Interval Temporal Logic
A not so short introduction

Antonio Cau

Website ITL course
Slides ITL course

Overview (2)

This course will give a not so short introduction to Interval Temporal Logic (ITL)

ITL is a

- discrete, linear temporal logic
- for both finite and infinite intervals (sequences of states) which includes
- a basic construct for sequential composition and
- an analog of Kleene star (regular expressions)

Features of ITL (3)

- Combines temporal logic, automata and regular expressions
- Modular reasoning about time (e.g., hardware, software, multimedia)
- Flexible notation for discrete linear order
- Analytical framework for going from infinite-time behaviour to finite-time behaviour.
- Supports sequential operators found in programs, etc.

Features of ITL (4)

- Compositionality with assumptions and commitments
- Refinement to derive concrete programs from abstract specifications.
- ITL with memory and framing can embed various kinds of imperative programs, including parallel ones.
- Executable specifications and imperative subsets.
- Temporal projection between different time granularities; enables imperative constructs for interleaved processes.

ITL's Influence (5)

- ITL helped influence Prof Mike Gordon, FRS to switch from LCF to HOL in his theorem proving tools. (E.g., see his article *From LCF to HOL: A short history*).
- Hybrid systems: Duration Calculus (real and discrete time).
- ITL is used to give semantics to Verilog (EPSRC project with Oxford University).
- ITL is used in hardware/software codesign (EPSRC projects with University of Newcastle upon Tyne and Oxford University).
- ITL is used for security and trust policies (DTC-DIF projects).
- ITL used in the projects Safemos (UK) and ProCoS (EU).
- Mexitl (Kent, Winnipeg) is based on ITL, Multimedia in Executable Interval Temporal Logic.

ITL's Influence (6)

- AnaTempura runtime verification tool uses executable subset of ITL.
- Influenced Verisity Ltd.'s language temporal e (part of IEEE Standard 1647). Verisity acquired by Cadence Design Systems, Inc., a major supplier of electronic design tools and services.
- Used in the KIV theorem prover (University of Augsburg, Germany) - e.g., for Statecharts, UML and medical protocols.
- ITL with framing and temporal projection extensively studied by Duan and group (Xidian University, Xi'an, China).

Part I: Propositional Logic (7)

Part II: Propositional ITL (8)

Part III: First Order Logic (9)

Syntax First Order Logic

Examples First Order Logic

State First Order Logic

Semantics of Expressions

Semantics of Formulae

Exercises First Order Logic

Part IV: First Order ITL (10)

Syntax First Order ITL

Examples First Order ITL

Informal Semantics First Order ITL

Semantics of Expressions

Semantics of Formulae

Derived First Order ITL formulae

Exercises First Order ITL

(11)

Part I

Propositional Logic

Outline (12)

Syntax Propositional Logic

Examples Propositional Logic

State Propositional Logic

Semantic Boolean Operators

Semantics of Propositional Logic

Exercises Propositional Logic

Propositional Logic (13)

Syntax of Propositional Logic:

- Boolean values: true, false
- Boolean variables: $p, q, \ldots$
- Boolean operators:

  $\wedge$ (and),
  $\vee$ (or),
  $\neg$ (not),
  $\supset$ (implication),
  $\equiv$ (equivalence)

Propositional Logic (14)

Syntax of propositional formulae in BNF:

$$f ::= \text{true} \mid p \mid \neg f \mid f_1 \wedge f_2$$

Derived Boolean operators:

$$\begin{array}{lcl} \text{false} & \widehat{=} & \neg\text{true} \\ f_1 \vee f_2 & \widehat{=} & \neg(\neg f_1 \wedge \neg f_2) \\ f_1 \supset f_2 & \widehat{=} & \neg f_1 \vee f_2 \\ f_1 \equiv f_2 & \widehat{=} & (f_1 \supset f_2) \wedge (f_2 \supset f_1) \end{array}$$

Propositional Logic Examples (15)

Example

true
$\neg p$
$\text{true} \wedge p$
$\text{false} \vee \neg p$
$\text{false} \supset p$
$p \equiv (p \wedge r)$

State (16)

A state is a mapping ***State*** from the set of propositional variables Var^b to the set of Boolean values $Bool \mathrel{\widehat{=}} \{\text{tt}, \text{ff}\}$.
tt is the semantic 'true' value and ff the semantic 'false' value.

$$State : Var^b \to Bool$$

We will use $\sigma_0, \sigma_1, \sigma_2, \ldots$ to denote states and Σ to denote the set of all possible states.

Example

Let σ_0 be a state such that

$$\sigma_0(p) = \text{tt}$$
$$\sigma_0(q) = \text{ff}$$

Semantic Boolean operators (17)

Truth tables for Semantic Boolean operators:

- not :

X	not X
tt	ff
ff	tt

- and :

X	Y	X and Y
tt	tt	tt
tt	ff	ff
ff	tt	ff
ff	ff	ff

or :

X	Y	X or Y
tt	tt	tt
tt	ff	tt
ff	tt	tt
ff	ff	ff

- if :

X	Y	X implies Y
tt	tt	tt
tt	ff	ff
ff	tt	tt
ff	ff	tt

iff :

X	Y	X iff Y
tt	tt	tt
tt	ff	ff
ff	tt	ff
ff	ff	tt

Semantics (18)

Let $[\![\ldots]\!]$ be the "meaning" function from ***Propositions*** $\times$ ***State*** to $\{\text{tt}, \text{ff}\}$ and let σ_0 be a state then

$$
\begin{array}{lcl}
[\![\text{true}]\!]_{\sigma_0} & \widehat{=} & \text{tt} \\
[\![p]\!]_{\sigma_0} & \widehat{=} & \sigma_0(p) \\
[\![\neg f]\!]_{\sigma_0} & \widehat{=} & \text{not } ([\![f]\!]_{\sigma_0}) \\
[\![f_1 \wedge f_2]\!]_{\sigma_0} & \widehat{=} & ([\![f_1]\!]_{\sigma_0} \text{ and } [\![f_2]\!]_{\sigma_0})
\end{array}
$$

Semantics (19)

Example

Let $\sigma_0(p) = \text{tt}$ and $\sigma_0(q) = \text{ff}$.

$$
\begin{array}{ll}
 & [\![p \vee q]\!]_{\sigma_0} \\
= & [\![\neg(\neg p \wedge \neg q)]\!]_{\sigma_0} \\
= & \text{not } ([\![\neg p \wedge \neg q]\!]_{\sigma_0}) \\
= & \text{not } ([\![\neg p]\!]_{\sigma_0} \text{ and } [\![\neg q]\!]_{\sigma_0}) \\
= & \text{not } (\text{not } ([\![p]\!]_{\sigma_0}) \text{ and not } ([\![q]\!]_{\sigma_0})) \\
= & \text{not } (\text{not } (\sigma_0(p)) \text{ and not } (\sigma_0(q))) \\
= & \text{not } (\text{not } (\text{tt}) \text{ and not } (\text{ff})) \\
= & \text{not } (\text{ff and tt}) \\
= & \text{not } (\text{ff}) \\
= & \text{tt}
\end{array}
$$

Exercises (20)

Exercise

One can give the truth table for Boolean formulae in a similar way as the Semantic Boolean operators, i.e., for $\wedge$:

p	q	$p \wedge q$
true	true	true
true	false	false
false	true	false
false	false	false

Give the truth table for the following Boolean formulae:

$(\neg p) \vee (q \wedge r)$

$(\neg p) \equiv (q \vee \neg r)$

Exercises (21)

Exercise
Let $\sigma_0(p) = tt$ and $\sigma_0(q) = tt$.
Give the semantics of $p \equiv q$, i.e., calculate $[\![p \equiv q]\!]_{\sigma_0}$.

Exercise
Show that for any state σ_0 and for propositional variables p and q the following holds $[\![p \vee q]\!]_{\sigma_0} = ([\![p]\!]_{\sigma_0}$ or $[\![q]\!]_{\sigma_0})$.

(22)

Part II

Propositional Interval Temporal Logic

Outline (23)

Propositional ITL (24)

Syntax of Propositional ITL:

- Boolean values: true, false
- Boolean static variables: $p, q, \ldots$
- Boolean state variables: $P, Q, \ldots$
- Boolean operators: $\wedge, \vee, \neg, \supset, \equiv$
- Temporal operators:

skip	(skip),
;	(chop),
*	(chopstar),
$\bigcirc$	(next),
$\Box$	(always),
$\Diamond$	(sometimes), ...

