DiEqvDi* **by** *auto*
have *3*: $\vdash di(halt(init\ w) \wedge_i f) \wedge_i di(halt(init\ w) \wedge_i g) \equiv_i$
$di(halt(init\ w) \wedge_i f \wedge_i g)$
using *DiHaltAndDiHaltAndEqvDiHaltAndAnd* **by** *simp*
have *4*: $\vdash di(\ halt(init\ w) \wedge_i f \wedge_i g) \equiv_i di(\mathcal{M}(HALT\ w) \wedge_i f \wedge_i g)$
using *MHaltAlt* **by** *auto*
have *5*: $\vdash di(\mathcal{M}(HALT\ w) \wedge_i f) \wedge_i di(\mathcal{M}(HALT\ w) \wedge_i g) \equiv_i di(\mathcal{M}(HALT\ w) \wedge_i f \wedge_i g)$
using *2 3 4* **by** *simp*
have *6*: $\vdash \triangleright(di(\mathcal{M}(HALT\ w) \wedge_i f) \wedge_i di(\mathcal{M}(HALT\ w) \wedge_i g)) \equiv_i \triangleright(di(\mathcal{M}(HALT\ w) \wedge_i f \wedge_i g))$
using *5 FstEqvRule* **by** *blast*
have *7*: $\vdash \triangleright(di(\mathcal{M}(HALT\ w) \wedge_i f \wedge_i g)) \equiv_i \triangleright(\mathcal{M}(HALT\ w) \wedge_i f \wedge_i g)$
using *FstDiEqvFst* **by** *simp*
have *8*: $\vdash \triangleright(\mathcal{M}(HALT\ w) \wedge_i f \wedge_i g) \equiv_i \triangleright(\mathcal{M}((HALT\ w)\ WITH\ (f \wedge_i g)))$
by *simp*
have *9*: $\vdash \mathcal{M}((HALT\ w)\ WITH\ (f \wedge_i g)) \equiv_i \triangleright(\mathcal{M}((HALT\ w)\ WITH\ (f \wedge_i g)))$
using *MFixFst* **by** *blast*
have *10*: $\vdash \mathcal{M}(((HALT\ w)\ WITH\ f)\ THRU\ ((HALT\ w)\ WITH\ g)) \equiv_i \mathcal{M}((HALT\ w)\ WITH\ (f \wedge_i g))$
using *1 2 3 4 5 6 7 8 9* **by** *simp*
have *11*: $\vdash \mathcal{M}(((HALT\ w)\ WITH\ f)\ AND\ ((HALT\ w)\ WITH\ g)) \equiv_i \mathcal{M}((HALT\ w)\ WITH\ (f \wedge_i g))$
using *MHaltWithAndDistrib* **using** *eq-d-def* **by** *blast*
have *12*: $\vdash \mathcal{M}((HALT\ w)\ WITH\ (f \wedge_i g)) \equiv_i \mathcal{M}(((HALT\ w)\ WITH\ f)\ AND\ ((HALT\ w)\ WITH\ g))$
using *11* **by** *simp*
from *10 12* **show** *?thesis* **by** (*simp add*: *eq-d-def*)
qed

lemma *MThenAndDistrib*:
$(a\ THEN\ (b\ AND\ c)) \simeq ((a\ THEN\ b)\ AND\ (a\ THEN\ c))$
proof –
have *1*: $\vdash \mathcal{M}(a\ THEN\ (b\ AND\ c)) \equiv_i (\mathcal{M}(a))\ ;\ (\mathcal{M}(b\ AND\ c))$
by *simp*
have *2*: $\vdash (\mathcal{M}(a))\ ;\ (\mathcal{M}(b\ AND\ c)) \equiv_i (\mathcal{M}(a))\ ;\ (\mathcal{M}(b) \wedge_i \mathcal{M}(c))$
by (*simp add*: *mAND-d-def*)
have *3*: $\vdash (\mathcal{M}(a))\ ;\ (\mathcal{M}(b) \wedge_i \mathcal{M}(c)) \equiv_i\ \triangleright (\mathcal{M}(a))\ ;\ (\mathcal{M}(b) \wedge_i \mathcal{M}(c))$
using *MFixFst LeftChopEqvChop* **by** *blast*
have *4*: $\vdash \triangleright (\mathcal{M}(a))\ ;\ (\mathcal{M}(b) \wedge_i \mathcal{M}(c)) \equiv_i\ ((\triangleright (\mathcal{M}(a))\ ;\ (\mathcal{M}(b))) \wedge_i (\triangleright (\mathcal{M}(a))\ ;\ (\mathcal{M}(c))))$
using *LFstAndDistrC* **by** *fastforce*
have *5*: $\vdash((\triangleright (\mathcal{M}(a))\ ;\ (\mathcal{M}(b))) \wedge_i (\triangleright (\mathcal{M}(a))\ ;\ (\mathcal{M}(c)))) \equiv_i$
$((\ (\mathcal{M}(a))\ ;\ (\mathcal{M}(b))) \wedge_i (\ (\mathcal{M}(a))\ ;\ (\mathcal{M}(c)))\ )$
using *MFixFst* **by** *auto*
have *6*: $\vdash ((\ (\mathcal{M}(a))\ ;\ (\mathcal{M}(b))) \wedge_i (\ (\mathcal{M}(a))\ ;\ (\mathcal{M}(c)))\ ) \equiv_i$
$(\mathcal{M}(a\ THEN\ b) \wedge_i \mathcal{M}(a\ THEN\ c))$
by *simp*
have *7*: $\vdash (\mathcal{M}(a\ THEN\ b) \wedge_i \mathcal{M}(a\ THEN\ c)) \equiv_i \mathcal{M}((a\ THEN\ b)\ AND\ (a\ THEN\ c))$
by (*simp add*: *mAND-d-def*)
from *1 2 3 4 5 6 7* **show** *?thesis* **using** *MonEq* **by** (*simp add*: *eq-d-def*)
qed

lemma *MThenUptoDistrib*:
$(a\ \mathit{THEN}\ (b\ \mathit{UPTO}\ c)) \simeq ((a\ \mathit{THEN}\ b)\ \mathit{UPTO}\ (a\ \mathit{THEN}\ c))$
proof −
have *1*: $\vdash (\mathcal{M}\ (a\ \mathit{THEN}\ (b\ \mathit{UPTO}\ c))) \equiv_i ((\mathcal{M}\ a);(\rhd((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))))$
  by *simp*
have *2*: $\vdash ((\mathcal{M}\ a);(\rhd((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))) \equiv_i (\rhd(\mathcal{M}\ a);(\rhd((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))))$
  using *MFixFst LeftChopEqvChop* **by** *blast*
have *3*: $\vdash (\rhd(\mathcal{M}\ a);(\rhd((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))) \equiv_i ((\rhd(\rhd(\mathcal{M}\ a);((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)))))$
  using *FstFstChopEqvFstChopFst* **by** *fastforce*
have *4*: $\vdash \rhd(\mathcal{M}\ a);((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)) \equiv_i (\mathcal{M}\ a);((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))$
  using *MFixFst LeftChopEqvChop itl-prop*(*30*) **by** *blast*
have *5*: $\vdash (\mathcal{M}\ a);((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)) \equiv_i ((\mathcal{M}\ a);(\mathcal{M}\ b) \vee_i (\mathcal{M}\ a);(\mathcal{M}\ c))$
  using *ChopOrEqv* **by** *blast*
have *6*: $\vdash ((\mathcal{M}\ a);(\mathcal{M}\ b) \vee_i (\mathcal{M}\ a);(\mathcal{M}\ c)) \equiv_i (\mathcal{M}(a\ \mathit{THEN}\ b) \vee_i \mathcal{M}(a\ \mathit{THEN}\ c))$
  by *simp*
have *7*: $\vdash \rhd(\mathcal{M}\ a);((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c)) \equiv_i (\mathcal{M}(a\ \mathit{THEN}\ b) \vee_i \mathcal{M}(a\ \mathit{THEN}\ c))$
  using *6 5 4* **by** *fastforce*
have *8*: $\vdash \rhd(\rhd(\mathcal{M}\ a);((\mathcal{M}\ b) \vee_i (\mathcal{M}\ c))) \equiv_i \rhd(\mathcal{M}(a\ \mathit{THEN}\ b) \vee_i \mathcal{M}(a\ \mathit{THEN}\ c))$
  using *7 FstEqvRule* **by** *blast*
have *9*: $\vdash \rhd(\mathcal{M}(a\ \mathit{THEN}\ b) \vee_i \mathcal{M}(a\ \mathit{THEN}\ c)) \equiv_i \mathcal{M}((a\ \mathit{THEN}\ b)\ \mathit{UPTO}\ (a\ \mathit{THEN}\ c))$
  by *simp*
from *9 7 1 2 3* **show** *?thesis* **by** (*meson 8 eq-d-def prop03*)
qed

lemma *MThenThruDistrib*:
$(a\ \mathit{THEN}\ (b\ \mathit{THRU}\ c)) \simeq ((a\ \mathit{THEN}\ b)\ \mathit{THRU}\ (a\ \mathit{THEN}\ c))$
proof −
have *1*: $\vdash \mathcal{M}(a\ \mathit{THEN}\ (b\ \mathit{THRU}\ c)) \equiv_i (\mathcal{M}\ a);\rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))$
  by *simp*
have *2*: $\vdash (\mathcal{M}\ a);\rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)) \equiv_i \rhd(\mathcal{M}\ a);\rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c))$
  using *MFixFst LeftChopEqvChop* **by** *blast*
have *3*: $\vdash \rhd(\mathcal{M}\ a);\rhd(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)) \equiv_i \rhd(\rhd(\mathcal{M}\ a);(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)\ ))$
  using *FstFstChopEqvFstChopFst* **by** *fastforce*
have *4*: $\vdash \rhd(\mathcal{M}\ a);(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)\ ) \equiv_i (\rhd(\mathcal{M}\ a);di(\mathcal{M}\ b) \wedge_i \rhd(\mathcal{M}\ a);di(\mathcal{M}\ c))$
  using *LFstAndDistrC* **using** *itl-prop*(*30*) **by** *blast*
have *5*: $\vdash (\rhd(\mathcal{M}\ a);di(\mathcal{M}\ b) \wedge_i \rhd(\mathcal{M}\ a);di(\mathcal{M}\ c)) \equiv_i ((\mathcal{M}\ a);di(\mathcal{M}\ b) \wedge_i (\mathcal{M}\ a);di(\mathcal{M}\ c))$
  using *MFixFst* **by** *auto*
have *6*: $\vdash (\mathcal{M}\ a);di(\mathcal{M}\ b) \equiv_i (\mathcal{M}\ a);((\mathcal{M}\ b);true_i)$
  by (*simp add*: *di-d-def*)
have *7*: $\vdash (\mathcal{M}\ a);((\mathcal{M}\ b);true_i) \equiv_i ((\mathcal{M}\ a);(\mathcal{M}\ b));true_i$
  using *ChopAssoc* **by** *blast*
have *8*: $\vdash ((\mathcal{M}\ a);(\mathcal{M}\ b));true_i \equiv_i di((\mathcal{M}\ a);(\mathcal{M}\ b))$
  by (*simp add*: *di-d-def*)
have *9*: $\vdash (\mathcal{M}\ a);di(\mathcal{M}\ b) \equiv_i di((\mathcal{M}\ a);(\mathcal{M}\ b))$
  using *8 7 6* **by** *fastforce*
have *10*: $\vdash (\mathcal{M}\ a);di(\mathcal{M}\ c) \equiv_i (\mathcal{M}\ a);((\mathcal{M}\ c);true_i)$
  by (*simp add*: *di-d-def*)
have *11*: $\vdash (\mathcal{M}\ a);((\mathcal{M}\ c);true_i) \equiv_i ((\mathcal{M}\ a);(\mathcal{M}\ c));true_i$
  using *ChopAssoc* **by** *blast*
have *12*: $\vdash ((\mathcal{M}\ a);(\mathcal{M}\ c));true_i \equiv_i di((\mathcal{M}\ a);(\mathcal{M}\ c))$

by (*simp add*: *di-d-def*)
have *13*: $\vdash (\mathcal{M}\ a);di(\mathcal{M}\ c) \equiv_i di((\mathcal{M}\ a);(\mathcal{M}\ c))$
using *12 11 10* **by** *fastforce*
have *14*: $\vdash ((\mathcal{M}\ a);di(\mathcal{M}\ b) \wedge_i (\mathcal{M}\ a);di(\mathcal{M}\ c)) \equiv_i (di((\mathcal{M}\ a);(\mathcal{M}\ b)) \wedge_i di((\mathcal{M}\ a);(\mathcal{M}\ c)))$
using *13 9* **by** *fastforce*
have *15*: $\vdash (di((\mathcal{M}\ a);(\mathcal{M}\ b)) \wedge_i di((\mathcal{M}\ a);(\mathcal{M}\ c))) \equiv_i (di(\mathcal{M}(a\ THEN\ b)) \wedge_i di(\mathcal{M}(a\ THEN\ c)))$
by *simp*
have *16*: $\vdash \triangleright(\mathcal{M}\ a);(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)\ ) \equiv_i (di(\mathcal{M}(a\ THEN\ b)) \wedge_i di(\mathcal{M}(a\ THEN\ c)))$
using *15 14 4 5* **by** *fastforce*
have *17*: $\vdash \triangleright(\triangleright(\mathcal{M}\ a);(di(\mathcal{M}\ b) \wedge_i di(\mathcal{M}\ c)\ )) \equiv_i \triangleright(di(\mathcal{M}(a\ THEN\ b)) \wedge_i di(\mathcal{M}(a\ THEN\ c)))$
using *16 FstEqvRule* **by** *blast*
have *18*: $\vdash \triangleright(di(\mathcal{M}(a\ THEN\ b)) \wedge_i di(\mathcal{M}(a\ THEN\ c))) \equiv_i \mathcal{M}((a\ THEN\ b)\ THRU\ (a\ THEN\ c))$
by *simp*
from *18 16 1 2 3* **show** *?thesis* **by** (*meson 17 eq-d-def prop03*)
qed

end

theory *ITA*
imports *ITL*

begin

8 Interval Temporal Algebra

8.1 Definition of fuse operator

The *fuse* operation corresponds to the chop operation of ITL at semantic level. Although *fuse* is not needed to define the semantics of the ITL fusion/chop operation, it is introduced to link with the work of [1]. The ITL proof system is derived from a collection of algebraic laws. This work is a continuation of [2].

primrec *fuse* :: *'a interval* ⇒ *'a interval* ⇒ *'a interval* **where**
fuse-St : *fuse* (*St x*) *ys* = *ys*
| *fuse-Cons* : *fuse* (*x* ⊙ *xs*) *ys* = *x* ⊙ (*fuse xs ys*)

8.1.1 Fuse lemmas

lemma *interval-fuse-leftneutral* :
fuse (*St* (*intfirst xs*)) *xs* = *xs*
by *simp*

lemma *interval-fuse-rightneutral* :
fuse xs (*St* (*intlast xs*)) = *xs*
by (*induct xs*) *simp-all*

lemma *interval-intfirst-fuse* :
assumes *intlast xs* =*intfirst ys*

shows *intfirst* (*fuse xs ys*) = *intfirst xs*
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-intlast-fuse* :
assumes *intlast xs* = *intfirst ys*
shows *intlast* (*fuse xs ys*) = *intlast ys*
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-FusionAssoc* :
assumes (*intlast xs*) = (*intfirst ys*) ∧ (*intlast ys*) = (*intfirst zs*)
shows (*fuse xs* (*fuse ys zs*)) = (*fuse* (*fuse xs ys*) *zs*)
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-fuse-intlen* :
assumes *intlast xs* = *intfirst ys*
shows *intlen* (*fuse xs ys*) = (*intlen xs*) + (*intlen ys*)
using *assms* **by** (*induct xs*) *simp-all*

lemma *interval-intlast-intfirst*:
(*intlast* (*prefix i xs*)) = (*intfirst* (*suffix i xs*))
by (*induct xs arbitrary*: *i*, *simp*, *simp add*: *Nitpick*.*case-nat-unfold*)

lemma *interval-fuse-pref-suf* :
(*fuse* (*prefix i xs*) (*suffix i xs*)) = *xs*
by (*induct xs arbitrary*: *i*, *simp*, *simp add*: *Nitpick*.*case-nat-unfold*)

lemma *interval-prefix-fuse* :
assumes *intlast xs* = *intfirst ys*
shows (*prefix* (*intlen xs*) (*fuse xs ys*)) = *xs*
using *assms* **by** (*induct xs arbitrary*: *ys*, *simp*, *simp*)

lemma *interval-suffix-fuse* :
assumes *intlast xs* = *intfirst ys*
shows (*suffix* (*intlen xs*) (*fuse xs ys*)) = *ys*
using *assms* **by** (*induct xs arbitrary*: *ys*, *simp*, *simp*)

lemma *chop-fuse-1* :
(∃ *σ1 σ2*. *σ* = *fuse σ1 σ2* ∧
(*σ1* ⊨ *f*) ∧ (*σ2* ⊨ *g*) ∧
(*intlast σ1* = *intfirst σ2*)) ⟷
(∃ *i*. *0* ≤ *i* ∧ *i*≤ *intlen σ* ∧ (*prefix i σ* ⊨ *f*) ∧ (*suffix i σ* ⊨ *g*))
by (*metis interval-fuse-intlen interval-fuse-pref-suf interval-intlast-intfirst*
interval-intlen-gr-zero interval-prefix-fuse interval-suffix-fuse le-add-same-cancel1)

lemma *chop-fuse-2* :
(∃ *σ1 σ2*. *σ* = *fuse σ1 σ2* ∧
(*σ1* ∈ *X*) ∧ (*σ2* ∈ *Y*) ∧
(*intlast σ1* = *intfirst σ2*)) ⟷
(∃ *i*≤ *intlen σ*. (*prefix i σ*)∈*X* ∧ (*suffix i σ*)∈*Y*)
by (*metis interval-fuse-intlen interval-fuse-pref-suf interval-intlast-intfirst*

interval-prefix-fuse interval-suffix-fuse le-add1)

lemma *chop-fuse*:
(∃ *σ1 σ2*. *σ* = *fuse σ1 σ2* ∧
(*σ1* ⊨ *f*) ∧ (*σ2* ⊨ *g*) ∧
(*intlast σ1* = *intfirst σ2*)) ⟷
(*σ* ⊨ *f*;*g*)
using *chop-fuse-1* **by** (*simp add*: *chop-fuse-1*)

8.2 Definition of Set of intervals and Operations on them

type-synonym *'a intervals* = *'a interval set*

definition *lan*:: *'a pitl* ⇒ *'a intervals*
where *lan f* = { *σ* . (*σ* ⊨ *f*) }

definition *fusion* :: *'a intervals* ⇒ *'a intervals* ⇒ *'a intervals* (**infixl** · *70*)
where *X*·*Y* = {*fuse σ1 σ2*| *σ1 σ2*. *σ1* ∈ *X* ∧ *σ2* ∈ *Y* ∧ *intlast σ1* = *intfirst σ2*}

definition *reverse* :: *'a intervals* ⇒ *'a intervals* ((*SRev* -) [*85*] *85*)
where (*SRev X*) = {*intrev σ* |*σ*. *σ* ∈ *X* }

definition *sempty* :: *'a intervals* (*SEmpty*)
where
SEmpty ≡ *range St*

definition *smore* :: *'a intervals* (*SMore*)
where
SMore ≡ − *SEmpty*

definition *sskip* :: *'a intervals* (*SSkip*)
where
SSkip ≡ −(*SEmpty* ∪ (*SMore*·*SMore*))

definition *sfalse* :: *'a intervals* (*SFalse*)
where
SFalse ≡ {}

definition *strue* :: *'a intervals* (*STrue*)
where
STrue ≡ −{}

definition *sinit* :: *'a intervals* ⇒ *'a intervals* ((*SInit* -) [*85*] *85*)
where
SInit X ≡ (*X* ∩ *SEmpty*)·*STrue*

definition *sfin* :: *'a intervals* ⇒ *'a intervals* ((*SFin* -) [*85*] *85*)
where
SFin X ≡ *STrue*·(*X* ∩ *SEmpty*)

```
definition ssometime :: 'a intervals ⇒ 'a intervals ((SSometime -) [85] 85)
where
  SSometime X ≡ STrue·X

definition salways :: 'a intervals ⇒ 'a intervals ((SAlways -) [85] 85)
where
  SAlways X ≡ −(SSometime (−X))

definition sdi :: 'a intervals ⇒ 'a intervals ((SDi -) [85] 85)
where
  SDi X ≡ X·STrue

definition sbi :: 'a intervals ⇒ 'a intervals ((SBi -) [85] 85)
where
  SBi X ≡ −(SDi (−X))

definition sda :: 'a intervals ⇒ 'a intervals ((SDa -) [85] 85)
where
  SDa X ≡ STrue·X·STrue

definition sba :: 'a intervals ⇒ 'a intervals ((SBa -) [85] 85)
where
  SBa X ≡ −(SDa (−X))

definition snext :: 'a intervals ⇒ 'a intervals ((SNext -) [85] 85)
where
  SNext X ≡ SSkip·X

definition swnext :: 'a intervals ⇒ 'a intervals ((SWnext -) [85] 85)
where
  SWnext X ≡ (−(SSkip·(−X)))

definition sprev :: 'a intervals ⇒ 'a intervals ((SPrev -) [85] 85)
where
  SPrev X ≡ X·SSkip

definition swprev :: 'a intervals ⇒ 'a intervals ((SWprev -) [85] 85)
where
  SWprev X ≡ (−((−X)·SSkip))

primrec spower :: 'a intervals ⇒ nat ⇒ 'a intervals ((SPower - -) [88,88] 87)
where
   pwr-0  : SPower X 0       = SEmpty
 | pwr-Suc: SPower X (Suc n) = ((X ∩ SMore)·(SPower X n))

definition sstar :: 'a intervals ⇒ 'a intervals ((SStar -) [85] 85)
where
  SStar X ≡ (⋃ n. SPower X n)
```

8.3 Simplification Lemmas

lemma *snot-elim* :
$x \in -X \longleftrightarrow x \notin X$
by *simp*

lemma *sor-elim* :
$x \in (X \cup Y) \longleftrightarrow (x \in X \vee x \in Y)$
by *simp*

lemma *sand-elim* :
$x \in (X \cap Y) \longleftrightarrow (x \in X \wedge x \in Y)$
by *simp*

lemma *sfalse-elim* :
$\sigma \notin \mathit{SFalse}$
by (*simp add*: *sfalse-def*)

lemma *strue-elim* :
$\sigma \in \mathit{STrue}$
by (*simp add*: *strue-def*)

lemma *sempty-elim* :
$\sigma \in \mathit{SEmpty} \longleftrightarrow \mathit{intlen}\ \sigma = 0$
by (*simp add*: *image-iff interval-st-intlen sempty-def*)

lemma *smore-elim* :
$\sigma \in \mathit{SMore} \longleftrightarrow \mathit{intlen}\ \sigma > 0$
by (*simp add*: *sempty-elim smore-def*)

lemma *fusion-iff*:
$\sigma \in X {\cdot} Y \longleftrightarrow (\exists \sigma 1\ \sigma 2.\ \sigma = \mathit{fuse}\ \sigma 1\ \sigma 2 \wedge \sigma 1 \in X \wedge \sigma 2 \in Y \wedge \mathit{intlast}\ \sigma 1 = \mathit{intfirst}\ \sigma 2)$
by (*unfold fusion-def*) *auto*

lemma *fusion-iff-1*:
$\sigma \in X {\cdot} Y \longleftrightarrow (\exists\ i \le \mathit{intlen}\ \sigma.\ (\mathit{prefix}\ i\ \sigma) \in X \wedge (\mathit{suffix}\ i\ \sigma) \in Y\ )$
by (*simp add*: *chop-fuse-2 fusion-iff*)

lemma *smore-fusion-smore* :
$\sigma \in (\mathit{SMore} {\cdot} \mathit{SMore}) \longleftrightarrow \mathit{intlen}\ \sigma > 1$
using *fusion-iff-1*
by (*metis interval-prefix-length-good interval-suffix-length-good less-one not-less not-less-iff-gr-or-eq smore-elim zero-less-diff*)

lemma *sskip-elim* :
$\sigma \in \mathit{SSkip} \longleftrightarrow \mathit{intlen}\ \sigma = 1$
using *sskip-def smore-fusion-smore*
by (*metis One-nat-def Suc-lessI Un-iff less-numeral-extra*(*4*) *sempty-elim smore-def smore-elim snot-elim zero-neq-one*)

lemma *spower-elim-zero* :

$\sigma \in \mathit{SPower}\ X\ 0 \longleftrightarrow \sigma \in \mathit{SEmpty}$
by *simp*

lemma *spower-elim-suc* :
$\sigma \in \mathit{SPower}\ X\ (\mathit{Suc}\ n) \longleftrightarrow \sigma \in (X \cap \mathit{SMore}){\cdot}(\mathit{SPower}\ X\ n)$
by *simp*

lemma *spower-elim-suc-1* :
$\sigma \in (X \cap \mathit{SMore}){\cdot}(\mathit{SPower}\ X\ n) \longleftrightarrow$
$(\exists \sigma 1\ \sigma 2.\ \sigma = \mathit{fuse}\ \sigma 1\ \sigma 2 \wedge \sigma 1 \in X \wedge \mathit{intlen}\ \sigma 1 > 0 \wedge \sigma 2 \in (\mathit{SPower}\ X\ n) \wedge$
$\mathit{intlast}\ \sigma 1 = \mathit{intfirst}\ \sigma 2)$
by (*meson IntD1 IntD2 IntI smore-elim fusion-iff*)

lemma *sstar-elim* :
$\sigma \in \mathit{SStar}\ X \longleftrightarrow (\exists\ n.\ \sigma \in \mathit{SPower}\ X\ n)$
by (*simp add*: *sstar-def*)

lemma *sstar-elim-1* :
$(\exists\ n\ .\ \sigma \in \mathit{SPower}\ X\ n) \longleftrightarrow$
$(\ \sigma \in \mathit{SPower}\ X\ 0 \vee (\exists\ n.\ \sigma \in \mathit{SPower}\ X\ (\mathit{Suc}\ n)))$
by (*metis not0-implies-Suc*)

lemma *spower-suc* :
$(\exists\ n.\ \sigma \in \mathit{SPower}\ X\ (\mathit{Suc}\ n)) \longleftrightarrow$
$(\exists\ n.\ \sigma \in (X \cap \mathit{SMore}){\cdot}(\mathit{SPower}\ X\ n))$
by *simp*

lemma *spower-suc-1* :
$(\exists\ n.\ \sigma \in (X \cap \mathit{SMore}){\cdot}(\mathit{SPower}\ X\ n)) \longleftrightarrow$
$\sigma \in (X \cap \mathit{SMore}){\cdot}(\mathit{SStar}\ X)$
by (*metis fusion-iff sstar-elim*)

lemma *sstar-eqv* :
$\sigma \in \mathit{SStar}\ X \longleftrightarrow$
$(\ \sigma \in \mathit{SEmpty} \vee \sigma \in (X \cap \mathit{SMore}){\cdot}(\mathit{SStar}\ X))$
by (*metis spower.simps*(*1*) *spower-elim-suc spower-suc-1 sstar-elim sstar-elim-1*)

lemma *spower-sskip-elim* :
$(\sigma \in \mathit{SPower}\ \mathit{SSkip}\ n) \longleftrightarrow \mathit{intlen}\ \sigma = n$
by (*induct n arbitrary*: σ, *simp add*: *sempty-elim*, *smt chop-fuse diff-Suc-1 interval-fuse-intlen*
more-d-def more-defs next-d-def plus-1-eq-Suc skip-defs spower-elim-suc
spower-elim-suc-1 sskip-elim zero-less-Suc zero-less-one)

lemma *srev-elim*:
$\sigma \in (\mathit{SRev}\ X) \longleftrightarrow \mathit{intrev}\ \sigma \in X$
by (*smt interval-rev-rev-ident mem-Collect-eq reverse-def*)