Propositional ITL (25)

Syntax of Propositional ITL in BNF:

$$f ::= \text{true} \mid q \mid Q \mid \neg f \mid f_1 \wedge f_2 \mid \text{skip} \mid f_1 ; f_2 \mid f^*$$

Derived ITL operators:

$$\begin{array}{lcl} \bigcirc f & \widehat{=} & \text{skip} ; f \\ \text{inf} & \widehat{=} & \text{true} ; \text{false} \\ \text{finite} & \widehat{=} & \neg\text{inf} \\ \Diamond f & \widehat{=} & \text{finite} ; f \\ \Box f & \widehat{=} & \neg(\Diamond \neg f) \\ \ldots & & \end{array}$$

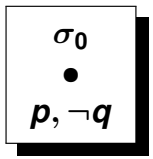
Propositional ITL Examples (26)

Example

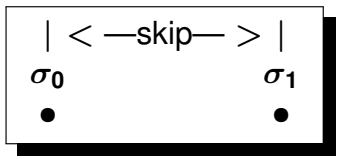
skip
skip ; skip
$Q \wedge$ skip
$Q \wedge$ (skip ; $\neg Q$)
$P \wedge (((\neg R) \wedge \text{skip}) ; Q)$
$Q \wedge (\bigcirc Q) \wedge (\bigcirc\bigcirc\neg Q)$
$(\Diamond P) \wedge (\Diamond Q)$
$\Box Q$
skip*

Informal Semantics (27)

In propositional logic semantics is given wrt a state:
If $\sigma_0(p) = \text{tt}$ and $\sigma_0(q) = \text{ff}$ then $[\![p \vee q]\!]_{\sigma_0} = \text{tt}$

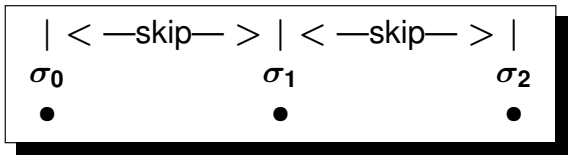


In propositional ITL semantics is given wrt a sequence of states:
Let σ_0 and σ_1 be states then $[\![\text{skip}]\!]_{\sigma_0\sigma_1} = \text{tt}$

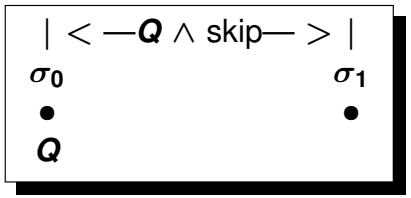


Informal Semantics (28)

Let σ_0, σ_1 and σ_2 be states then $[\![\text{skip ; skip}]\!]_{\sigma_0\sigma_1\sigma_2} = \text{tt}$

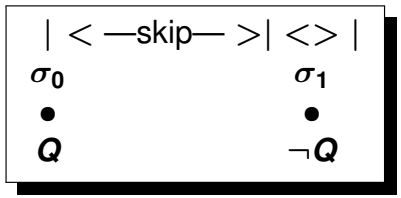


Let σ_0 and σ_1 be states and $\sigma_0(Q) = \text{tt}$ then
$[\![Q \wedge \text{skip}]\!]_{\sigma_0\sigma_1} = \text{tt}$

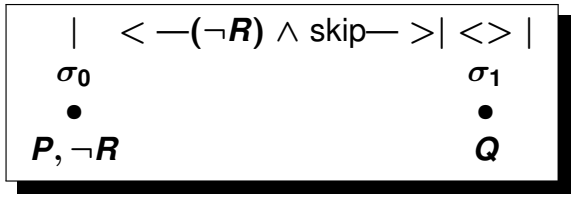


Informal Semantics (29)

Let σ_0 and σ_1 be states and $\sigma_0(Q) = \text{tt}$ and $\sigma_1(Q) = \text{ff}$ then $[\![Q \wedge (\text{skip} ; \neg Q)]\!]_{\sigma_0\sigma_1} = \text{tt}$

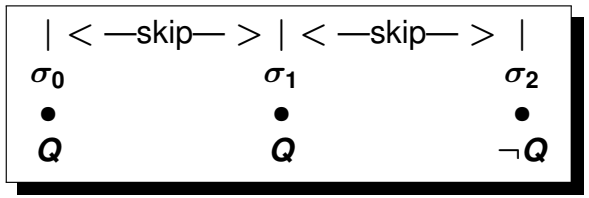


Let σ_0 and σ_1 be states and $\sigma_0(P) = \text{tt}$, $\sigma_0(R) = \text{ff}$ and $\sigma_1(Q) = \text{tt}$ then $[\![P \wedge (((\neg R) \wedge \text{skip}) ; Q)]\!]_{\sigma_0\sigma_1} = \text{tt}$

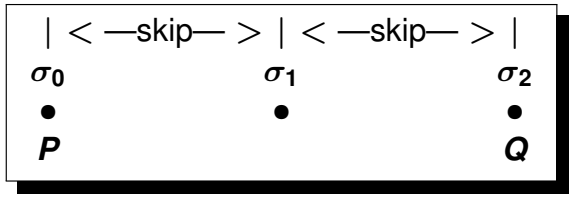


Informal Semantics (30)

Let σ_0, σ_1 and σ_2 be states and $\sigma_0(Q) = \text{tt}$, $\sigma_1(Q) = \text{tt}$ and $\sigma_2(Q) = \text{ff}$ then $[\![Q \wedge (\bigcirc Q) \wedge (\bigcirc\bigcirc\neg Q)]\!]_{\sigma_0\sigma_1\sigma_2} = \text{tt}$

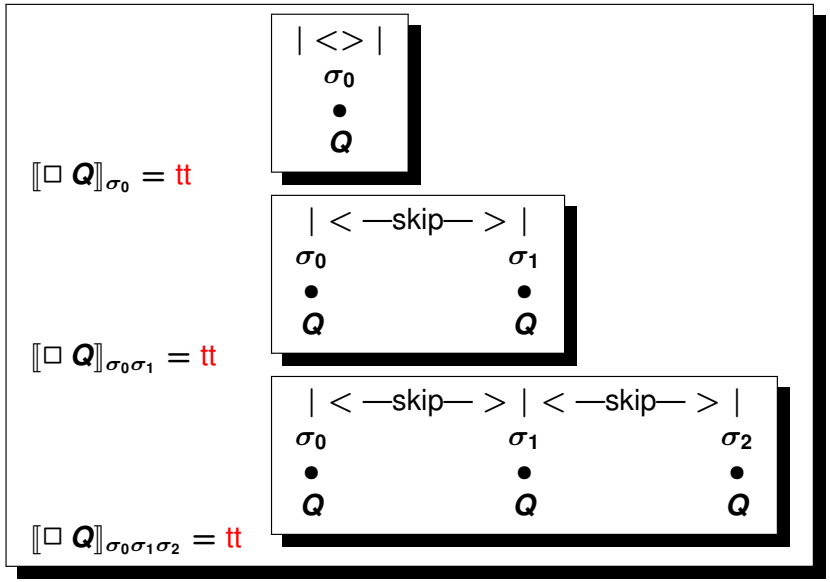


Let σ_0, σ_1 and σ_2 be states and $\sigma_0(P) = \text{tt}$ and $\sigma_2(Q) = \text{tt}$ then $[\![(\Diamond P) \wedge (\Diamond Q)]\!]_{\sigma_0\sigma_1\sigma_2} = \text{tt}$