8.4 Algebraic Laws

8.4.1 Commutative Additive Monoid

lemma *UnionCommute*:
$(X\text{::}'a\ intervals) \cup Y = Y \cup X$
by (*simp add*: *Un-commute*)

lemma *UnionSFalse*:
$X \cup SFalse = X$
by (*simp add*: *sfalse-def*)

lemma *UnionAssoc*:
$(X\text{::}'a\ intervals) \cup (Y \cup Z) = (X \cup Y) \cup Z$
by (*simp add*: *sup-assoc*)

8.4.2 Boolean algebra

lemma *Huntington*:
$(X\text{::}'a\ intervals) = -(-X \cup -Y) \cup -(-X \cup Y)$
by *auto*

lemma *Morgan*:
$(X\text{::}'a\ intervals) \cap Y = -(-X \cup -Y)$
by *auto*

— identities

lemma *STrueTop*:
$STrue = X \cup -X$
by (*simp add*: *strue-def*)

lemma *SFalseBottom*:
$SFalse = X \cap -X$
by (*simp add*: *sfalse-def*)

8.4.3 multiplicative monoid

lemma *FusionSEmptyL* :
$SEmpty \cdot X = X$
using *fusion-iff-1 set-eqI*[*of SEmpty·X X*]
by (*metis interval-intlen-gr-zero interval-prefix-length-good interval-suffix-zero sempty-elim*)

lemma *FusionSEmptyR* :
$X \cdot SEmpty = X$
using *fusion-iff-1 set-eqI*[*of X·SEmpty X*]
by (*metis add-cancel-right-right fusion-iff interval-fuse-intlen interval-fuse-pref-suf*
interval-intlast-intfirst interval-prefix-fuse interval-prefix-intlen sempty-elim)

lemma *FusionAssoc* :
$X\cdot(Y\cdot Z) = (X\cdot Y)\cdot Z$
using *set-eqI*[*of X·(Y·Z) (X·Y)·Z*]

by (*smt fusion-iff interval-intfirst-fuse interval-FusionAssoc interval-intlast-fuse*)

— left and right distributivity

lemma *FusionUnionDistL*:
$(X \cup Y) \cdot Z = (X \cdot Z) \cup (Y \cdot Z)$
using *fusion-iff set-eqI*[*of* $(X \cup Y) \cdot Z$ $(X \cdot Z) \cup (Y \cdot Z)$]
by (*metis* (*no-types*, *lifting*) *sor-elim*)

lemma *FusionUnionDistR*:
$X \cdot (Y \cup Z) = (X \cdot Y) \cup (X \cdot Z)$
using *fusion-iff set-eqI*[*of* $X \cdot (Y \cup Z)$ $(X \cdot Y) \cup (X \cdot Z)$]
by (*metis* (*no-types*, *lifting*) *sor-elim*)

— left and right annihilation
lemma *SFalseFusion*:
SFalse · *X* = *SFalse*
by (*simp add*: *fusion-def sfalse-def*)

lemma *FusionSFalse*:
X·*SFalse* = *SFalse*
by (*simp add*: *fusion-def sfalse-def*)

— idempotency
lemma *UnionIdem*:
(*X*::'*a intervals*) ∪ *X* = *X*
by *simp*

8.4.4 Subsumption order

lemma *Subsumption*:
((*X* :: '*a intervals*) ⊆ *Y*) = (*X* ∪ *Y* = *Y*)
by *auto*

8.4.5 Helper lemmas

lemma *FusionRuleR*:
assumes $X \subseteq Y$
shows $Z \cdot X \subseteq Z \cdot Y$
using *assms FusionUnionDistR* **by** (*metis Subsumption*)

lemma *FusionRuleL*:
assumes $X \subseteq Y$
shows $X \cdot Z \subseteq Y \cdot Z$
using *assms* **by** (*metis FusionUnionDistL subset-Un-eq*)

lemma *spower-commutes*:
(*X* ∩ *SMore*)·(*SPower X n*) = (*SPower X n*)·(*X* ∩ *SMore*)
by (*induct n*, *simp add*: *FusionSEmptyL FusionSEmptyR*, *simp add*: *FusionAssoc*)

lemma *fusion-inductl*:

assumes *Y ∪ X·Z ⊆ Z*
shows *(SPower X n)·Y ⊆ Z*
using *assms*
by (*induct n*, *simp add*: *FusionSEmptyL*, *smt IntD1 UnI2 FusionAssoc fusion-iff pwr-Suc subset-eq*)

lemma *fusion-inductr*:
assumes *Y ∪ Z·X ⊆ Z*
shows *Y·(SPower X n) ⊆ Z*
using *assms*
by (*induct n*, *simp add*: *FusionSEmptyR*, *smt abel-semigroup.commute FusionAssoc FusionUnionDistL FusionUnionDistR le-iff-sup pwr-Suc spower-commutes sup.abel-semigroup-axioms sup-assoc sup-inf-absorb*)

lemma *sstar-contl*:
Y · (SStar X) = (⋃n. Y · (SPower X n))
using *set-eqI*[*of Y · (SStar X) (⋃n. Y · (SPower X n))*]
by (*smt UN-iff fusion-iff sstar-def*)

lemma *sstar-contr*:
(SStar X)·Y = (⋃n. (SPower X n)·Y)
using *set-eqI*[*of (SStar X)·Y (⋃n. (SPower X n)·Y)*]
by (*smt UN-iff fusion-iff sstar-def*)

8.4.6 Kleene Algebra

— left unfold
lemma *UnfoldL*:
SEmpty ∪ X·(SStar X) = (SStar X)
proof −
have *1*: *(SStar X) = SEmpty ∪ (X ∩ SMore)·(SStar X)*
  by (*meson Un-iff set-eqI sstar-eqv*)
have *2*: *(X ∩ SMore)·(SStar X) ⊆ X·(SStar X)*
  by (*simp add*: *FusionRuleL*)
have *3*: *(SStar X) ⊆ SEmpty ∪ X·(SStar X)*
  using *1 2* **by** *blast*
have *4*: *SEmpty ⊆ (SStar X)*
  using *1* **by** *auto*
have *5*: *X ⊆ SEmpty ∪ (X ∩ SMore)*
  by (*simp add*: *Un-Int-distrib smore-def*)
have *6*: *X·(SStar X) ⊆ (SStar X) ∪ (X ∩ SMore)·(SStar X)*
  using *5* **by** (*metis FusionRuleL FusionUnionDistL FusionSEmptyL*)
have *7*: *(SStar X) ⊆ SEmpty ∪ (X ∩ SMore)·(SStar X)*
  using *1* **by** *auto*
have *8*: *X·(SStar X) ⊆ SEmpty ∪ (X ∩ SMore)·(SStar X)*
  using *6 7* **by** *blast*
hence *9*: *X·(SStar X) ⊆ (SStar X)*
  using *1* **by** *auto*
have *10*: *SEmpty ∪ X·(SStar X) ⊆ (SStar X)*
  using *9 4* **by** *simp*
from *3 10* **show** *?thesis* **by** *auto*

```
qed


— Left induction
lemma SStarInductL:
 assumes Y ∪ X·Z ⊆ Z
 shows   (SStar X)·Y ⊆ Z
by (metis UN-least assms fusion-inductl sstar-contr)

— Right induction
lemma SStarInductR:
 assumes Y ∪ Z·X ⊆ Z
 shows   Y·(SStar X) ⊆ Z
using sstar-contl assms fusion-inductr by blast
```

8.4.7 ITL specific Laws

```
lemma PwrFusionInterL:
 ((((SPower SSkip n) ∩ X)·V) ∩ (((SPower SSkip n) ∩ Y)·W)) =
  (((SPower SSkip n) ∩ X ∩ Y)·(V ∩ W))
using set-eqI[of ((((SPower SSkip n) ∩ X)·V) ∩ (((SPower SSkip n) ∩ Y)·W))
                 (((SPower SSkip n) ∩ X ∩ Y)·(V ∩ W)) ]
using fusion-iff
by (smt interval-prefix-fuse interval-suffix-fuse sand-elim spower-sskip-elim)

lemma PwrFusionInterR:
 ((V·((SPower SSkip n) ∩ X)) ∩ ((W·((SPower SSkip n) ∩ Y)))) =
   ((V ∩ W)·((SPower SSkip n) ∩ X ∩ Y))
using set-eqI[of ((V·((SPower SSkip n) ∩ X)) ∩ ((W·((SPower SSkip n) ∩ Y))))
                 ((V ∩ W)·((SPower SSkip n) ∩ X ∩ Y)) ]
using fusion-iff
by (smt add-right-imp-eq interval-fuse-intlen interval-prefix-fuse
        interval-suffix-fuse sand-elim spower-sskip-elim)

lemma SSkipFusionImpSMore:
  SSkip·STrue ⊆ SMore
by (metis fusion-iff gr0I interval-fuse-intlen nat.distinct(1) plus-1-eq-Suc
          smore-elim sskip-elim subsetI)

lemma SStarSkip:
  (SStar SSkip) = STrue
using set-eqI[of (SStar SSkip) STrue]
by (simp add: strue-def spower-sskip-elim sstar-elim)
```

8.5 Derived Laws

8.5.1 Helper Lemmas

```
lemma B01:
 assumes (X:: 'a intervals) ⊆ Y
 shows   −Y ⊆ −X
```

using *assms* **by** *auto*

lemma *B04*:
$((X{::}\ \mathit{'a\ intervals}) = Y\ ) \longleftrightarrow (X \subseteq Y) \wedge (Y \subseteq X)$
by *auto*

lemma *B09*:
assumes $-X \cup Y = \mathit{STrue}$
shows $(X{::}\ \mathit{'a\ intervals}) \subseteq Y$
using *assms* **using** *strue-def* **by** *auto*

lemma *B20*:
$(X{::}\ \mathit{'a\ intervals}) \subseteq Y \cup Z \longleftrightarrow X \cap -Y \subseteq Z$
by *auto*

lemma *B28*:
$((X{::}\ \mathit{'a\ intervals}) \cap Y) \cup (X \cap Z) = X \cap (Y \cup Z)$
by *auto*

lemma *CH01*:
$\mathit{STrue}\cdot\mathit{STrue} = \mathit{STrue}$
by (*metis FusionSEmptyR FusionUnionDistR Int-commute SStarSkip STrueTop UnfoldL inf-sup-absorb*)

lemma *CH07*:
$(((\mathit{SSkip} \cap X)\cdot V) \cap ((\mathit{SSkip} \cap Y)\cdot W)) = ((\mathit{SSkip} \cap X \cap Y)\cdot(V \cap W))$
using *PwrFusionInterL*[*of 1 X V Y W*]
by (*simp add*: *FusionSEmptyR inf-commute smore-def sskip-def*)

lemma *CH08*:
$((V\cdot(\mathit{SSkip} \cap X)) \cap ((W\cdot(\mathit{SSkip} \cap Y)))) = ((V \cap W)\cdot(\mathit{SSkip} \cap X \cap Y))$
using *PwrFusionInterR*[*of V 1 X W Y*]
by (*simp add*: *FusionSEmptyR inf-commute smore-def sskip-def*)

lemma *CH09*:
$(((X \cap \mathit{SEmpty})\cdot V) \cap ((Y \cap \mathit{SEmpty})\cdot W)) = (((X \cap Y) \cap \mathit{SEmpty})\cdot(V \cap W))$
using *PwrFusionInterL*[*of 0 X V Y W*]
by (*metis* (*no-types*, *lifting*) *inf-assoc inf-commute pwr-0*)

lemma *CH10*:
$((V\cdot(X \cap \mathit{SEmpty})) \cap ((W\cdot(Y \cap \mathit{SEmpty})))) = ((V \cap W)\cdot((X \cap Y) \cap \mathit{SEmpty}))$
using *PwrFusionInterR*[*of V 0 X W Y*]
by (*metis* (*no-types*, *lifting*) *inf-assoc inf-commute pwr-0*)

lemma *ST13*:
$((X \cap \mathit{SEmpty})\cdot Z) \cap ((Y \cap \mathit{SEmpty})\cdot Z) = ((X \cap Y) \cap \mathit{SEmpty})\cdot Z$
by (*simp add*: *CH09*)

lemma *ST15*:
$(\mathit{SStar}\ (X \cap \mathit{SEmpty})) = \mathit{SEmpty}$
by (*metis FusionSEmptyL inf .right-idem inf-le2 UnfoldL*