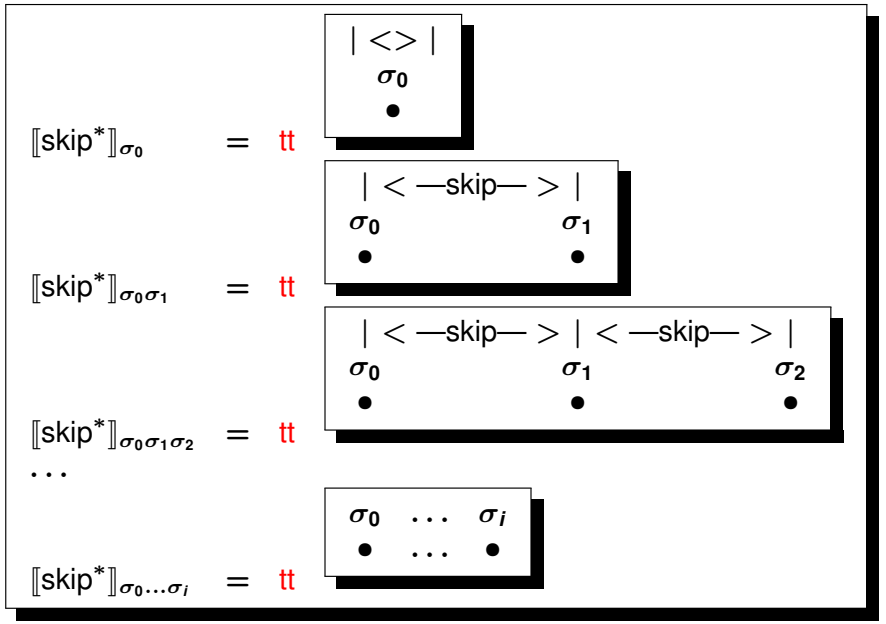
Informal Semantics (31)

Let σ_0, σ_1 and σ_2 be states and $\sigma_0(Q) = \text{tt}$, $\sigma_1(Q) = \text{tt}$ and $\sigma_3(Q) = \text{tt}$ then



Informal Semantics (32)

Let σ_i be a state ($i \geq 0$) then



Interval and Length (33)

An interval σ is a (in)finite sequence of states

$$\sigma : \sigma_0\sigma_1\sigma_2\ldots$$

Let Σ^+ denote the set of all possible non-empty finite intervals.
Let Σ^ω denote the set of infinite intervals.

The length of an interval σ is denoted by $|\sigma|$ and is the number of states minus 1.

Example

$$\begin{array}{ll} \sigma = \sigma_0 & |\sigma| = 0 \\ \sigma = \sigma_0\sigma_1 & |\sigma| = 1 \\ \sigma = \sigma_0\sigma_1\ldots\sigma_n & |\sigma| = n \end{array}$$

Static and State Variables (34)

Static vs State Variables

- Static variables don't change their values within an interval.
- State variables can change their values within an interval.

  $\boldsymbol{Var} = \boldsymbol{StateVar} \bigcup \boldsymbol{StaticVar}$ and
  $\boldsymbol{StateVar} \bigcap \boldsymbol{StaticVar} = \emptyset$.

State variables are denoted by capital first symbols and static variables are denoted by small symbols.

Example

Let $\sigma : \sigma_0\sigma_1$ be an interval where

$$\begin{array}{lcl} \sigma_0(Q) & = & \text{tt} \\ \sigma_0(q) & = & \text{ff} \\ \sigma_1(Q) & = & \text{ff} \\ \sigma_1(q) & = & \text{ff} \end{array}$$

Q is a state variable and q is a static variable.

Prefix, Suffix and Sub Interval (35)

Let $\sigma = \sigma_0\sigma_1\sigma_2\ldots$ be an interval then

- $\sigma_0\ldots\sigma_k$ (where $0 \leq k \leq |\sigma|$) denotes a prefix interval of σ
- $\sigma_k\ldots\sigma_{|\sigma|}$ (where $0 \leq k \leq |\sigma|$) denotes a suffix interval of σ
- $\sigma_k\ldots\sigma_l$ (where $0 \leq k \leq l \leq |\sigma|$) denotes a sub interval of σ

Example

Let $\sigma = \sigma_0\sigma_1\sigma_2\sigma_3$ be an interval then

$\sigma_0\sigma_1$	is a prefix interval of σ
$\sigma_1\sigma_2\sigma_3$	is a suffix interval of σ
$\sigma_1\sigma_2$	is a sub interval of σ

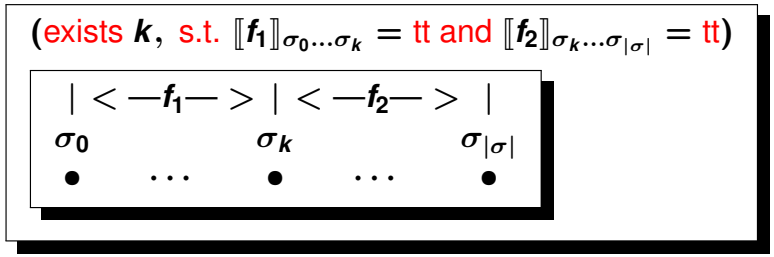
Semantics of PITL (36)

Let $[\![\ldots]\!]$ be the "meaning" function from $\boldsymbol{PITL} \times (\Sigma^+ \cup \Sigma^\omega)$ to $\{\text{tt}, \text{ff}\}$ and let σ be an interval ($\sigma \in \Sigma^+ \cup \Sigma^\omega$) then

$$\begin{array}{lcl} [\![\text{true}]\!]_\sigma & \widehat{=} & \text{tt} \\ [\![q]\!]_\sigma & \widehat{=} & \sigma_0(q) \text{ and} \\ & & \text{for all } 0 < i \leq |\sigma|, \sigma_i(q) = \sigma_0(q) \\ [\![Q]\!]_\sigma & \widehat{=} & \sigma_0(Q) \\ [\![\neg f]\!]_\sigma & \widehat{=} & \text{not } ([\![f]\!]_\sigma) \\ [\![f_1 \wedge f_2]\!]_\sigma & \widehat{=} & [\![f_1]\!]_\sigma \text{ and } [\![f_2]\!]_\sigma \\ [\![\text{skip}]\!]_\sigma = \text{tt} & \text{iff} & |\sigma| = 1 \end{array}$$

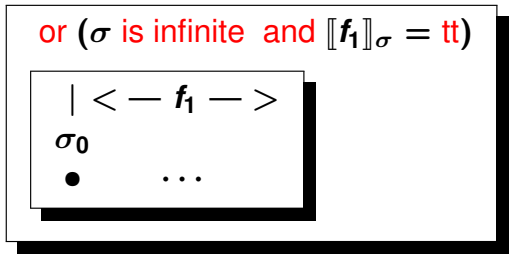
Semantics of PITL (37)

The semantics of 'chop' is as follows $[\![f_1 ; f_2]\!]_\sigma$ = tt iff



Interval σ is a fusion of two intervals $\sigma_0 \ldots \sigma_k$ (satisfies f_1) and $\sigma_k \ldots \sigma_{|\sigma|}$ (satisfies f_2). State σ_k is shared by both.

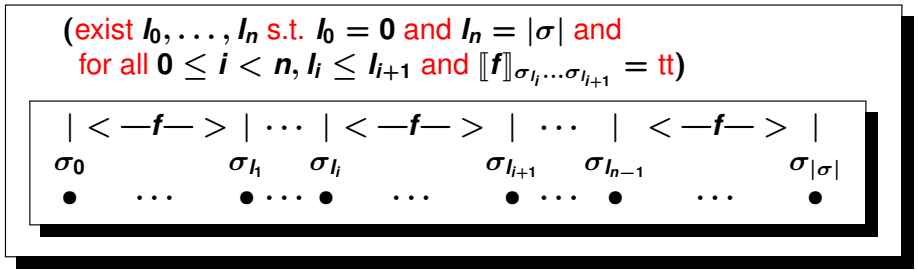
Semantics of PITL (38)



Interval σ is infinite and satisfies f_1, so f_2 is irrelevant.