SStarInductR sup.orderE sup-inf-absorb)

lemma *ST21*:
$((-X) \cap \mathit{SEmpty}) \cup (X \cap \mathit{SEmpty}) = \mathit{SEmpty}$
by *blast*

lemma *ST24*:
$(\mathit{SInit}\ X) \cap (\mathit{SInit}\ Y) = (\mathit{SInit}\ (X \cap Y))$
by (*simp add*: *ST13 sinit-def*)

lemma *ST25*:
$(\mathit{SInit}\ \mathit{STrue}) = \mathit{STrue}$
by (*simp add*: *sinit-def strue-def FusionSEmptyL*)

lemma *ST26*:
$(\mathit{SInit}\ (-X)) \cup (\mathit{SInit}\ X) = \mathit{STrue}$
by (*metis Compl-disjoint2 ST21 ST25 Un-Int-distrib compl-bot-eq FusionUnionDistL sinit-def strue-def sup-bot.right-neutral sup-top-right*)

lemma *ST28*:
$(\mathit{SDi}\ (\mathit{SInit}\ X)) = (\mathit{SInit}\ X)$
by (*metis compl-bot-eq FusionAssoc FusionUnionDistR FusionSEmptyR sdi-def sinit-def strue-def sup-top-right UnionCommute*)

lemma *ST33*:
$(\mathit{STrue} \cap \mathit{SEmpty}) \cdot \mathit{SEmpty} = \mathit{SEmpty}$
by (*simp add*: *strue-def FusionSEmptyL*)

lemma *ST36*:
$(\mathit{SInit}\ (-X)) \subseteq -(\mathit{SInit}\ X)$
by (*metis Compl-disjoint ST24 compl-bot-eq disjoint-eq-subset-Compl double-complement inf.coboundedI2 inf.orderE sfalse-def SFalseFusion sinit-def strue-def*)

lemma *ST37*:
$-(\mathit{SInit}\ X) \subseteq (\mathit{SInit}\ (-X))$
using *B09 ST26* **by** *auto*

lemma *ST38*:
$-(\mathit{SInit}\ X) = (\mathit{SInit}\ (-X))$
using *ST37 ST36* **by** *auto*

lemma *ST47*:
$X \cup Y \cdot X = (\mathit{SEmpty} \cup Y) \cdot X$
by (*simp add*: *FusionUnionDistL FusionSEmptyL*)

lemma *SStar01*:
assumes $X \cdot (\mathit{SStar}\ Y) \cup \mathit{SEmpty} \subseteq (\mathit{SStar}\ Y)$
shows $(\mathit{SStar}\ X) \subseteq (\mathit{SStar}\ Y)$
using *assms*
by (*metis Un-commute FusionSEmptyR SStarInductL*)

lemma *SStar03*:
$(\mathit{SStar}\ X)\cdot(\mathit{SStar}\ X) \subseteq (\mathit{SStar}\ X)$
by (*metis SStarInductL Un-absorb UnfoldL sup.absorb-iff1 sup.right-idem*)

lemma *SStar05*:
assumes $((\mathit{SStar}\ X)\cdot(\mathit{SStar}\ X)) \cup \mathit{SEmpty} \subseteq (\mathit{SStar}\ X)$
shows $(\mathit{SStar}\ (\mathit{SStar}\ X)) \subseteq (\mathit{SStar}\ X)$
using *assms*
by (*simp add*: *SStar01*)

lemma *SStar12*:
$(\mathit{SEmpty} \cup (X\cdot(\mathit{SStar}\ X))) \subseteq (\mathit{SStar}\ X)$
using *UnfoldL* **by** *blast*

lemma *SStar06*:
$((\mathit{SStar}\ X)\cdot(\mathit{SStar}\ X)) \cup \mathit{SEmpty} \subseteq (\mathit{SStar}\ X)$
using *SStar03 SStar12* **by** *force*

lemma *SStar07*:
$(\mathit{SStar}\ X) \subseteq (\mathit{SStar}\ (\mathit{SStar}\ X))$
by (*metis FusionUnionDistR FusionSEmptyR Subsumption Un-commute UnfoldL ST47 sup.right-idem*)

lemma *SStar08*:
$(\mathit{SStar}\ X) = (\mathit{SStar}\ (\mathit{SStar}\ X))$
by (*meson B04 SStar05 SStar06 SStar07*)

lemma *SStar15*:
$\mathit{SEmpty} \subseteq (\mathit{SStar}\ \mathit{SSkip})$
by (*simp add*: *SStarSkip strue-def*)

lemma *SStar16*:
$\mathit{SSkip} \subseteq (\mathit{SStar}\ \mathit{SSkip})$
by (*simp add*: *SStarSkip strue-def*)

lemma *SStar22*:
$(\mathit{SEmpty} \cap X)\cdot(\mathit{SStar}\ (\mathit{SEmpty} \cap X)) = (\mathit{SEmpty} \cap X)$
by (*metis ST15 FusionSEmptyR inf-commute*)

lemma *SStar23*:
$(\mathit{SStar}\ (\mathit{SEmpty} \cap X)) = \mathit{SEmpty}$
using *SStar22 UnfoldL* **by** *auto*

lemma *SStar25*:
$(\mathit{SStar}\ \mathit{STrue}) = \mathit{STrue}$
by (*metis SStar08 SStarSkip*)

lemma *SStar28*:
$(\mathit{SStar}\ X)\cdot X \subseteq X\cdot(\mathit{SStar}\ X)$
by (*metis B04 FusionUnionDistR FusionSEmptyR UnfoldL SStarInductL*)

```
lemma SStar29:
  X·(SStar X) ⊆ (SStar X)·X
by (metis B04 SStar28 SStarInductR UnfoldL FusionRuleL ST47 sup.mono)

lemma SStar17:
  (SStar SSkip)·SSkip ⊆ SSkip·(SStar SSkip)
by (simp add: SStar28)

lemma SStar18:
  SSkip·(SStar SSkip) ⊆ (SStar SSkip)·SSkip
by (simp add: SStar29)

lemma SStar19:
  (SStar SSkip)·SSkip = SSkip·(SStar SSkip)
using SStar17 SStar18 by auto

lemma SStar30:
  X·(SStar X) = (SStar X)·X
using SStar28 SStar29 by auto

lemma SStar34:
 assumes SEmpty ∪ (X ∪ Y)·((SStar X)·(SStar (Y·(SStar X))) ) ⊆ (SStar X)·(SStar (Y·(SStar X)))
 shows  (SStar (X ∪ Y)) ⊆ (SStar X)·(SStar (Y·(SStar X)))
by (metis assms FusionSEmptyR SStarInductL)

lemma SStar35:
 SEmpty ∪ (X ∪ Y)·((SStar X)·(SStar (Y·(SStar X))) ) ⊆ (SStar X)·(SStar (Y·(SStar X)))
by (smt B04 FusionAssoc FusionUnionDistL FusionSEmptyL UnfoldL UnionAssoc UnionCommute)

lemma SStar36:
  (SStar (X ∪ Y)) ⊆ (SStar X)·(SStar (Y·(SStar X)))
using SStar34 SStar35 by blast

lemma SStar46:
  (SStar X)·(SStar (Y·(SStar X))) ⊆ (SStar (X ∪ Y))
proof −
 have (SEmpty ∪ SStar (X ∪ Y) · Y) · SStar X ⊆ SStar (X ∪ Y)
 by (metis (no-types) FusionUnionDistR SStar12 SStar30 SStarInductR sup.bounded-iff)
 then show ?thesis by (simp add: SStarInductR ST47 FusionAssoc)
qed

lemma SStar47:
  (SStar Z) = (SStar (Z ∩ SMore))
proof −
 have 1: (SStar Z) = (SStar ((SEmpty ∩ Z) ∪ (SMore ∩ Z)))
     by (metis Int-Un-distrib2 compl-bot-eq inf-top.left-neutral smore-def strue-def STrueTop)
 have 2: (SStar ((SEmpty ∩ Z) ∪ (SMore ∩ Z))) =
          (SStar (SEmpty ∩ Z))·(SStar ((SMore ∩ Z)·(SStar (SEmpty ∩ Z))))
     by (simp add: SStar36 SStar46 subset-antisym)
 have 3: (SStar (SEmpty ∩ Z))·(SStar ((SMore ∩ Z)·(SStar (SEmpty ∩ Z)))) =
         (SStar (Z ∩ SMore))
     by (simp add: FusionSEmptyL FusionSEmptyR SStar23 inf-commute)
 from 1 2 3 show ?thesis by auto
qed

lemma SStar48:
  (SStar SMore) = STrue
by (metis Compl-Un Compl-disjoint2 SStar25 SStar47 ST21 ST33 FusionSEmptyR
         inf.right-idem smore-def strue-def)

lemma SStar50:
assumes SSkip·((−X) ∪ ((SStar SSkip)·(X ∩ (SSkip·(−X))))) ∪ (−X)
        ⊆ (−X) ∪ (SStar SSkip)·(X ∩ (SSkip·(−X)))
shows  ((SStar SSkip)·(−X)) ⊆ ((−X) ∪ ((SStar SSkip)·(X ∩ (SSkip·(−X)))))
using SStarInductL assms by blast

lemma SStar51:
  SSkip·((−X) ∪ ((SStar SSkip)·(X ∩ (SSkip·(−X))))) ∪ (−X)
        ⊆ (−X) ∪ (SStar SSkip)·(X ∩ (SSkip·(−X)))
by (smt B20 double-compl FusionAssoc FusionUnionDistR inf-commute le-sup-iff
       UnfoldL ST47 Subsumption sup-ge2)

lemma SStar52:
  (SStar X) ⊆ SEmpty ∪ (X ∩ SMore)·(SStar X)
by (metis B04 SStar47 UnfoldL)

lemma SStar53:
  SEmpty ∪ (X ∩ SMore)·(SStar X) ⊆ (SStar X)
by (metis SStar12 SStar47)

lemma BD45:
 (SBi ((−X) ∪ X1)) ∩ (X·Y) ⊆ X1·Y
proof −
 have 1: (SBi ((−X) ∪ X1)) = −(−((−X) ∪ X1)·(Y ∪ (−Y)))
     by (metis sbi-def sdi-def STrueTop)
 have 2: −(−((−X) ∪ X1)·(Y ∪ (−Y))) ∩ (X·Y) ⊆
          −(−((−X) ∪ X1)·(Y)) ∩ (X·Y)
     by (smt B01 B28 FusionUnionDistR inf-commute sup.absorb-iff1 sup-ge1)
 have 3: −(−((−X) ∪ X1)·(Y)) ∩ (X·Y) ⊆  (((−X) ∪ X1) ∩ X)·Y
     by (smt B09 Compl-disjoint2 FusionUnionDistL Huntington Morgan STrueTop UnionAssoc
            UnionCommute compl-inf sup-bot.left-neutral)
 have 4:  (((−X) ∪ X1) ∩ X)·Y ⊆ X1·Y
     by (metis B20 double-compl FusionRuleL inf.right-idem inf-le1)
 from 1 2 3 4 show ?thesis by blast
qed

lemma BD46:
 (SAlways ((−Y) ∪ Y1)) ∩ (X1·Y) ⊆ (X1·Y1)
proof −
```

```
 have 1: (SAlways ((−Y) ∪ Y1)) = −((X1 ∪ (−X1))·(−((−Y) ∪ Y1 ) ) )
     by (metis salways-def ssometime-def STrueTop)
 have 2: −((X1 ∪ (−X1))·(−((−Y) ∪ Y1 ) ) ) ∩ (X1·Y) ⊆
         −(X1·(−((−Y) ∪ Y1 ) ) ) ∩  (X1·Y)
      by (smt B01 B28 FusionUnionDistL inf-commute sup.absorb-iff2 sup-ge1)
 have 3: −(X1·(−((−Y) ∪ Y1 ) ) ) ∩  (X1·Y) ⊆ X1·(((−Y) ∪ Y1) ∩ Y)
      by (metis (no-types, lifting) B20 B04 compl-inf FusionUnionDistR Huntington
              Morgan Subsumption sup-ge1 UnionCommute)
 have 4: X1·(((−Y) ∪ Y1) ∩ Y) ⊆ (X1·Y1)
     by (metis B20 double-compl FusionRuleR inf.right-idem inf-le1)
 from 1 2 3 4 show ?thesis by blast
qed
```

8.5.2 ITL Axioms derived

```
lemma SBoxGen:
 assumes  X=STrue
 shows    (SAlways X) = STrue
using assms
by (metis double-compl FusionSFalse salways-def sfalse-def ssometime-def strue-def)

lemma SBiGen:
 assumes  X=STrue
 shows    (SBi X) = STrue
using assms
by (metis double-compl sbi-def sdi-def sfalse-def SFalseFusion strue-def)

lemma SMP:
 assumes X ⊆ Y
 assumes X=STrue
 shows    Y=STrue
using assms(1) assms(2)
using strue-def by blast

lemma SChopAssoc:
 X·(Y·Z) = (X·Y)·Z
by (simp add: FusionAssoc)

lemma SOrChopImp:
  (X ∪ Y)·Z ⊆ (X·Z) ∪ (Y·Z)
by (simp add: FusionUnionDistL)

lemma SChopOrImp:
  X·(Y ∪ Z) ⊆ (X·Y) ∪ (X·Z)
by (simp add: FusionUnionDistR)

lemma SEmptyChop:
  SEmpty·X = X
by (simp add: FusionSEmptyL)
```

lemma *SChopEmpty*:
$X \cdot \mathit{SEmpty} = X$
by (*simp add*: *FusionSEmptyR*)

lemma *SStateImpBi*:
$(\mathit{SInit}\ X) \subseteq (\mathit{SBi}\ (\mathit{SInit}\ X))$
by (*simp add*: *ST28 ST38 sbi-def*)

lemma *SNextImpNotNextNot*:
$(\mathit{SNext}\ X) \subseteq -(\mathit{SNext}\ (-X))$
proof −
have *1*: $((\mathit{SNext}\ X) \subseteq -(\mathit{SNext}\ (-X))) = (((\mathit{SNext}\ X) \cap (\mathit{SNext}\ (-X))) \subseteq \mathit{SFalse})$
by (*simp add*: *disjoint-eq-subset-Compl sfalse-def*)
have *2*: $((\mathit{SNext}\ X) \cap (\mathit{SNext}\ (-X))) = \mathit{SSkip} \cdot (X \cap (-X))$
by (*metis CH07 SStar16 inf.orderE snext-def*)
have *3*: $(\mathit{SSkip}) \cdot (X \cap (-X)) = \mathit{SSkip} \cdot \mathit{SFalse}$
by (*simp add*: *sfalse-def*)
have *4*: $\mathit{SSkip} \cdot \mathit{SFalse} = \mathit{SFalse}$ **by** (*simp add*: *FusionSFalse*)
from *1 2 3 4* **show** *?thesis* **by** *auto*
qed

lemma *SBiBoxChopImpChop*:
$(\mathit{SBi}\ ((-X) \cup X1)) \cap (\mathit{SAlways}\ ((-Y) \cup Y1)) \cap (X \cdot Y) \subseteq (X1 \cdot Y1)$
using *BD45 BD46* **by** *blast*

lemma *SBoxInduct*:
$(\mathit{SAlways}\ (-X \cup (\mathit{SWnext}\ X))) \cap X \subseteq (\mathit{SAlways}\ X)$
proof −
have *1*: $((\mathit{SAlways}\ (-X \cup (\mathit{SWnext}\ X))) \cap X \subseteq (\mathit{SAlways}\ X)) =$
$((\mathit{SSometime}\ (-X)) \subseteq ((-X) \cup (\mathit{SSometime}\ (X \cap (\mathit{SNext}\ (-X))))))$
by (*smt Compl-subset-Compl-iff compl-sup double-compl inf-commute salways-def snext-def swnext-def*)
have *2*: $((\mathit{SSometime}\ (-X)) \subseteq ((-X) \cup (\mathit{SSometime}\ (X \cap (\mathit{SNext}\ (-X)))))) =$
$(((\mathit{SStar}\ \mathit{SSkip}) \cdot (-X)) \subseteq ((-X) \cup ((\mathit{SStar}\ \mathit{SSkip}) \cdot (X \cap (\mathit{SSkip} \cdot (-X))))))$
by (*simp add*: *SStarSkip snext-def ssometime-def*)
have *3*: $(((\mathit{SStar}\ \mathit{SSkip}) \cdot (-X)) \subseteq ((-X) \cup ((\mathit{SStar}\ \mathit{SSkip}) \cdot (X \cap (\mathit{SSkip} \cdot (-X))))))$
using *SStar51 SStar50* **by** *blast*
from *1 2 3* **show** *?thesis* **by** *auto*
qed

lemma *SChopstarEqv*:
$(\mathit{SStar}\ X) = \mathit{SEmpty} \cup (X \cap \mathit{SMore}) \cdot (\mathit{SStar}\ X)$
using *SStar52 SStar53* **by** *blast*