Semantics of PITL (39)

The semantics of 'chopstar' is as follows $[\![f^*]\!]_\sigma$ = tt iff
if σ is finite then

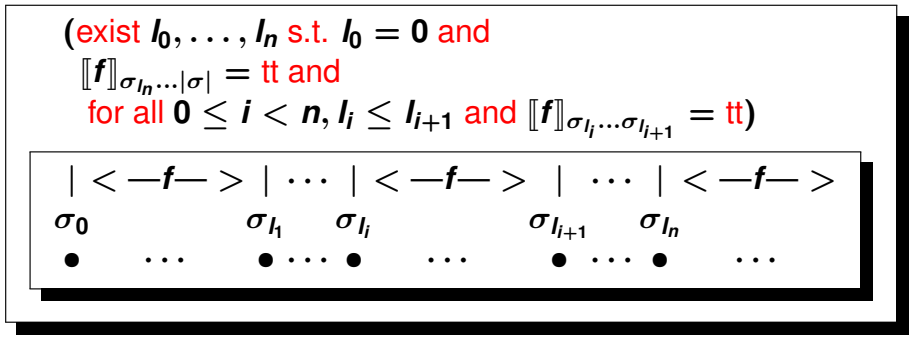


Finite interval σ is the fusion of a finite number of finite sub-intervals each satisfying f.

else

Semantics of PITL (40)

else



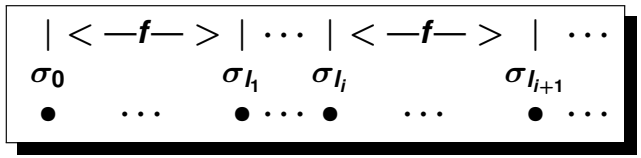
Infinite interval σ is the fusion of a finite number of sub-intervals each satisfying f. Each sub-interval is finite except the last one which is infinite.

or

Semantics of PITL (41)

or

(exist an infinite number of l_i s.t. $l_0 = 0$ and for all $0 \leq i, l_i \leq l_{i+1}$ and $[\![f]\!]_{\sigma_{l_i} \ldots \sigma_{l_{i+1}}} = \text{tt}$)



Infinite interval σ is the fusion of an infinite number of finite sub-intervals each satisfying f.

Derived PITL formulae (42)

We will now introduce some derived temporal formulae.

$\bigcirc f$	$\widehat{=}$	skip ; f	next
$\textcircled{w} f$	$\widehat{=}$	$\neg \bigcirc \neg f$	weak next
more	$\widehat{=}$	$\bigcirc$true	non-empty interval
empty	$\widehat{=}$	$\neg$more	empty interval
inf	$\widehat{=}$	true ; false	infinite interval
finite	$\widehat{=}$	$\neg$inf	finite interval
$\Diamond f$	$\widehat{=}$	finite ; f	sometimes
$\Box f$	$\widehat{=}$	$\neg \Diamond \neg f$	always
$\textcircled{i} f$	$\widehat{=}$	f ; true	some initial subinterval
$\boxed{i} f$	$\widehat{=}$	$\neg(\textcircled{i} \neg f)$	all initial subintervals
$\textcircled{a} f$	$\widehat{=}$	finite ; f ; true	some subinterval
$\boxed{a} f$	$\widehat{=}$	$\neg(\textcircled{a} \neg f)$	all subintervals

Next and Weak Next (43)

- Next, f holds in the next state of the interval:

  $\bigcirc f \ \widehat{=}$ skip ; f

  $[\![\bigcirc f]\!]_\sigma = \text{tt}$ iff $|\sigma| > 0$ and $([\![f]\!]_{\sigma_1 \ldots \sigma_{|\sigma|}} = \text{tt})$

- Weak Next, interval has only one state or f holds in the next state of the interval:

  $\textcircled{w} f \ \widehat{=} \ \neg \bigcirc \neg f$

  $[\![\textcircled{w} f]\!]_\sigma = \text{tt}$ iff $|\sigma| = 0$ or $([\![f]\!]_{\sigma_1 \ldots \sigma_{|\sigma|}} = \text{tt})$

More and Empty (44)

- More, interval with at least two states:

  more $\widehat{=} \bigcirc$true

  $[\![\text{more}]\!]_\sigma = \text{tt}$ iff $|\sigma| > 0$

- Empty, interval with only one state:

  empty $\widehat{=} \neg$more

  $[\![\text{empty}]\!]_\sigma = \text{tt}$ iff $|\sigma| = 0$

Infinite and Finite (45)

- Infinite, interval with an infinite number of states:

  $\mathsf{inf} \ \widehat{=}\ \mathsf{true} ; \mathsf{false}$

  $[\![\mathsf{inf}]\!]_\sigma = \mathsf{tt}$ iff σ is infinite

- Finite, interval with a finite number of states:

  $\mathsf{finite} \ \widehat{=}\ \neg \mathsf{inf}$

  $[\![\mathsf{finite}]\!]_\sigma = \mathsf{tt}$ iff σ is finite

Sometimes and Always (46)

- Sometimes, there exist a suffix interval that satisfies f:

  $\diamond f \ \widehat{=}\ \mathsf{finite} ; f$

  $[\![\diamond f]\!]_\sigma = \mathsf{tt}$ iff (exists a $0 \leq k \leq |\sigma|$, s.t. $[\![f]\!]_{\sigma_k \ldots \sigma_{|\sigma|}} = \mathsf{tt}$)

- Always, all suffix intervals satisfy f:

  $\square f \ \widehat{=}\ \neg \diamond \neg f$

  $[\![\square f]\!]_\sigma = \mathsf{tt}$ iff (for all $0 \leq k \leq |\sigma|$, s.t. $[\![f]\!]_{\sigma_k \ldots \sigma_{|\sigma|}} = \mathsf{tt}$)

Diamond-i and Box-i (47)

- Diamond-i, there exists a prefix interval that satisfies f:

  $\diamond_i f \ \widehat{=}\ f ; \mathsf{true}$

  $[\![\diamond_i f]\!]_\sigma = \mathsf{tt}$ iff (exists a $0 \leq k \leq |\sigma|$, s.t. $[\![f]\!]_{\sigma_0 \ldots \sigma_k} = \mathsf{tt}$)

- Box-i, all prefix intervals satisfy f:

  $\square_i f \ \widehat{=}\ \neg \diamond_i \neg f$

  $[\![\square_i f]\!] = \mathsf{tt}$ iff (for all $0 \leq k \leq |\sigma|$, s.t. $[\![f]\!]_{\sigma_0 \ldots \sigma_k} = \mathsf{tt}$)

Diamond-a and Box-a (48)

- Diamond-a, there exists a sub-interval that satisfies f:

  $\diamond_a f \ \widehat{=}\ \mathsf{finite} ; f ; \mathsf{true}$

  $[\![\diamond_a f]\!] = \mathsf{tt}$ iff (exist $0 \leq k \leq l \leq |\sigma|$ s.t. $[\![f]\!]_{\sigma_k \ldots \sigma_l} = \mathsf{tt}$)

- Box-a, all sub-intervals satisfy f:

  $\square_a f \ \widehat{=}\ \neg \diamond_a \neg f$

  $[\![\square_a f]\!] = \mathsf{tt}$ iff (for all $0 \leq k \leq l \leq |\sigma|$ s.t. $[\![f]\!]_{\sigma_k \ldots \sigma_l} = \mathsf{tt}$)

Programming constructs (49)

- Following are derived temporal formulae that are "programming constructs":

if f_0 then f_1 else f_2	$\widehat{=}$	$(f_0 \wedge f_1) \vee (\neg f_0 \wedge f_2)$	if then else
if f_0 then f_1	$\widehat{=}$	if f_0 then f_1 else empty	if then
halt f	$\widehat{=}$	$\Box(\text{empty} \equiv f)$	halt when
keep f	$\widehat{=}$	$\boxdot(\text{skip} \supset f)$	all unit sub-intervals
while f_0 do f_1	$\widehat{=}$	$(f_0 \wedge f_1)^* \wedge \text{fin} \neg f_0$	while loop
repeat f_0 until f_1	$\widehat{=}$	$f_0 ; (\text{while} \neg f_1 \text{ do } f_0)$	repeat loop

Exercises (50)

Exercise

Let $\sigma = \sigma_0\sigma_1\sigma_2\sigma_3$
Give all prefix intervals of σ
Give all suffix intervals of σ
Give all sub-intervals of σ

Exercise (51)

Exercise
Give the formal semantics of following formulae:

$\Box(\text{empty} \supset f)$
$\boxdot(\text{empty} \supset f)$
$\boxdot(\text{empty} \supset f)$
$\Box(\text{more} \supset f)$
$\boxdot(\text{more} \supset f)$
$\boxdot(\text{more} \supset f)$
$\Box(\text{skip} \supset f)$
$\boxdot(\text{skip} \supset f)$
$\boxdot(\text{skip} \supset f)$

Exercise (52)

Exercise
Give the informal semantics (picture) of following formulae:

if f_0 then f_1 else f_2
fin f
halt f
keep f
while f_0 do f_1
repeat f_0 until f_1