8.6 Extra Laws

8.6.1 Boolean Laws

lemma *B02*:
assumes $-Y \subseteq -X$
shows $(X :: \mathit{'a\ intervals}) \subseteq Y$

```
using assms by auto

lemma B03:
 ((X:: 'a intervals) = Y ) ⟷ (−X = −Y)
by auto

lemma B05:
 assumes (X:: 'a intervals) ∪ Y ⊆ Z
 shows   X ⊆ Z ∧ Y ⊆ Z
using assms by auto

lemma B06:
 assumes X ⊆ Z ∧ Y ⊆ Z
 shows   (X:: 'a intervals) ∪ Y ⊆ Z
using assms by auto

lemma B07:
 (X:: 'a intervals) ∪ Y ⊆ Z ⟷
  X ⊆ Z ∧ Y ⊆ Z
by auto

lemma B08:
 assumes (X:: 'a intervals) ⊆ Y
 shows   −X ∪ Y = STrue
using assms
using strue-def by auto

lemma B10:
 (X:: 'a intervals) ⊆ Y ⟷ −X ∪ Y = STrue
using strue-def by auto

lemma B11:
 assumes (X:: 'a intervals) ⊆ Y
 shows    X ∩ −Y = SFalse
using assms sfalse-def by auto

lemma B12:
 assumes  X ∩ −Y = SFalse
 shows   (X:: 'a intervals) ⊆ Y
using assms sfalse-def by auto

lemma B13:
 (X:: 'a intervals) ⊆ Y ⟷ X ∩ −Y = SFalse
using sfalse-def by auto

lemma B14:
 assumes (X:: 'a intervals) ⊆ Y
 shows   X ∩ Y =X
using assms by auto
```

```
lemma B15:
 assumes (X:: 'a intervals) ⊆ Y ∩ Z
 shows   X ⊆ Y ∧ X ⊆ Z
using assms by auto

lemma B16:
 assumes X ⊆ Y ∧ X ⊆ Z
 shows   (X:: 'a intervals) ⊆ Y ∩ Z
using assms by auto

lemma B17:
 (X:: 'a intervals) ⊆ Y ∩ Z ⟷ X ⊆ Y ∧ X ⊆ Z
by auto

lemma B18:
 assumes (X:: 'a intervals) ⊆ Y ∪ Z
 shows   X ∩ −Y ⊆ Z
using assms by auto

lemma B19:
 assumes X ∩ −Y ⊆ Z
 shows   (X:: 'a intervals) ⊆ Y ∪ Z
using assms by auto

lemma B21:
 (X:: 'a intervals) ∪ (Y ∩ Z) ⊆ (X ∪ Y) ∩ (X ∪ Z) ⟷
  X ∪ (Y ∩ Z) ⊆ (X ∪ Y) ∧ X ∪ (Y ∩ Z) ⊆ (X ∪ Z)
by auto

lemma B22:
   (X:: 'a intervals) ∪ (Y ∩ Z) ⊆ X ∪ Y
by auto

lemma B23:
  (X:: 'a intervals) ∪ (Y ∩ Z) ⊆ X ∪ Z
by auto

lemma B24:
 ((X:: 'a intervals) ∪ Y) ∩ (X ∪ Z) ⊆ X ∪ (Y ∩ Z) ⟷
  ((X ∪ Y) ∩ (X ∪ Z)) ∩ −X ⊆ Y ∩ Z
by auto

lemma B25:
 (((X:: 'a intervals) ∪ Y) ∩ (X ∪ Z)) ∩ −X ⊆ Y ∩ Z ⟷
  ((X ∪ Y) ∩ (X ∪ Z)) ∩ −X ⊆ Y ∧
  ((X ∪ Y) ∩ (X ∪ Z)) ∩ −X ⊆ Z
by auto

lemma B26:
 (((X:: 'a intervals) ∪ Y) ∩ (X ∪ Z)) ∩ −X ⊆ Y
```

by *auto*

lemma *B27*:
$(((X{::}\ 'a\ \mathit{intervals}) \cup Y) \cap (X \cup Z)) \cap -X \subseteq Z$
by *auto*

lemma *B29*:
$(X{::}\ 'a\ \mathit{intervals}) \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$
by *auto*

8.6.2 Chop

lemma *CH02*:
$X{\cdot}Y \cap -(X{\cdot}Z) \subseteq X{\cdot}(Y \cap -Z)$
by (*metis B20 FusionRuleR FusionUnionDistR inf.right-idem inf-le1*)

lemma *CH03*:
$X{\cdot}Y \cap -(Z{\cdot}Y) \subseteq (X \cap -Z){\cdot}Y$
by (*metis B20 FusionRuleL FusionUnionDistL inf.right-idem inf-le1*)

lemma *CH04*:
$X{\cdot}Y \cap -(X{\cdot}{-Z}) \subseteq X{\cdot}(Y \cap Z)$
using *CH02* **by** *fastforce*

lemma *CH05*:
$X{\cdot}Y \cap -(-Z{\cdot}Y) \subseteq (X \cap Z){\cdot}Y$
using *CH03* **by** *fastforce*

lemma *CH06*:
assumes $X \subseteq X1$
$Y \subseteq Y1$
shows $X{\cdot}Y \subseteq X1{\cdot}Y1$
using *assms*(1) *assms*(2)
by (*smt FusionUnionDistL FusionUnionDistR UnionAssoc le-iff-sup*)

lemma *CH11*:
$((X \cap (\mathit{SPower}\ \mathit{SSkip}\ n)){\cdot}\mathit{STrue}) \cap ((\mathit{SPower}\ \mathit{SSkip}\ n){\cdot}Y) = (X \cap (\mathit{SPower}\ \mathit{SSkip}\ n)){\cdot}Y$
by (*smt PwrFusionInterL compl-bot-eq inf.absorb2 inf-commute strue-def top-greatest*)

lemma *CH12*:
$(\mathit{STrue}{\cdot}(X \cap (\mathit{SPower}\ \mathit{SSkip}\ n))) \cap (Y{\cdot}(\mathit{SPower}\ \mathit{SSkip}\ n)) = (Y{\cdot}(X \cap (\mathit{SPower}\ \mathit{SSkip}\ n)))$
by (*metis PwrFusionInterR compl-bot-eq inf-commute inf-top.right-neutral strue-def*)

lemma *CH13*:
$(\mathit{SPower}\ \mathit{SSkip}\ n) \cdot (\mathit{SPower}\ \mathit{SSkip}\ m) = (\mathit{SPower}\ \mathit{SSkip}\ (n{+}m))$
proof
(*induct n arbitrary*: *m*)
case *0*
then show *?case* **by** (*simp add*: *FusionSEmptyL*)
next

case (*Suc n*)
then show *?case*
by (*metis FusionAssoc add-Suc pwr-Suc*)
qed

8.6.3 Next

lemma *N01*:
$(\mathit{SNext}\ \mathit{SEmpty}) = \mathit{SSkip}$
by (*simp add*: *FusionSEmptyR snext-def*)

lemma *N02*:
$(\mathit{SNext}\ \mathit{SFalse}) = \mathit{SFalse}$
by (*simp add*: *FusionSFalse snext-def*)

lemma *N03*:
$(\mathit{SNext}\ X) \cdot Y = (\mathit{SNext}\ (X \cdot Y))$
by (*simp add*: *snext-def FusionAssoc*)

lemma *N04*:
$(\mathit{SNext}\ (X \cup Y)) = (\mathit{SNext}\ X) \cup (\mathit{SNext}\ Y)$
by (*simp add*: *FusionUnionDistR snext-def*)

lemma *N05*:
$(\mathit{SNext}\ (X \cap Y)) = (\mathit{SNext}\ X) \cap (\mathit{SNext}\ Y)$
by (*metis CH07 SStar16 inf.orderE snext-def*)

lemma *N06*:
assumes $X \subseteq Y$
shows $(\mathit{SNext}\ X) \subseteq (\mathit{SNext}\ Y)$
using *assms*
by (*metis FusionUnionDistR Subsumption snext-def*)

lemma *N07*:
$(\mathit{SNext}\ ((-X) \cup Y)) = (\mathit{SNext}\ (-X)) \cup (\mathit{SNext}\ Y)$
by (*simp add*: *N04*)

lemma *N08*:
$\mathit{SMore} \subseteq \mathit{SSkip} \cdot \mathit{STrue}$
by (*smt B09 Morgan SStar15 SStarSkip UnfoldL compl-inf inf.absorb2 smore-def*)

lemma *N23*:
$(\mathit{SWprev}\ X) \subseteq (\mathit{SEmpty} \cup (\mathit{SPrev}\ X))$
by (*metis B09 FusionUnionDistL SStar30 SStarSkip STrueTop UnfoldL UnionAssoc abel-semigroup.commute double-compl sprev-def sup.abel-semigroup-axioms swprev-def*)

lemma *N24*:
$(\mathit{SEmpty}) \subseteq (\mathit{SWprev}\ X)$
by (*metis B10 B02 FusionRuleL SSkipFusionImpSMore SStar30 SStarSkip UnfoldL compl-bot-eq double-compl smore-def strue-def subset-antisym swprev-def top-greatest*)

```
lemma N25:
  (SPrev X) ⊆ (SWprev X)
proof −
 have 1: ((SPrev X) ⊆ (SWprev X) ) = (((SPrev X) ∩ (SPrev (−X)))⊆ SFalse )
      by (simp add: B10 sfalse-def sprev-def swprev-def)
 have 2: ((SPrev X) ∩ (SPrev (−X))) = (X ∩ (−X))·SSkip
      by (metis CH08 SStar16 inf.orderE sprev-def)
 have 3:  (X ∩ (−X))·SSkip = SFalse·SSkip
      by (simp add: sfalse-def)
 have 4: SFalse·SSkip = SFalse
      by (simp add: SFalseFusion)
 from 1 2 3 4 show ?thesis by auto
qed
```

```
lemma N26:
 (SWprev X) =  (SEmpty ∪  (SPrev X))
using N23 N24 N25 by blast
```

```
lemma N09:
  SSkip ∪ SMore·SSkip ⊆ SMore
proof −
 have 1:  SSkip ⊆ SMore by (simp add: smore-def sskip-def)
 have 2:  SMore·SSkip ⊆ SMore
 by (metis N26 UnionCommute compl-le-swap1 smore-def sup-ge2 swprev-def)
 from 1 2 show ?thesis by auto
qed
```

```
lemma N10:
 assumes SSkip ∪ SMore·SSkip ⊆ SMore
 shows   SSkip·(SStar SSkip) ⊆ SMore
using assms
using SStarInductR N09 by blast
```

```
lemma N11:
  SSkip · STrue ⊆ SMore
by (metis SStarSkip N09 N10)
```

```
lemma N12:
  (SNext X) = −(SWnext (−X))
by (simp add: snext-def swnext-def)
```

```
lemma N13:
  SMore·STrue = SMore
by (metis FusionAssoc N11 N08 SStar48 SStarSkip ST47 UnfoldL subset-antisym sup.right-idem)
```

```
lemma N14:
  STrue·SSkip ⊆ SMore
by (metis N11 SStar19 SStarSkip)
```

```
lemma N15:
  SMore ⊆ STrue·SSkip
by (metis N08 SStar19 SStarSkip)

lemma N19:
 (SWnext X) ⊆ (SEmpty ∪ (SNext X))
by (smt B02 B20 B09 FusionUnionDistR Morgan SStarSkip STrueTop UnfoldL compl-inf
        inf-sup-absorb snext-def swnext-def)

lemma N20:
  (SEmpty ) ⊆ (SWnext X)
proof −
 have 1: ((SEmpty ) ⊆ (SWnext X) ) = (−(SWnext X) ⊆ SMore)
      by (simp add: smore-def)
 have 2: (−(SWnext X) ⊆ SMore) = ((SNext (−X)) ⊆ SMore)
      by (simp add: snext-def swnext-def)
 have 3: (SNext (−X)) ⊆ SSkip·STrue
      by (metis FusionUnionDistR STrueTop snext-def sup.orderI sup.right-idem)
 hence 4: (SNext (−X)) ⊆ SMore  using SSkipFusionImpSMore by auto
 from 1 2 4 show ?thesis by auto
qed

lemma N21:
  (SEmpty ∪ (SNext X)) ⊆ (SWnext X)
using N20
by (metis B06 SNextImpNotNextNot snext-def swnext-def)

lemma N22:
 (SWnext X) = (SEmpty ∪ (SNext X))
using N21 N19 by blast

lemma N16:
  (SNext X) = SMore ∩ (SWnext X)
using N12 N22 smore-def by blast

lemma N17:
  (SWnext (X ∩ Y)) = (SWnext X) ∩ (SWnext Y)
by (simp add: N05 N22 Un-Int-distrib)

lemma N18:
  (SWnext (X ∪ Y)) = (SWnext X) ∪ (SWnext Y)
by (smt CH07 SStar16 compl-sup double-compl inf.orderE swnext-def)

lemma N27:
  (SNext ((−X) ∪ Y)) ⊆ (−(SNext X) ∪ (SNext Y))
by (metis N12 N16 N18 Un-Int-distrib double-compl sup-ge2 sup-left-idem)

lemma N28:
 (SPrev ((−X) ∪ Y)) ⊆ ( −(SPrev X) ∪ (SPrev Y))
by (metis B01 B05 B06 FusionUnionDistL Huntington N25 double-compl sprev-def sup-ge2 swprev-def)
```

lemma *N29*:
$(\mathit{SPrev}\ X) = -(\mathit{SWprev}\ (-X)\ )$
by (*simp add*: *sprev-def swprev-def*)