Part III

First Order Logic

Outline

First Order Logic

Syntax of Integer Expressions:

- Integer values: $\mathbf{0}, \mathbf{1}, \ldots,$
- Static integer variable: $\boldsymbol{a}, \boldsymbol{b}, \ldots$
- State integer variable: $\boldsymbol{A}, \boldsymbol{B}, \ldots$
- Integer operators: $+, -, *, **, \mathit{mod}, \mathit{div}, \ldots$

Syntax of Formulae:

- Boolean values: true, false
- Boolean predicates: $\boldsymbol{e_1} = \boldsymbol{e_2}, \boldsymbol{e_1} \neq \boldsymbol{e_2}, \boldsymbol{e_1} < \boldsymbol{e_2}, \boldsymbol{e_1} \leq \boldsymbol{e_2}, \boldsymbol{e_1} > \boldsymbol{e_2}, \boldsymbol{e_1} \geq \boldsymbol{e_2}, \ldots$
- Boolean operators: $\wedge, \vee, \neg, \supset, \equiv,$ $\forall \boldsymbol{v} \bullet \boldsymbol{f}$ (for all $\boldsymbol{v}$ such that $\boldsymbol{f}$ holds)

First Order Logic

Syntax of Integer Expressions in BNF:

$$\boldsymbol{e} ::= \quad \boldsymbol{z} \mid \boldsymbol{a} \mid \boldsymbol{A} \mid \boldsymbol{g}(\boldsymbol{e_1}, \ldots, \boldsymbol{e_n})$$

where $\boldsymbol{z}$ is an integer constant and $\boldsymbol{g}$ an integer operator.

Syntax of First Order Formulae in BNF:

$$\boldsymbol{f} ::= \quad \text{true} \mid \boldsymbol{q} \mid \boldsymbol{Q} \mid \boldsymbol{h}(\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}) \mid \neg \boldsymbol{f} \mid \boldsymbol{f_1} \wedge \boldsymbol{f_2} \mid \forall \boldsymbol{v} \bullet \boldsymbol{f}$$

where $\boldsymbol{h}$ is a Boolean predicate.

Derived formulae:

$$\exists \boldsymbol{v} \bullet \boldsymbol{f} \quad \widehat{=} \quad \neg \forall \boldsymbol{v} \bullet \neg \boldsymbol{f}$$

First Order Logic Examples (57)

Example

$4 > 3$
$A + 5 \leq B \wedge B = D - 7$
$P \equiv (A + 5 \leq B)$
$Q \equiv (B = D - 7)$
$\forall A \bullet (A + 1 > A)$
$A = 8 \wedge \exists b \bullet (A = 2 * *b)$

State (58)

A ***State*** is a union of

- an integer state $\mathbf{State}^e$ which is a mapping from the set of integer variables $\mathbf{Var}^e$ to the set of integer values ***Val*** and
- a Boolean state $\mathbf{State}^b$ which is a mapping from the set of propositional variable $\mathbf{Var}^b$ to the set of Boolean values ***Bool***.

  $\mathbf{State} : (\mathbf{Var}^e \rightarrow \mathbf{Val}) \cup (\mathbf{Var}^b \rightarrow \mathbf{Bool})$

where $\mathbf{Var}^e \cap \mathbf{Var}^b = \emptyset$.
We will use $\sigma_0, \sigma_1, \sigma_2, \ldots$ to denote states and Σ to denote the set of all possible states.

Example

Let ***p*** be a Boolean variable and ***A*** be an integer variable then σ_0 s.t. $\sigma_0(p) =$ tt and $\sigma_0(A) = 5$ is a state.

Semantics of expressions (59)

Let $[\![\ldots]\!]^e$ be the "meaning" (semantic) function from $\mathbf{Expressions} \times \Sigma$ to ***Val*** (integer values) and let σ_0 be a state then

$$
\begin{array}{lcl}
[\![z]\!]^e_{\sigma_0} & = & z \\
[\![a]\!]^e_{\sigma_0} & = & \sigma_0(a) \\
[\![A]\!]^e_{\sigma_0} & = & \sigma_0(A) \\
[\![g(e_1, \ldots, e_n)]\!]^e_{\sigma_0} & = & g([\![e_1]\!]^e_{\sigma_0}, \ldots, [\![e_n]\!]^e_{\sigma_0})
\end{array}
$$

Example

$$
\begin{array}{lcl}
[\![\mathbf{Account}]\!]^e_{\sigma_0} & = & \sigma_0(\mathbf{Account}) \\
[\![\mathbf{deposit}]\!]^e_{\sigma_0} & = & \sigma_0(\mathbf{deposit}) \\
[\![\mathbf{Account} + \mathbf{deposit}]\!]^e_{\sigma_0} & = & [\![\mathbf{Account}]\!]^e_{\sigma_0} + [\![\mathbf{deposit}]\!]^e_{\sigma_0} \\
 & = & \sigma_0(\mathbf{Account}) + \sigma_0(\mathbf{deposit})
\end{array}
$$

Semantics of formulae (60)

Let $[\![\ldots]\!]$ be the "meaning" function from $\mathbf{Formulae} \times \Sigma$ to ***Bool*** (set of Boolean values, $\{$tt, ff$\}$) and let σ_0 be a state then

$$
\begin{array}{lcl}
[\![\text{true}]\!]_{\sigma_0} & = & \text{tt} \\
[\![q]\!]_{\sigma_0} & = & \sigma_0(q) \\
[\![Q]\!]_{\sigma_0} & = & \sigma_0(Q) \\
[\![h(e_1, \ldots, e_n)]\!]_{\sigma_0} = \text{tt} & \text{iff} & h([\![e_1]\!]^e_{\sigma_0}, \ldots, [\![e_n]\!]^e_{\sigma_0}) \\
[\![\neg f]\!]_{\sigma_0} = \text{tt} & \text{iff} & \text{not } ([\![f]\!]_{\sigma_0} = \text{tt}) \\
[\![f_1 \wedge f_2]\!]_{\sigma_0} = \text{tt} & \text{iff} & ([\![f_1]\!]_{\sigma_0} = \text{tt}) \text{ and } ([\![f_2]\!]_{\sigma_0} = \text{tt})
\end{array}
$$

Example

$$
\begin{array}{ll}
([\![\neg(\mathbf{Account} < 0)]\!]_{\sigma_0} = \text{tt}) & \text{iff} \\
\text{not } ([\![\mathbf{Account} < 0]\!]_{\sigma_0} = \text{tt}) & \text{iff} \\
\text{not } ([\![\mathbf{Account}]\!]^e_{\sigma_0} < [\![0]\!]^e_{\sigma_0}) & \text{iff} \\
\text{not } (\sigma_0(\mathbf{Account}) < 0) &
\end{array}
$$

Example (61)

Example

$(\llbracket \mathit{Account} = 50 \wedge \mathit{Deposit} \geq 0 \rrbracket_{\sigma_0} = \mathrm{tt})$ iff
$(\llbracket \mathit{Account} = 50 \rrbracket_{\sigma_0} = \mathrm{tt})$ and $(\llbracket \mathit{Deposit} \geq 0 \rrbracket_{\sigma_0} = \mathrm{tt})$ iff
$(\llbracket \mathit{Account} \rrbracket^e_{\sigma_0} = \llbracket 50 \rrbracket^e_{\sigma_0})$ and $(\llbracket \mathit{Deposit} \rrbracket^e_{\sigma_0} \geq \llbracket 0 \rrbracket^e_{\sigma_0})$ iff
$\sigma_0(\mathit{Account}) = 50$ and $\sigma_0(\mathit{Deposit}) \geq 0$

Semantics of formulae (62)

Let $\sigma_0 \sim_v \sigma'_0$ denote that the states σ_0 and σ'_0 are identical with the possible exception of the mapping for the variable v.

Example

Let σ_0 and σ'_0 be states.

- Let $\sigma_0(\mathit{Cash}) = 0$ and $\sigma_0(\mathit{Deposit}) = 2$
- Let $\sigma'_0(\mathit{Cash}) = 0$ and $\sigma'_0(\mathit{Deposit}) = 3$

Then $\sigma_0 \sim_{\mathit{Deposit}} \sigma'_0$.

Semantics of formulae (63)

The semantics of $\forall v \bullet f$ is defined in terms of $\sigma_0 \sim_v \sigma'_0$

$$\llbracket \forall v \bullet f \rrbracket = \mathrm{tt} \quad \text{iff} \quad (\text{for all } \sigma'_0 \text{ s.t. } \sigma_0 \sim_v \sigma'_0, \llbracket f \rrbracket_{\sigma'_0} = \mathrm{tt})$$

Example

$\llbracket \forall \mathit{Deposit} \bullet \mathit{Deposit} \geq 0 \rrbracket_{\sigma_0} = \mathrm{tt}$ iff
$(\text{for all } \sigma'_0 \text{ s.t. } \sigma_0 \sim_{\mathit{Deposit}} \sigma'_0, \llbracket \mathit{Deposit} \geq 0 \rrbracket_{\sigma_0} = \mathrm{tt})$ iff
$(\text{for all } \sigma'_0 \text{ s.t. } \sigma_0 \sim_{\mathit{Deposit}} \sigma'_0, \sigma'_0(\mathit{Deposit}) \geq 0)$

Exercises (64)

Exercise

Give the semantics of the following formulae

$4 > 3$
$A + 5 \leq B \wedge B = D - 7$
$P \equiv (A + 5 \leq B)$
$Q \equiv (B = D - 7)$
$\forall A \bullet (A + 1 > A)$
$A = 8 \wedge \exists b \bullet (A = 2 * * b)$

Part IV

First Order ITL

Outline

First Order ITL

Syntax of Integer Expressions:

- Integer values: $\mathbf{0}, \mathbf{1}, \ldots,$
- Static integer variable: $\boldsymbol{a}, \boldsymbol{b}, \ldots$
- State integer variable: $\boldsymbol{A}, \boldsymbol{B}, \ldots$
- Integer operators: $+, -, *, **, \mathit{mod}, \mathit{div}, \ldots$
- Temporal integer variable: $\bigcirc \boldsymbol{A}$ (next A), $\bigcirc \boldsymbol{B}, \ldots,$ fin $\boldsymbol{A}$ (fin A), fin $\boldsymbol{B}, \ldots$

Syntax of Formulae:

- Boolean values: true, false
- Boolean static variables: $\boldsymbol{p}, \boldsymbol{q}, \ldots$
- Boolean state variables: $\boldsymbol{P}, \boldsymbol{Q}, \ldots$
- Boolean predicates: $\boldsymbol{e_1} = \boldsymbol{e_2}, \boldsymbol{e_1} \neq \boldsymbol{e_2}, \boldsymbol{e_1} < \boldsymbol{e_2}, \boldsymbol{e_1} \leq \boldsymbol{e_2}, \boldsymbol{e_1} > \boldsymbol{e_2}, \boldsymbol{e_1} \geq \boldsymbol{e_2}, \ldots$
- Boolean operators: $\wedge, \vee, \neg, \supset, \equiv, \forall \boldsymbol{v} \bullet \boldsymbol{f}$
- Temporal operators: skip, $;, *, \bigcirc, \Box, \Diamond, \ldots$

First Order ITL

Syntax of Integer Expressions in BNF:

$$\boldsymbol{e} ::= \quad \boldsymbol{z} \mid \boldsymbol{a} \mid \boldsymbol{A} \mid \boldsymbol{g}(\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}) \mid \bigcirc \boldsymbol{A} \mid \text{fin } \boldsymbol{A}$$

where $\boldsymbol{z}$ is an integer constant and $\boldsymbol{g}$ an integer operator.

Syntax of First Order Formulae in BNF:

$$\begin{aligned} \boldsymbol{f} ::= \quad & \text{true} \mid \boldsymbol{q} \mid \boldsymbol{Q} \mid \boldsymbol{h}(\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}) \mid \neg \boldsymbol{f} \mid \boldsymbol{f_1} \wedge \boldsymbol{f_2} \mid \forall \boldsymbol{v} \bullet \boldsymbol{f} \mid \\ & \text{skip} \mid \boldsymbol{f_1} ; \boldsymbol{f_2} \mid \boldsymbol{f}^* \end{aligned}$$

where $\boldsymbol{h}$ is a Boolean predicate.

First Order ITL (69)

Derived formulae:

$$
\begin{array}{lcl}
\text{false} & \widehat{=} & \neg\text{true} \\
f_1 \vee f_2 & \widehat{=} & \neg(\neg f_1 \wedge \neg f_2) \\
f_1 \supset f_2 & \widehat{=} & \neg f_1 \vee f_2 \\
f_1 \equiv f_2 & \widehat{=} & (f_1 \supset f_2) \wedge (f_2 \supset f_1) \\
\exists v \bullet f & \widehat{=} & \neg\forall v \bullet \neg f \\
\bigcirc f & \widehat{=} & \text{skip} ; f \\
\text{inf} & \widehat{=} & \text{true} ; \text{false} \\
\text{finite} & \widehat{=} & \neg\text{inf} \\
\Diamond f & \widehat{=} & \text{finite} ; f \\
\Box f & \widehat{=} & \neg(\Diamond \neg f) \\
\ldots &&
\end{array}
$$

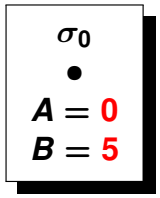
First Order ITL Examples (70)

Example

$(\bigcirc B) = 0 \wedge (\text{skip} ; A = 1)$
$A = 0 \wedge ((B = 1 \wedge \text{skip}) ; (A = 1 \wedge B = 0))$
$A = 0 \wedge \bigcirc(A = 1) \wedge \bigcirc\bigcirc(A = 2)$
$(\Diamond A = 0) \wedge (\Diamond B = 0)$
$\Box A = 5$
$(A = 0 \wedge \text{skip})^* \wedge (\text{fin } A) = 0$

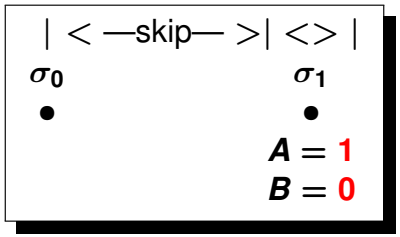
Informal Semantics (71)

In first order logic semantics is given wrt a state:
Let σ_0 be a state and $\sigma_0(A) = 0$ and $\sigma_0(B) = 5$ then
$[\![A = 0 \vee B = 1]\!]_{\sigma_0} = \text{tt}$



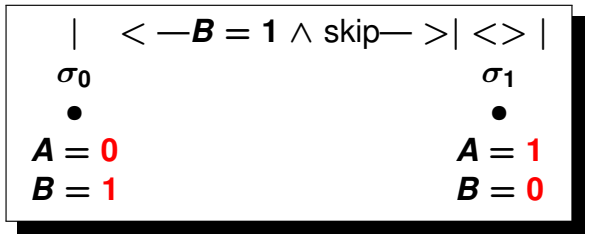
Informal Semantics (72)

In first order ITL semantics is given wrt a sequence of states:
Let σ_0 and σ_1 be states and $\sigma_1(B) = 0$ and $\sigma_1(A) = 1$ then
$[\![(\bigcirc B) = 0 \wedge (\text{skip} ; A = 1)]\!]_{\sigma_0\sigma_1} = \text{tt}$



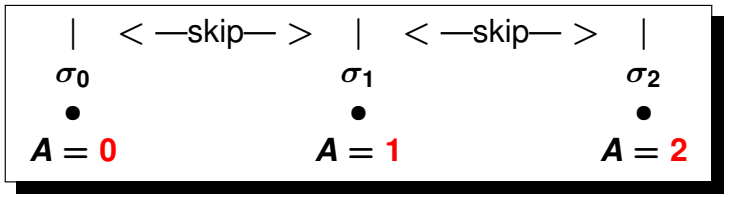
Informal Semantics (73)

Let σ_0 and σ_1 be states and $\sigma_0(A) = 0$, $\sigma_0(B) = 1$, $\sigma_1(A) = 1$ and $\sigma_1(B) = 0$ then
$[\![A = 0 \wedge ((B = 1 \wedge \text{skip}) ; (A = 1 \wedge B = 0))]\!]_{\sigma_0\sigma_1} = \text{tt}$