8.6.4 SInit

lemma *ST01*:
$(X \cap \mathit{SEmpty}) \cdot Y \subseteq Y$
by (*metis Int-commute FusionUnionDistL FusionSEmptyL le-iff-sup sup-inf-absorb UnionCommute*)

lemma *ST02*:
$(X \cap \mathit{SEmpty}) \cdot Y \subseteq (X \cap \mathit{SEmpty}) \cdot \mathit{STrue}$
by (*simp add*: *FusionRuleR strue-def*)

lemma *ST03*:
$(X \cap \mathit{SEmpty}) \cdot (X \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty})$
using *ST01* **by** *auto*

lemma *ST04*:
$(X \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty}) \cdot (X \cap \mathit{SEmpty})$
by (*metis B04 Int-commute FusionSEmptyL FusionSEmptyR inf.right-idem inf-top.right-neutral CH10*)

lemma *ST05*:
$(X \cap \mathit{SEmpty}) \subseteq -((-X) \cap \mathit{SEmpty})$
by *blast*

lemma *ST06*:
$((-X) \cap \mathit{SEmpty}) \subseteq -(X \cap \mathit{SEmpty})$
by *auto*

lemma *ST07*:
$(X \cap \mathit{SEmpty}) \cap (Y \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty}) \cdot \mathit{STrue}$
using *ST02 FusionSEmptyR* **by** *blast*

lemma *ST08*:
$(X \cap \mathit{SEmpty}) \cap (Y \cap \mathit{SEmpty}) \subseteq (\mathit{STrue} \cap \mathit{SEmpty}) \cdot (Y \cap \mathit{SEmpty})$
by (*metis FusionSEmptyL FusionSEmptyR ST33 inf.cobounded2*)

lemma *ST09*:
$((X \cap \mathit{SEmpty}) \cdot \mathit{STrue}) \cap (\mathit{STrue} \cap \mathit{SEmpty}) \cdot (Y \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty}) \cdot (Y \cap \mathit{SEmpty})$
by (*metis compl-bot-eq eq-refl FusionAssoc FusionSEmptyR inf.commute inf-top.left-neutral CH09 strue-def*)

lemma *ST10*:
$(X \cap \mathit{SEmpty}) \cdot (Y \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty})$
by (*metis FusionRuleR FusionSEmptyR inf-le2*)

lemma *ST11*:

$(X \cap \mathit{SEmpty}) \cdot (Y \cap \mathit{SEmpty}) \subseteq (Y \cap \mathit{SEmpty})$
using *ST01* **by** *blast*

lemma *ST12*:
$(X \cap \mathit{SEmpty}) \cap (Y \cap \mathit{SEmpty}) = (X \cap \mathit{SEmpty}) \cdot \mathit{SEmpty} \cap (Y \cap \mathit{SEmpty}) \cdot \mathit{SEmpty}$
by (*simp add*: *FusionSEmptyR*)

lemma *ST14*:
$((X \cap Y) \cap \mathit{SEmpty}) \cdot \mathit{SEmpty} = ((X \cap Y) \cap \mathit{SEmpty})$
by (*simp add*: *FusionSEmptyR*)

lemma *ST16*:
$(X \cap \mathit{SEmpty}) \cap (Y \cap \mathit{SEmpty}) \subseteq \mathit{SEmpty}$
by (*simp add*: *le-infI2*)

lemma *ST17*:
$(X \cap \mathit{SEmpty}) \cdot (Y \cap \mathit{SEmpty}) \subseteq \mathit{SEmpty}$
using *ST10* **by** *auto*

lemma *ST18*:
$-((X \cap \mathit{SEmpty}) \cup (Y \cap \mathit{SEmpty})) = -(X \cap \mathit{SEmpty}) \cap -(Y \cap \mathit{SEmpty})$
by *auto*

lemma *ST19*:
$(X \cap \mathit{SEmpty}) \cdot ((-X) \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty})$
using *ST10* **by** *blast*

lemma *ST20*:
$(X \cap \mathit{SEmpty}) \cdot ((-X) \cap \mathit{SEmpty}) \subseteq ((-X) \cap \mathit{SEmpty})$
using *ST01* **by** *auto*

lemma *ST22*:
$((X \cap \mathit{SEmpty}) \cdot \mathit{SSkip}) \cdot (Y \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty}) \cdot \mathit{SSkip}$
using *FusionRuleR FusionSEmptyR* **by** *blast*

lemma *ST23*:
$((X \cap \mathit{SEmpty}) \cdot \mathit{SSkip}) \cdot (Y \cap \mathit{SEmpty}) \subseteq \mathit{SSkip} \cdot (Y \cap \mathit{SEmpty})$
by (*simp add*: *ST01 FusionRuleL*)

lemma *ST27*:
$(\mathit{SInit}\ X) \cap (Y \cdot Z) \subseteq ((\mathit{SInit}\ X) \cap Y) \cdot Z$
by (*metis B04 compl-bot-eq FusionAssoc FusionSEmptyL inf-commute inf-top.left-neutral CH09 sinit-def strue-def*)

lemma *ST29*:
$(\mathit{SInit}\ X) \cdot Y \subseteq (\mathit{SInit}\ X)$
using *ST02 FusionAssoc sinit-def* **by** *fastforce*

lemma *ST30*:
$(\mathit{SInit}\ X) \cap (\mathit{SDi}\ Y) = (\mathit{SDi}\ (\ (\mathit{SInit}\ X) \cap Y))$

by (*metis FusionAssoc FusionSEmptyL CH09 compl-bot-eq inf-top.left-neutral*
sdi-def sinit-def strue-def)

lemma *ST31*:
$(X\cdot(\mathit{STrue} \cap \mathit{SEmpty})) \cap (\mathit{STrue}\cdot(Y \cap \mathit{SEmpty})) = X\cdot(Y \cap \mathit{SEmpty})$
by (*metis Int-commute compl-bot-eq inf-top.right-neutral CH10 strue-def*)

lemma *ST32*:
$(\mathit{STrue} \cap \mathit{SEmpty})\cdot\mathit{SEmpty} \cap (\mathit{SInit}\ X) = (X \cap \mathit{SEmpty})$
by (*metis Compl-empty-eq Int-commute CH09 ST14 inf-top.right-neutral*
sinit-def strue-def)

lemma *ST34*:
$((X \cap \mathit{SEmpty})\cdot Y) = (\mathit{SInit}\ X) \cap Y$
by (*metis FusionSEmptyL Int-commute CH09 compl-bot-eq inf-top-right sinit-def strue-def*)

lemma *ST35*:
$((\mathit{SInit}\ X) \cap Y)\cdot Z \subseteq (\mathit{SInit}\ X) \cap (Y\cdot Z)$
by (*metis B04 ST34 FusionAssoc*)

lemma *ST39*:
$\mathit{SEmpty} \cap (\mathit{SInit}\ X) \subseteq (X \cap \mathit{SEmpty})$
using *ST32* **by** *blast*

lemma *ST40*:
$(X \cap \mathit{SEmpty}) \subseteq \mathit{SEmpty} \cap (\mathit{SInit}\ X)$
using *ST32* **by** *auto*

lemma *ST41*:
$\mathit{SEmpty} \cap (\mathit{SInit}\ X) = (X \cap \mathit{SEmpty})$
using *ST40 ST39* **by** *auto*

lemma *ST42*:
$(X \cap \mathit{SEmpty}) \subseteq ((X \cup Y) \cap \mathit{SEmpty})$
by *blast*

lemma *ST43*:
$(Y \cap \mathit{SEmpty}) \subseteq ((X \cup Y) \cap \mathit{SEmpty})$
by *blast*

lemma *ST44*:
$(X \cap \mathit{SEmpty}) \cap ((-X) \cap \mathit{SEmpty}) = \mathit{SFalse}$
by (*simp add*: *sfalse-def*)

lemma *ST45*:
$((X \cup Y) \cap \mathit{SEmpty}) \subseteq (X \cap \mathit{SEmpty}) \cup (Y \cap \mathit{SEmpty})$
by *auto*

lemma *ST46*:
$(\mathit{SInit}\ X) \cup (\mathit{SInit}\ Y) = (\mathit{SInit}\ (X \cup Y))$

by (*simp add*: *Int-Un-distrib2 FusionUnionDistL sinit-def*)

lemma *ST48*:
$-(\mathit{STrue}\cdot(X \cap \mathit{SEmpty})) \subseteq \mathit{STrue}\cdot((-X) \cap \mathit{SEmpty})$
by (*metis B09 FusionSEmptyR FusionUnionDistR ST21 double-compl*)

lemma *ST49*:
$\mathit{STrue}\cdot((-X) \cap \mathit{SEmpty}) \subseteq -(\mathit{STrue}\cdot(X \cap \mathit{SEmpty}))$
by (*metis CH10 Compl-disjoint2 FusionSEmptyR FusionSFalse ST33 disjoint-eq-subset-Compl inf-compl-bot-left2 sfalse-def strue-def*)

lemma *ST50*:
$-(\mathit{STrue}\cdot(X \cap \mathit{SEmpty})) = \mathit{STrue}\cdot((-X) \cap \mathit{SEmpty})$
using *ST48 ST49* **by** *blast*

8.6.5 SStar

lemma *SStar02*:
assumes $X \subseteq Y$
shows $X\cdot(\mathit{SStar}\ Y) \cup \mathit{SEmpty} \subseteq (\mathit{SStar}\ Y)$
using *assms*
by (*smt FusionUnionDistL semigroup.assoc UnfoldL Subsumption sup.semigroup-axioms sup-left-idem UnionCommute*)

lemma *SStar04*:
$(\mathit{SStar}\ X) \subseteq (\mathit{SStar}\ X)\cdot(\mathit{SStar}\ X)$
by (*metis Un-absorb UnfoldL UnionAssoc ST47 sup.absorb-iff2*)

lemma *SStar09*:
assumes $(X\cdot(\mathit{SEmpty} \cup (X\cdot(\mathit{SStar}\ X)))) \cup \mathit{SEmpty} \subseteq (\mathit{SEmpty} \cup (X\cdot(\mathit{SStar}\ X)))$
shows $(\mathit{SStar}\ X) \subseteq \mathit{SEmpty} \cup (X\cdot(\mathit{SStar}\ X))$
using *assms*
by (*simp add*: *UnfoldL*)

lemma *SStar10*:
$(X\cdot(\mathit{SEmpty} \cup (X\cdot(\mathit{SStar}\ X)))) \subseteq (\mathit{SEmpty} \cup (X\cdot(\mathit{SStar}\ X)))$
by (*metis UnfoldL sup-ge2*)

lemma *SStar11*:
$\mathit{SEmpty} \subseteq (\mathit{SEmpty} \cup (X\cdot(\mathit{SStar}\ X)))$
by *auto*

lemma *SStar13*:
$(\mathit{SStar}\ \mathit{SSkip}) = \mathit{STrue}$
by (*simp add*: *SStarSkip*)

lemma *SStar14*:
$(\mathit{SSometime}\ X) = (\mathit{SStar}\ \mathit{SSkip})\cdot X$
by (*simp add*: *SStarSkip ssometime-def*)

lemma *SStar20*:
(*SStar SEmpty*) = *SEmpty*
by (*metis FusionSEmptyR ST15 ST33*)

lemma *SStar21*:
(*SStar* (*SEmpty* ∩ *X*))·(*SEmpty* ∩ *X*) = (*SEmpty* ∩ *X*)
by (*metis ST15 FusionSEmptyL inf-commute*)

lemma *SStar24*:
(*SStar SFalse*) = *SEmpty*
by (*metis SStar20 SStar47 inf-compl-bot sfalse-def smore-def*)

lemma *SStar26*:
X ⊆ (*SStar X*)
by (*smt FusionUnionDistR FusionSEmptyR SStar03 SStar04 Subsumption UnfoldL UnionAssoc UnionCommute*)

lemma *SStar27*:
SEmpty ⊆ (*SStar X*)
using *UnfoldL* **by** *blast*

lemma *SStar31*:
assumes *X* ∪ (*X*·*Y*)·(*X*·(*SStar* (*Y*·*X*))) ⊆ *X*·(*SStar* (*Y*·*X*))
shows (*SStar* (*X*·*Y*))·*X* ⊆ *X*·(*SStar* (*Y*·*X*))
using *assms SStarInductL* **by** *blast*

lemma *SStar32*:
X ∪ (*X*·*Y*)·(*X*·(*SStar* (*Y*·*X*))) ⊆ *X*·(*SStar* (*Y*·*X*))
by (*metis B06 SStar10 SStar11 FusionAssoc FusionRuleR FusionSEmptyR UnfoldL*)

lemma *SStar33*:
(*SStar* (*X*·*Y*))·*X* ⊆ *X*·(*SStar* (*Y*·*X*))
using *SStar31 SStar32* **by** *blast*

lemma *SStar37*:
assumes *X*·*Z* ⊆ *Z*·*Y*
shows (*SStar X*)·*Z* ⊆ *Z*·(*SStar Y*)
by (*smt Un-commute assms FusionAssoc FusionUnionDistL FusionUnionDistR FusionSEmptyR semigroup.assoc UnfoldL SStarInductL Subsumption sup.right-idem sup.semigroup-axioms*)

lemma *SStar38*:
assumes *Z*·*X* ⊆ *Y*·*Z*
shows *Z*·(*SStar X*) ⊆ (*SStar Y*)·*Z*
by (*smt SStar30 assms FusionAssoc FusionUnionDistL FusionUnionDistR FusionSEmptyL semigroup.assoc UnfoldL SStarInductR Subsumption sup.right-idem sup.semigroup-axioms UnionCommute*)

lemma *SStar39*:
Y·(*SStar* ((*SStar X*)·*Y*)) ⊆ (*SStar* (*Y*·(*SStar X*)))·*Y*
by (*simp add*: *SStar38 FusionAssoc*)

lemma *SStar40*:
$(\mathit{SStar}\ (Y \cdot (\mathit{SStar}\ X))) \cdot Y \subseteq Y \cdot (\mathit{SStar}\ ((\mathit{SStar}\ X) \cdot Y))$
by (*simp add*: *SStar33*)

lemma *SStar41*:
$Y \cdot (\mathit{SStar}\ ((\mathit{SStar}\ X) \cdot Y)) = (\mathit{SStar}\ (Y \cdot (\mathit{SStar}\ X))) \cdot Y$
using *SStar39 SStar40* **by** *blast*

lemma *SStar42*:
$Z \cdot (\mathit{SStar}\ (Y \cdot Z)) \subseteq (\mathit{SStar}\ (Z \cdot Y)) \cdot Z$
by (*simp add*: *SStar38 FusionAssoc*)

lemma *SStar43*:
$(\mathit{SStar}\ (Z \cdot Y)) \cdot Z \subseteq Z \cdot (\mathit{SStar}\ (Y \cdot Z))$
by (*simp add*: *SStar33*)

lemma *SStar44*:
$Z \cdot (\mathit{SStar}\ (Y \cdot Z)) = (\mathit{SStar}\ (Z \cdot Y)) \cdot Z$
using *SStar42 SStar43* **by** *blast*

lemma *SStar49*:
$(\mathit{SStar}\ X) = \mathit{SEmpty} \cup (\mathit{SStar}\ X) \cdot X$
using *SStar30 UnfoldL* **by** *blast*

8.6.6 Box and Diamond

lemma *BD01*:
$(\mathit{SSometime}\ \mathit{SEmpty}) = \mathit{STrue}$
by (*simp add*: *ssometime-def FusionSEmptyR*)

lemma *BD02*:
$X \subseteq (\mathit{SSometime}\ X)$
by (*metis FusionUnionDistL SEmptyChop STrueTop Subsumption Un-absorb semigroup.assoc ssometime-def sup.semigroup-axioms*)

lemma *BD03*:
$(\mathit{SNext}\ (\mathit{SSometime}\ X)) \subseteq (\mathit{SSometime}\ X)$
by (*metis FusionUnionDistL N03 SStar16 SStar03 SStar04 SStarSkip snext-def ssometime-def sup.absorb-iff2 sup.orderE*)

lemma *BD04*:
$(\mathit{SSometime}\ (\mathit{SNext}\ X)) \subseteq (\mathit{SSometime}\ X)$
by (*metis CH01 FusionAssoc FusionUnionDistL FusionUnionDistR SStar16 SStarSkip snext-def ssometime-def sup.absorb-iff2*)

lemma *BD05*:
$(\mathit{SSometime}\ X) \cup (\mathit{SSometime}\ Y) = (\mathit{SSometime}\ (X \cup Y))$
by (*simp add*: *FusionUnionDistR ssometime-def*)

lemma *BD06*:

```
  (SSometime STrue) = STrue
by (simp add: CH01 ssometime-def)

lemma BD07:
  (SSometime (X ∩ Y)) ⊆ (SSometime X) ∩ (SSometime Y)
by (simp add: FusionRuleR ssometime-def)

lemma BD08:
  (SAlways STrue) = STrue
by (simp add: SBoxGen)

lemma BD09:
  −(SAlways X) = (SSometime (−X))
by (simp add: salways-def)

lemma BD10:
  (SAlways X) ⊆ (SSometime X)
by (metis B02 BD02 BD09 set-rev-mp subsetI)

lemma BD11:
  (SSometime (SSometime X)) = (SSometime X)
by (simp add: CH01 ssometime-def FusionAssoc)

lemma BD12:
  (SAlways X) ⊆ X
by (simp add: B02 BD02 BD09)

lemma BD13:
  (SDi STrue) = STrue
by (simp add: CH01 sdi-def)

lemma BD14:
  (SDi SEmpty) = STrue
by (simp add: sdi-def FusionSEmptyL)

lemma BD15:
  (SBi STrue) = STrue
by (simp add: SBiGen)

lemma BD16:
  (SDi (X ∪ Y)) = (SDi X) ∪ (SDi Y)
by (simp add: FusionUnionDistL sdi-def)

lemma BD17:
 assumes  X ⊆ Y
 shows   (SDi X) ⊆ (SDi Y)
using assms
by (metis FusionUnionDistL Subsumption sdi-def)

lemma BD18:
  (SDi (SDi X)) = (SDi X)
by (metis CH01 FusionAssoc sdi-def)

lemma BD19:
  (SDa SEmpty) = STrue
by (simp add: CH01 sda-def FusionSEmptyR)

lemma BD20:
  (SDa STrue) = STrue
by (simp add: CH01 sda-def)

lemma BD21:
  (SBa STrue) = STrue
by (metis BD15 BD08 BD09 sba-def sbi-def sda-def sdi-def ssometime-def)

lemma BD22:
  (SDa (X ∪ Y)) = (SDa X) ∪ (SDa Y)
by (simp add: FusionUnionDistL FusionUnionDistR sda-def)

lemma BD23:
 assumes  X ⊆ Y
 shows    (SDa X) ⊆ (SDa Y)
using assms
by (metis BD22 Subsumption)

lemma BD24:
 assumes  X ⊆ Y
 shows    (SDa (−Y)) ⊆ (SDa (−X))
using assms
by (simp add: BD23)

lemma BD25:
  (SDi X) ⊆ (SDa X)
by (metis BD02 FusionAssoc sda-def sdi-def ssometime-def)

lemma BD26:
  (SSometime X) ⊆ (SDa X)
by (metis BD01 BD02 FusionSEmptyR FusionUnionDistR SStar14 le-iff-sup sda-def)

lemma BD27:
  (SBa X) ⊆ (SBi X)
by (simp add: BD25 sba-def sbi-def)

lemma BD28:
  (SBa X) ⊆ (SAlways X)
by (simp add: B02 BD26 BD09 sba-def)

lemma BD29:
  (SAlways X) ∩ (SAlways Y) = (SAlways (X ∩ Y))
by (metis BD05 BD09 Morgan compl-inf salways-def)
```

lemma *BD30*:
(*SAlways X*) ∪ (*SAlways Y*) ⊆ (*SAlways* (*X* ∪ *Y*))
using *BD07*
by (*metis B02 BD09 compl-sup*)

lemma *BD31*:
(*SDi* (*X* ∩ *Y*)) ⊆ (*SDi X*) ∩ (*SDi Y*)
by (*simp add*: *BD17*)

lemma *BD32*:
(*SBi X*) ∪ (*SBi Y*) ⊆ (*SBi* (*X* ∪ *Y*))
using *BD31*
by (*metis* (*mono-tags*, *lifting*) *B02 compl-sup double-compl sbi-def*)

lemma *BD33*:
(*SDa* (*X* ∩ *Y*)) ⊆ (*SDa X*) ∩ (*SDa Y*)
by (*simp add*: *BD23*)

lemma *BD34*:
(*SBa X*) ∪ (*SBa Y*) ⊆ (*SBa* (*X* ∪ *Y*))
using *BD33*
by (*metis* (*mono-tags*, *lifting*) *B02 compl-sup double-compl sba-def*)

lemma *BD35*:
(*SAlways SEmpty*) = *SEmpty*
by (*metis N13 SStar14 SStar30 SStar48 SStarSkip double-complement salways-def smore-def*)

lemma *BD36*:
(*SBi SEmpty*) = *SEmpty*
using *N13 sbi-def sdi-def smore-def* **by** *fastforce*

lemma *BD37*:
(*SBa SEmpty*) = *SEmpty*
by (*metis N13 SStar30 SStar48 double-complement sba-def sda-def smore-def*)

lemma *BD38*:
assumes *X* ⊆ *Y*
shows (*SAlways X*) ⊆ (*SAlways Y*)
using *assms*
by (*simp add*: *BD29 inf.absorb-iff2*)

lemma *BD39*:
assumes *X* ⊆ *Y*
shows (*SBi X*) ⊆ (*SBi Y*)
using *assms*
by (*simp add*: *BD17 sbi-def*)

lemma *BD40*:
assumes *X* ⊆ *Y*

shows $(\mathit{SBa}\ X) \subseteq (\mathit{SBa}\ Y)$
using *assms*
by (*simp add*: *BD24 sba-def*)

lemma *BD41*:
$(\mathit{SBi}\ (\mathit{SBi}\ X)) = (\mathit{SBi}\ X)$
by (*simp add*: *BD18 sbi-def*)

lemma *BD42*:
$(\mathit{SAlways}\ (\mathit{SAlways}\ X)) = (\mathit{SAlways}\ X)$
by (*simp add*: *BD11 salways-def*)

lemma *BD43*:
$(\mathit{SDa}\ (\mathit{SDa}\ X)) = (\mathit{SDa}\ X)$
by (*metis CH01 FusionAssoc sda-def*)

lemma *BD44*:
$(\mathit{SBa}\ (\mathit{SBa}\ X)) = (\mathit{SBa}\ X)$
by (*simp add*: *BD43 sba-def*)

lemma *BD47*:
$(\mathit{SAlways}\ (\ (-X) \cup Y)) \subseteq (\ -(\mathit{SAlways}\ X) \cup (\mathit{SAlways}\ Y))$
by (*metis B20 BD12 BD29 BD38 BD42 double-compl*)

lemma *BD48*:
$(\mathit{SAlways}\ X) \subseteq X \cap (\mathit{SWnext}\ (\mathit{SAlways}\ X))$
by (*metis B02 B16 BD03 BD09 BD12 N12 salways-def*)

lemma *BD49*:
$(\mathit{SBi}\ (\ (-X) \cup Y)) \subseteq (-(\mathit{SBi}\ X) \cup (\mathit{SBi}\ Y)\ )$
by (*smt B20 FusionUnionDistL Huntington Subsumption compl-inf double-compl inf-commute sbi-def sdi-def sup-ge2 sup-left-commute*)

lemma *BD50*:
$(\mathit{SPrev}\ (\mathit{SDi}\ X)) \subseteq (\mathit{SDi}\ X)$
by (*metis BD13 FusionAssoc FusionUnionDistR SStar16 SStarSkip Subsumption sdi-def sprev-def*)

lemma *BD51*:
$-(\mathit{SBi}\ X) = (\mathit{SDi}\ (-X))$
by (*simp add*: *sbi-def*)

lemma *BD52*:
$X \subseteq (\mathit{SDi}\ X)$
by (*metis FusionSEmptyR FusionUnionDistR ST33 Subsumption UnionCommute sdi-def sup-inf-absorb*)

lemma *BD53*:
$(\mathit{SBi}\ X) \subseteq X$
by (*simp add*: *B02 BD51 BD52*)

lemma *BD54*:

(*SBi X*) ⊆ *X* ∩ (*SWprev* (*SBi X*))
by (*metis B02 B16 BD50 BD51 BD53 N29 sbi-def*)

lemma *BD55*:
(*SBi* (*SMore* ∪ *X*)) = (*SInit X*)
by (*smt FusionSEmptyR Morgan ST33 ST38 UnionCommute compl-inf compl-sup sbi-def sdi-def sinit-def smore-def*)

lemma *BD56*:
(*SAlways* (*SMore* ∪ *X*)) = *STrue*·(*X* ∩ *SEmpty*)
by (*simp add*: *SStar14 SStarSkip ST50 UnionCommute salways-def smore-def*)

8.7 Time Reversal

8.7.1 Time Reversal Axioms

lemma *SRevSEmpty*:
(*SRev SEmpty*) = *SEmpty*
using *set-eqI*[*of* (*SRev SEmpty*) *SEmpty*]
by (*simp add*: *sempty-elim srev-elim*)

lemma *SRevSNot*:
(*SRev* (− *X*)) = (− (*SRev X*))
using *set-eqI*[*of* (*SRev* (− *X*)) (− (*SRev X*))]
by (*simp add*: *srev-elim*)

lemma *SRevFusion*:
(*SRev* (*X*·*Y*)) = (*SRev Y*)·(*SRev X*)
using *set-eqI*[*of* (*SRev* (*X*·*Y*)) (*SRev Y*)·(*SRev X*)]
using *fusion-iff-1*
by (*smt interval-intrev-intlen interval-intrev-prefix interval-intrev-suffix interval-prefix-length interval-suffix-length-good order-refl srev-elim*)

lemma *SRevUnion*:
(*SRev* (*X* ∪ *Y*)) = (*SRev X*) ∪ (*SRev Y*)
using *set-eqI*[*of* (*SRev* (*X* ∪ *Y*)) (*SRev X*) ∪ (*SRev Y*)]
using *srev-elim* **by** *auto*

lemma *SRevSPower*:
(*SRev* (*SPower X n*)) = (*SPower* (*SRev X*) *n*)
by (*induct n*, *simp add*: *SRevSEmpty*, *smt Morgan SRevFusion SRevSEmpty SRevSNot SRevUnion pwr-Suc smore-def spower-commutes*)

lemma *SRevSStar*:
(*SRev* (*SStar X*)) = (*SStar* (*SRev X*))
proof −
have *1*: (*SRev* (*SStar X*)) = (*SRev* (⋃ *n*. *SPower X n*)) **by** (*simp add*: *sstar-def*)
have *2*: (*SRev* (⋃ *n*. *SPower X n*)) = (⋃ *n*. *SPower* (*SRev X*) *n*)
using *set-eqI*[*of* (*SRev* (⋃ *n*. *SPower X n*)) (⋃ *n*. *SPower* (*SRev X*) *n*)]
by (*metis* (*mono-tags*, *lifting*) *SRevSPower UN-iff srev-elim*)
have *3*: (⋃ *n*. *SPower* (*SRev X*) *n*) = (*SStar* (*SRev X*)) **by** (*simp add*: *sstar-def*)

from *1 2 3* **show** *?thesis* **by** *auto*
qed

lemma *SRevSRev*:
(*SRev* (*SRev X*)) = *X*
using *set-eqI*[*of* (*SRev* (*SRev X*)) *X*]
by (*simp add*: *srev-elim*)