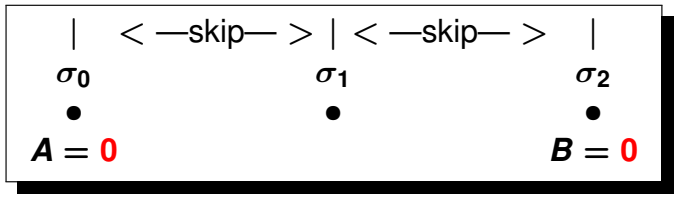
Informal Semantics (74)

Let σ_0, σ_1 and σ_2 be states and $\sigma_0(A) = 0$, $\sigma_1(A) = 1$ and $\sigma_2(A) = 2$ then $[\![A = 0 \wedge \bigcirc(A = 1) \wedge \bigcirc\bigcirc(A = 2)]\!]_{\sigma_0\sigma_1\sigma_2} = \text{tt}$



Informal Semantics (75)

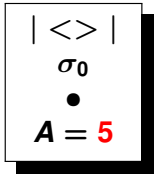
Let σ_0, σ_1 and σ_2 be states and $\sigma_0(A) = 0$ and $\sigma_2(B) = 0$ then $[\![(\Diamond A = 0) \wedge (\Diamond B = 0)]\!]_{\sigma_0\sigma_1\sigma_2} = \text{tt}$



Informal Semantics (76)

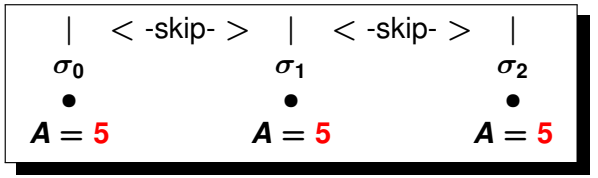
Let σ_0, σ_1 and σ_2 be states and $\sigma_0(A) = 5$, $\sigma_1(A) = 5$ and $\sigma_3(A) = 5$ then

$[\![\Box A = 5]\!]_{\sigma_0} = \text{tt}$



$[\![\Box A = 5]\!]_{\sigma_0\sigma_1} = \text{tt}$

$[\![\Box A = 5]\!]_{\sigma_0\sigma_1\sigma_2} = \text{tt}$



Informal Semantics (77)

Let σ_i be a state ($i \geq i$) and $\sigma_i(A) = 0$ then

$$
\begin{array}{lcl}
[\![(A = 0 \wedge \text{skip})^* \wedge (\text{fin } A) = 0]\!]_{\sigma_0} & = & \text{tt} \\
[\![(A = 0 \wedge \text{skip})^* \wedge (\text{fin } A) = 0]\!]_{\sigma_0\sigma_1} & = & \text{tt} \\
[\![(A = 0 \wedge \text{skip})^* \wedge (\text{fin } A) = 0]\!]_{\sigma_0\sigma_1\sigma_2} & = & \text{tt} \\
\ldots & & \\
[\![(A = 0 \wedge \text{skip})^* \wedge (\text{fin } A) = 0]\!]_{\sigma_0\ldots\sigma_i} & = & \text{tt}
\end{array}
$$

Note: looks similar to $\Box A = 0$ but is it really equal?

Semantics of expressions (78)

Let $[\![\ldots]\!]^e$ be the "meaning" (semantic) function from ***Expressions*** $\times$ $(\Sigma^+ \cup \Sigma^\omega)$ to ***Val*** and let $\sigma = \sigma_0\sigma_1\ldots$ be an interval then

$$
\begin{array}{lcl}
[\![z]\!]^e_\sigma & = & z \\
[\![a]\!]^e_\sigma & = & \sigma_0(a) \text{ and for all } 0 < i \leq |\sigma|, \sigma_i(a) = \sigma_0(a) \\
[\![A]\!]^e_\sigma & = & \sigma_0(A) \\
[\![g(e_1, \ldots, e_n)]\!]^e_\sigma & = & g([\![e_1]\!]^e_\sigma, \ldots, [\![e_n]\!]^e_\sigma) \\
[\![\bigcirc A]\!]^e_\sigma & = & \begin{cases} \sigma_1(A) & \text{implies } |\sigma| > 0 \\ \text{choose-any-from}(Val) & \text{otherwise} \end{cases} \\
[\![\text{fin } A]\!]^e_\sigma & = & \begin{cases} \sigma_{|\sigma|}(A) & \text{implies } \sigma \text{ is finite} \\ \text{choose-any-from}(Val) & \text{otherwise} \end{cases}
\end{array}
$$

Example of semantics (79)

Example

$[\![Account]\!]^e_\sigma = \sigma_0(Account)$

$[\![deposit]\!]^e_\sigma = \sigma_0(deposit)$ and
for all i s.t. $0 < i \leq |\sigma|, \sigma_i(deposit) = \sigma_0(deposit)$

$[\![Account + deposit]\!]^e_\sigma =$
$[\![Account]\!]^e_\sigma + [\![deposit]\!]^e_\sigma =$
$\sigma_0(Account) + \sigma_0(deposit)$
and for all i s.t. $0 < i \leq |\sigma|, \sigma_i(deposit) = \sigma_0(deposit)$

Semantics of formulae (80)

Let $[\![\ldots]\!]$ be the "meaning" function from ***Formulae*** $\times$ $(\Sigma^+ \cup \Sigma^\omega)$ to ***Bool*** (set of Boolean values, $\{\text{tt}, \text{ff}\}$) and let $\sigma = \sigma_0\sigma_1\ldots$ be an interval then

$$
\begin{array}{lcl}
[\![\text{true}]\!]_\sigma & = & \text{tt} \\
[\![q]\!]_\sigma & = & \sigma_0(q) \text{ and for all } 0 < i \leq |\sigma|, \sigma_i(q) = \sigma_0(q) \\
[\![Q]\!]_\sigma & = & \sigma_0(Q) \\
[\![h(e_1, \ldots, e_n)]\!]_\sigma = \text{tt} & \text{iff} & h([\![e_1]\!]^e_\sigma, \ldots, [\![e_n]\!]^e_\sigma) \\
[\![\neg f]\!]_\sigma = \text{tt} & \text{iff} & \text{not } ([\![f]\!]_\sigma = \text{tt}) \\
[\![f_1 \wedge f_2]\!]_\sigma = \text{tt} & \text{iff} & ([\![f_1]\!]_\sigma = \text{tt}) \text{ and } ([\![f_2]\!]_\sigma = \text{tt}) \\
[\![\text{skip}]\!]_\sigma = \text{tt} & \text{iff} & |\sigma| = 1
\end{array}
$$

Example (81)

Example

$(\llbracket \neg(\mathit{Account} < 0) \rrbracket_\sigma = \mathrm{tt})$ iff
not $(\llbracket \mathit{Account} < 0 \rrbracket_\sigma = \mathrm{tt})$ iff
not $(\llbracket \mathit{Account} \rrbracket^e_\sigma < \llbracket 0 \rrbracket^e_\sigma)$ iff
not $(\sigma_0(\mathit{Account}) < 0)$

Example

$(\llbracket \mathit{Account} = 50 \wedge \mathit{Deposit} \geq 0 \rrbracket_\sigma = \mathrm{tt})$ iff
$(\llbracket \mathit{Account} = 50 \rrbracket_\sigma = \mathrm{tt})$ and $(\llbracket \mathit{Deposit} \geq 0 \rrbracket_\sigma = \mathrm{tt})$ iff
$(\llbracket \mathit{Account} \rrbracket^e_\sigma = \llbracket 50 \rrbracket^e_\sigma)$ and $(\llbracket \mathit{Deposit} \rrbracket^e_\sigma \geq \llbracket 0 \rrbracket^e_\sigma)$ iff
$\sigma_0(\mathit{Account}) = 50$ and $\sigma_0(\mathit{Deposit}) \geq 0$

Semantics of formulae (82)

Let $\sigma \sim_v \sigma'$ denote that the intervals σ and σ' are identical with the possible exception of the mapping for the variable v.

Example

Let σ and σ' be intervals.

- Let $\sigma_0(\mathit{Cash}) = 0$ and $\sigma_0(\mathit{Deposit}) = 2$.
  Let $\sigma_1(\mathit{Cash}) = 1$ and $\sigma_1(\mathit{Deposit}) = 4$.
  Let $\sigma_2(\mathit{Cash}) = 1$ and $\sigma_2(\mathit{Deposit}) = 3$.
- Let $\sigma'_0(\mathit{Cash}) = 0$ and $\sigma'_0(\mathit{Deposit}) = 3$.
  Let $\sigma'_1(\mathit{Cash}) = 1$ and $\sigma'_0(\mathit{Deposit}) = 2$.
  Let $\sigma'_2(\mathit{Cash}) = 1$ and $\sigma'_2(\mathit{Deposit}) = 3$.