8.7.2 Time Reversal Laws

lemma *TR01*:
(*SRev SMore*) = *SMore*
by (*simp add*: *SRevSEmpty SRevSNot smore-def*)

lemma *TR02*:
(*SRev SSkip*) = *SSkip*
by (*metis SRevFusion SRevSEmpty SRevSNot SRevUnion TR01 sskip-def*)

lemma *TR03*:
(*SRev STrue*) = *STrue*
by (*metis SRevSStar SStarSkip TR02*)

lemma *TR04*:
(*SRev SFalse*) = *SFalse*
by (*metis Compl-eq-Compl-iff SRevSNot TR03 sfalse-def strue-def*)

lemma *TR05*:
(*SRev* (*SSometime X*)) = (*SDi* (*SRev X*))
by (*simp add*: *SRevFusion TR03 sdi-def ssometime-def*)

lemma *TR06*:
(*SRev* (*SAlways X*)) = (*SBi* (*SRev X*))
by (*simp add*: *SRevSNot TR05 salways-def sbi-def*)

lemma *TR07*:
(*SRev* (*SDi X*)) = (*SSometime* (*SRev X*))
by (*simp add*: *SRevFusion TR03 sdi-def ssometime-def*)

lemma *TR08*:
(*SRev* (*SBi X*)) = (*SAlways* (*SRev X*))
by (*metis SRevSRev TR06*)

lemma *TR09*:
(*SRev* (*SNext X*)) = (*SPrev* (*SRev X*))
by (*simp add*: *SRevFusion TR02 snext-def sprev-def*)

lemma *TR10*:
(*SRev* (*SWnext X*)) = (*SWprev* (*SRev X*))
by (*simp add*: *SRevFusion SRevSNot TR02 swnext-def swprev-def*)

lemma *TR11*:
$(\mathit{SRev}\ (\mathit{SPrev}\ X)) = (\mathit{SNext}\ (\mathit{SRev}\ X))$
by (*simp add*: *SRevFusion TR02 snext-def sprev-def*)

lemma *TR12*:
$(\mathit{SRev}\ (\mathit{SWprev}\ X)) = (\mathit{SWnext}\ (\mathit{SRev}\ X))$
by (*metis SRevSRev TR10*)

lemma *TR13*:
$(\mathit{SRev}\ (\mathit{SDa}\ X)) = (\mathit{SDa}\ (\mathit{SRev}\ X))$
by (*simp add*: *SRevFusion TR03 sda-def FusionAssoc*)

lemma *TR14*:
$(\mathit{SRev}\ (\mathit{SBa}\ X)) = (\mathit{SBa}\ (\mathit{SRev}\ X))$
by (*simp add*: *SRevSNot TR13 sba-def*)

lemma *TR15*:
$(\mathit{SRev}\ (\mathit{SPower}\ \mathit{SSkip}\ n)) = (\mathit{SPower}\ \mathit{SSkip}\ n)$
by (*simp add*: *SRevSPower TR02*)

lemma *TR16*:
assumes $X \subseteq Y$
shows $(\mathit{SRev}\ X) \subseteq (\mathit{SRev}\ Y)$
using *assms* **by** (*metis SRevUnion le-iff-sup*)

lemma *TR17*:
assumes $X = Y$
shows $(\mathit{SRev}\ X) = (\mathit{SRev}\ Y)$
using *assms TR16* **by** *auto*

8.8 Link between Set of Intervals and ITL

lemma *interval-lan* [*simp*]:
$\sigma \in (\mathit{lan}\ f) \longleftrightarrow (\sigma \models f)$
by (*simp add*: *lan-def*)

lemma *valid-lan-eqv* :
$(\ (\mathit{lan}\ f) = (\mathit{lan}\ g)\ ) \longleftrightarrow (\ \vdash f \equiv_i g\ )$
using *interval-lan lan-def* **by** *force*

lemma *valid-lan-imp* :
$(\ (\mathit{lan}\ f) \subseteq (\mathit{lan}\ g)\ ) \longleftrightarrow (\vdash f \supset_i g)$
by (*meson implies-defs interval-lan subset-eq valid-def*)

lemma *valid-strue* :
$(\ (\mathit{lan}\ f) = \mathit{STrue}) \longleftrightarrow (\vdash f)$
using *strue-def* **by** *fastforce*

lemma *strue-true*:
$\sigma \in \mathit{STrue} \longleftrightarrow (\sigma \models \mathit{true}_i)$

by (*simp add*: *strue-elim*)

lemma *strue-true-1*:
$STrue = (lan\ true_i)$
using *lan-def strue-true* **by** *fastforce*

lemma *sfalse-false*:
$\sigma \in SFalse \longleftrightarrow (\sigma \models false_i)$
by (*simp add*: *sfalse-def*)

lemma *sfalse-false-1*:
$SFalse = (lan\ false_i)$
using *sfalse-false* **using** *lan-def* **by** *fastforce*

lemma *not-negation*:
$\sigma \in (-(lan\ f)) \longleftrightarrow (\sigma \models \neg_i\ f)$
by *simp*

lemma *not-negation-1*:
$-(lan\ f\ ) = (lan\ (\neg_i\ f))$
using *interval-lan lan-def* **by** *fastforce*

lemma *inter-and*:
$(\sigma \in (\ (lan\ f) \cap (lan\ g)\ )) \longleftrightarrow (\sigma \models f \wedge_i g)$
by (*simp add*: *lan-def*)

lemma *inter-and-1*:
$(\ (lan\ f) \cap (lan\ g)\ ) = (lan\ (f \wedge_i g))$
using *inter-and lan-def* **by** *fastforce*

lemma *union-or*:
$(\sigma \in (\ (lan\ f) \cup (lan\ g)\ )) \longleftrightarrow (\sigma \models f \vee_i g)$
by (*simp add*: *lan-def*)

lemma *union-or-1*:
$(\ (lan\ f) \cup (lan\ g)\ ) = (lan\ (f \vee_i g))$
using *union-or lan-def* **by** *fastforce*

lemma *subset-impl*:
$(\sigma \in (\ -(lan\ f) \cup (lan\ g))) \longleftrightarrow (\sigma \models f \supset_i g)$
by *simp*

lemma *subset-impl-1*:
$(\ -(lan\ f) \cup (lan\ g)) = (lan\ (f \supset_i g))$
using *subset-impl lan-def* **by** *fastforce*

lemma *fusion-chop*:
$(\sigma \in ((lan\ f)\cdot(lan\ g))) \longleftrightarrow (\sigma \models f;g)$
by (*simp add*: *chop-fuse fusion-iff*)

lemma *fusion-chop-1*:
$((\mathit{lan}\ f)\cdot(\mathit{lan}\ g)) = (\mathit{lan}\ (f;g))$
using *fusion-chop lan-def* **by** *blast*

lemma *sempty-empty*:
$\sigma \in \mathit{SEmpty} \longleftrightarrow (\sigma \models \mathit{empty})$
by (*simp add*: *sempty-elim*)

lemma *sempty-empty-1*:
$\mathit{SEmpty} = (\mathit{lan}\ \mathit{empty})$
using *sempty-empty lan-def* **by** *fastforce*

lemma *smore-more*:
$\sigma \in \mathit{SMore} \longleftrightarrow (\sigma \models \mathit{more})$
by (*simp add*: *smore-elim*)

lemma *smore-more-1*:
$\mathit{SMore} = (\mathit{lan}\ \mathit{more})$
using *smore-more lan-def* **by** *fastforce*

lemma *sskip-skip*:
$\sigma \in \mathit{SSkip} = (\sigma \models \mathit{skip})$
by (*simp add*: *sskip-elim*)

lemma *sskip-skip-1*:
$\mathit{SSkip} = (\mathit{lan}\ \mathit{skip})$
using *sskip-skip lan-def* **by** *fastforce*

lemma *snext-next*:
$\sigma \in (\mathit{SNext}\ (\mathit{lan}\ f)) \longleftrightarrow (\sigma \models \bigcirc\ f)$
by (*metis snext-def fusion-chop next-d-def sskip-skip-1*)

lemma *snext-next-1*:
$(\mathit{SNext}\ (\mathit{lan}\ f)) = (\mathit{lan}\ (\bigcirc\ f))$
using *snext-next lan-def* **by** *fastforce*

lemma *swnext-wnext*:
$\sigma \in (\mathit{SWnext}\ (\mathit{lan}\ f)) \longleftrightarrow (\sigma \models \mathit{wnext}\ f)$
by (*simp add*: *swnext-def fusion-chop-1 next-d-def not-negation-1 sskip-skip-1 wnext-d-def*)

lemma *swnext-wnext-1*:
$(\mathit{SWnext}\ (\mathit{lan}\ f)) = (\mathit{lan}\ (\mathit{wnext}\ f))$
using *swnext-wnext lan-def* **by** *fastforce*

lemma *sprev-prev*:
$\sigma \in (\mathit{SPrev}\ (\mathit{lan}\ f)) \longleftrightarrow (\sigma \models \mathit{prev}\ f)$
by (*metis fusion-chop prev-d-def sprev-def sskip-skip-1*)

lemma *sprev-prev-1*:
$(\mathit{SPrev}\ (\mathit{lan}\ f)) = (\mathit{lan}\ (\mathit{prev}\ f))$

using *sprev-prev lan-def* **by** *fastforce*

lemma *swprev-wprev*:
$\sigma \in (\mathit{SWprev}\ (\mathit{lan}\ f)) \longleftrightarrow (\sigma \models \mathit{wprev}\ f)$
by (*simp add*: *fusion-chop-1 not-negation-1 prev-d-def sskip-skip-1 swprev-def wprev-d-def*)

lemma *swprev-wprev-1*:
$(\mathit{SWprev}\ (\mathit{lan}\ f)) = (\mathit{lan}\ (\mathit{wprev}\ f))$
using *swprev-wprev lan-def* **by** *fastforce*

lemma *sinit-init*:
$\sigma \in \mathit{SInit}\ (\mathit{lan}\ f) \longleftrightarrow (\sigma \models \mathit{init}\ f)$
by (*simp add*: *Int-commute fusion-chop-1 init-d-def inter-and-1 sempty-empty-1 sinit-def strue-true-1*)

lemma *sinit-init-1*:
$\mathit{SInit}\ (\mathit{lan}\ f) = (\mathit{lan}\ (\mathit{init}\ f))$
using *sinit-init lan-def* **by** *fastforce*

lemma *and-inter-more*:
$\sigma \in (((\mathit{lan}\ f) \cap \mathit{SMore})) \longleftrightarrow (\sigma \models (f\ \wedge_i\ \mathit{more})\ )$
using *smore-more inter-and* **by** *auto*

lemma *and-inter-more-1*:
$\sigma \in (((\mathit{lan}\ f) \cap \mathit{SMore})) \longleftrightarrow (\sigma \in (\mathit{lan}\ (f\ \wedge_i\ \mathit{more})\ ))$
using *and-inter-more lan-def* **by** *fastforce*

lemma *and-inter-more-2*:
$((\mathit{lan}\ f) \cap \mathit{SMore}) = \ (\mathit{lan}\ (f\ \wedge_i\ \mathit{more})\ )$
using *and-inter-more-1* **by** *blast*

lemma *and-chop*:
$\sigma \in (((\mathit{lan}\ f) \cap \mathit{SMore})\cdot\ (\mathit{lan}\ g)) \longleftrightarrow (\sigma \models (f\ \wedge_i\ \mathit{more});g)$
by (*metis fusion-chop inter-and-1 smore-more-1*)

lemma *and-chop-1*:
$(((\mathit{lan}\ f) \cap \mathit{SMore})\cdot\ (\mathit{lan}\ g)) = (\mathit{lan}\ ((f\ \wedge_i\ \mathit{more});g))$
using *and-chop lan-def* **by** *blast*

lemma *spower-chop-power*:
$(\mathit{SPower}\ (\mathit{lan}\ f)\ n) = (\mathit{lan}\ (\mathit{power\text{-}chop\text{-}d}\ f\ n))$
by (*induct n*, *simp add*: *sempty-empty-1*, *simp add*: *and-chop-1*)

lemma *sstar-spower*:
$\sigma \in \mathit{SStar}\ (\mathit{lan}\ f) \longleftrightarrow (\exists\ n.\ \sigma \in \mathit{SPower}\ (\mathit{lan}\ f)\ n)$
by (*simp add*: *sstar-def*)

lemma *sstar-chopstar*:
$\sigma \in (\mathit{SStar}\ (\mathit{lan}\ f)) \longleftrightarrow \sigma \in (\mathit{lan}\ (f^{\star}))$
using *chopstar-eqv-power-chop sstar-spower interval-lan spower-chop-power* **by** *blast*

lemma *chopstar-sstar-1*:
$(SStar\ (lan\ f)) = (lan\ (f^{\star}))$
using *sstar-chopstar lan-def* **by** *blast*

lemma *chopstar-seqv*:
$\sigma \in (lan\ (f^{\star})) \longleftrightarrow \sigma \in (lan\ (empty\ \vee_i\ \ (f\ \wedge_i\ more);\ f^{\star}))$
using *ChopstarEqv valid-lan-eqv* **by** *blast*

lemma *chopstar-seqv-1*:
$(lan\ (f^{\star})) = (lan\ (empty\ \vee_i\ \ (f\ \wedge_i\ more);\ f^{\star}))$
using *chopstar-seqv lan-def* **by** *blast*

lemma *srev-rev*:
$\sigma \in (SRev\ (lan\ f)) \longleftrightarrow \sigma \in (lan\ (f^r))$
by (*metis TimeReverseSem interval-lan interval-rev-rev-ident srev-elim*)

lemma *srev-rev-1*:
$(SRev\ (lan\ f)) = (lan\ (f^r))$
using *srev-rev lan-def* **by** *blast*

end

References

[1] A. Armstrong, G. Struth, and T. Weber. Kleene algebra. *Archive of Formal Proofs*, 2013. http://isa-afp.org/entries/Kleene_Algebra.shtml, Formal proof development.

[2] A. Cau. Interval temporal algebra, 2009. http://antonio-cau.co.uk/ITL/itl-atp/index.html.

[3] B. Moszkowski. A Hierarchical Completeness Proof for Propositional Interval Temporal Logic with Finite Time. *Journal of Applied Non-Classical Logics*, 14(1–2):55–104, 2004.

[4] B. Moszkowski. Compositional reasoning using intervals and time reversal. *Annals of Mathematics and Artificial Intelligence*, 71(1-3):175–250, 2014.

[5] B. C. Moszkowski. Imperative reasoning in interval temporal logic. Technical report, University of Newcastle upon Tyne, 1996.

[6] D. Smallwood. *ITL Monitor: Compositional Runtime Analysis with Interval Temporal Logic*. PhD thesis, De Montfort University, 2018.