Then $\sigma \sim_{\mathit{Deposit}} \sigma'$.

Semantics of formulae (83)

The semantics of $\forall v \bullet f$ is defined in terms of $\sigma \sim_v \sigma'$

$\llbracket \forall v \bullet f \rrbracket = \mathrm{tt}$ iff (for all σ' s.t. $\sigma \sim_v \sigma'$, $\llbracket f \rrbracket_{\sigma'} = \mathrm{tt}$)

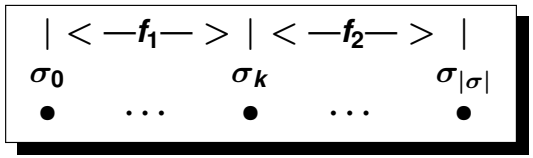
Example

$\llbracket \forall \mathit{Deposit} \bullet \mathit{Deposit} \geq 0 \rrbracket_\sigma = \mathrm{tt}$ iff
(for all σ' s.t. $\sigma \sim_{\mathit{Deposit}} \sigma'$, $\llbracket \mathit{Deposit} \geq 0 \rrbracket_\sigma = \mathrm{tt}$) iff
(for all σ' s.t. $\sigma \sim_{\mathit{Deposit}} \sigma'$, $\sigma'_0(\mathit{Deposit}) \geq 0$)

Semantics of formulae (84)

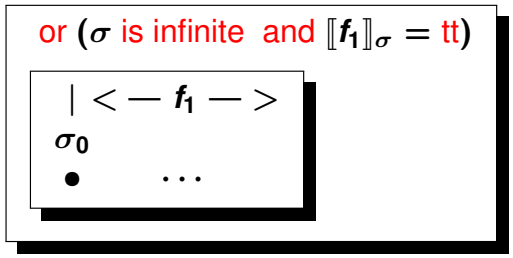
The semantics of 'chop' is as follows $\llbracket f_1 ; f_2 \rrbracket_\sigma = \mathrm{tt}$ iff

(exists k, s.t. $\llbracket f_1 \rrbracket_{\sigma_0 \ldots \sigma_k} = \mathrm{tt}$ and $\llbracket f_2 \rrbracket_{\sigma_k \ldots \sigma_{|\sigma|}} = \mathrm{tt}$)



Interval σ is a fusion of two intervals $\sigma_0 \ldots \sigma_k$ (satisfies f_1) and $\sigma_k \ldots \sigma_{|\sigma|}$ (satisfies f_2). State σ_k is shared by both.

Semantics of formulae (85)

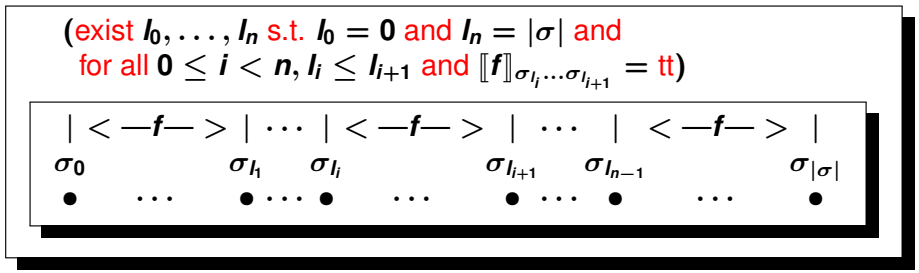


Interval σ is infinite and satisfies f_1, so f_2 is irrelevant.

Semantics of formulae (86)

The semantics of 'chopstar' is as follows $[\![f^*]\!]$ = tt iff

if σ is finite then

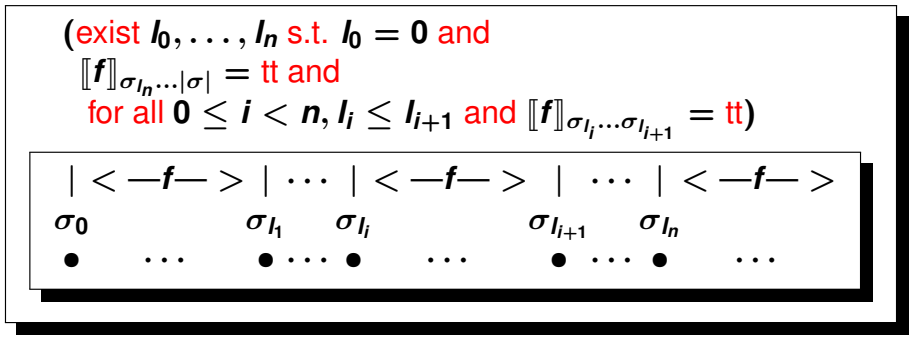


Finite interval σ is the fusion of a finite number of finite sub-intervals each satisfying f.

else

Semantics of formulae (87)

else

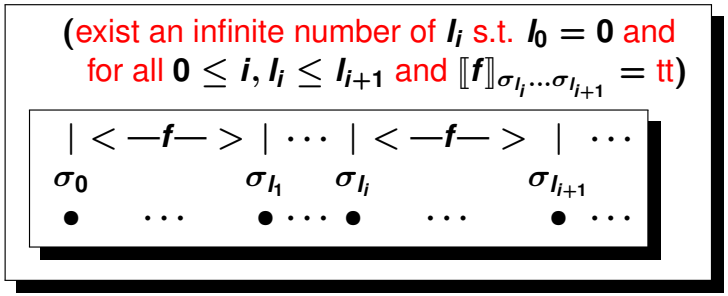


Infinite interval σ is the fusion of a finite number of sub-intervals each satisfying f. Each sub-interval is finite except the last one which is infinite.

or

Semantics of formulae (88)

or



Infinite interval σ is the fusion of an infinite number of finite sub-intervals each satisfying f.

Derived temporal formulae (89)

Here are derived ITL formulae that use temporal variables:

$A := e$	$\widehat{=}$	$(\bigcirc A) = e$	assignment
$A \leftarrow e$	$\widehat{=}$	finite $\wedge$ (fin A) $= e$	temporal assignment
A gets e	$\widehat{=}$	$\boxdot$(skip $\supset A \leftarrow e$)	gets
stable A	$\widehat{=}$	A gets A	stability
len(n)	$\widehat{=}$	$\exists l \bullet (l = 0) \wedge (l$ gets $l + 1)$ $\wedge (l \leftarrow n)$	interval length

Exercise (90)

Exercise

Give the semantics of the following formulae:

true
false
$f_1 \vee f_2$
$f_1 \supset f_2$
$f_1 \equiv f_2$
$\exists v \bullet f$
if f_0 then f_1 else f_2

Exercise (91)

Exercise

Give the formal semantics of (skip $\wedge$ $Account = 50)^*$

Exercise (92)

Exercise

Give the informal semantics (picture) of following formulae:

$A = 0 \wedge$ skip $\wedge A := A + 1$
len(6)
$A = 0 \wedge$ len(2) $\wedge A \leftarrow A + 1$
len(4) $\wedge A = 3 \wedge A$ gets $A + 1$
len(3) $\wedge A = 2 \wedge$ stable A

Exercise

Given informal specification

- *the system's behaviour consists of only two states,*
- *in the initial state the variable* ***Account*** *is equal to* **50** *and*
- *in the next state it is increased by* **100**.

Give the corresponding ITL formula.

Exercise

Given the following "formal specification"

$$\begin{aligned}\textbf{\textit{Spec}} = \{\sigma : \;& \sigma = \sigma_0 \ldots \sigma_n \textit{ and} \\ & \sigma_0(\textbf{\textit{Cash}}) = \textbf{\textit{initial}} \textit{ and} \\ & \sigma_{i+1}(\textbf{\textit{Cash}}) = \sigma_i(\textbf{\textit{Cash}}) + \mathbf{50} \\ & \textit{for } \mathbf{0} \leq \textbf{\textit{i}} < \textbf{\textit{n}} \\ \}\end{aligned}$$

Give the corresponding ITL formula.