Interval Temporal Logic

Antonio Cau and Ben Moszkowski

2019-01-02

HTML version of the ITL home page

Abstract

Interval Temporal Logic (ITL) is a flexible notation for both propositional and first-order reasoning about periods of time found in descriptions of hardware and software systems. Unlike most temporal logics, ITL can handle both sequential and parallel composition and offers powerful and extensible specification and proof techniques for reasoning about properties involving safety, liveness and projected time [129]. Timing constraints are expressible and furthermore most imperative programming constructs can be viewed as formulas in a slightly modified version of ITL [120]. Tempura provides an executable framework for developing and experimenting with suitable ITL specifications. In addition, ITL and its mature executable subset Tempura [152] have been extensively used to specify the properties of real-time systems where the primitive circuits can directly be represented by a set of simple temporal formulae. In addition, Tempura has been applied to hardware simulation and other areas where timing is important.

Contents

Table 1: Syntax of finite ITL

Integer Expressions	
$ie ::=$	$z \mid a \mid A \mid ig(ie_1, \ldots, ie_n) \mid \bigcirc A \mid \text{fin } A$
Boolean Expressions	
$be ::=$	$b \mid q \mid Q \mid bg(be_1, \ldots, be_n) \mid \bigcirc Q \mid \text{fin } Q$
Formulae	
$f ::=$	$\text{true} \mid h(e_1, \ldots, e_n) \mid \neg f \mid f_1 \wedge f_2 \mid \forall v \bullet\ f \mid \text{skip} \mid f_1\ ;\ f_2 \mid f^*$

1 Finite Interval Temporal Logic

The key notion of ITL is an *interval*. An interval σ is considered to be a finite sequence of states $\sigma_0 \ldots \sigma_k$.

1.1 Syntax

The syntax of finite ITL is defined in Table 1 where
z denotes an integer value,
a denotes a static integer variable,
A denotes a state integer variable,
b denotes a Boolean value,
q denotes a static propositional variable,
Q denotes a state propositional variable,
ig denotes a integer function symbol,
bg denotes a Boolean function symbol,
v denotes static or state (integer or Boolean) variable,
e_i denotes a Boolean or integer expression,
h denotes a predicate symbol.

1.2 Semantics

Each state σ_i is the union of the mapping from the set of integer variables IntVar to the set of integer values $\mathbb{Z}$ and the mapping from propositional variables PropVar to set of Boolean values $\{\text{tt}, \text{ff}\}$.

Each interval has at least one state. The *length* $|\sigma|$ of an interval $\sigma_0 \ldots \sigma_n$ is equal to n, one less than the number of states in the interval (this has always been a convention in ITL), i.e., a one state interval has length 0. Let $\sigma = \sigma_0\sigma_1\sigma_2 \ldots$ be an interval then

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

The informal semantics of the most interesting constructs are as follows:

- $\bigcirc A$: if the interval is non-empty then the value of A in the next state of that interval else an arbitrary value.
- fin A: the value of A in the last state of the interval.
- skip unit interval (length 1).

Table 2: Semantics of finite ITL

$$
\begin{array}{lll}
\mathsf{E}[\![z]\!](\sigma) & = & z \\
\mathsf{E}[\![a]\!](\sigma) & = & \sigma_0(a) \text{ and for all } 0 < i \leq |\sigma|, \sigma_i(a) = \sigma_0(a) \\
\mathsf{E}[\![A]\!](\sigma) & = & \sigma_0(A) \\
\mathsf{E}[\![ig(ie_1, \ldots, ie_n)]\!](\sigma) & = & ig(\mathsf{E}[\![ie_1]\!](\sigma), \ldots, \mathsf{E}[\![ie_n]\!](\sigma)) \\
\mathsf{E}[\![\bigcirc A]\!](\sigma) & = & \begin{array}{ll} \sigma_1(A) & \text{if } |\sigma| > 0 \\ \text{choose-any-from}(\mathbb{Z}) & \text{otherwise} \end{array} \\
\mathsf{E}[\![\text{fin } A]\!](\sigma) & = & \sigma_{|\sigma|}(A) \\
\mathsf{E}[\![b]\!](\sigma) & = & b \\
\mathsf{E}[\![q]\!](\sigma) & = & \sigma_0(q) \text{ and for all } 0 < i \leq |\sigma|, \sigma_i(q) = \sigma_0(q) \\
\mathsf{E}[\![Q]\!](\sigma) & = & \sigma_0(Q) \\
\mathsf{E}[\![bg(be_1, \ldots, be_n)]\!](\sigma) & = & bg(\mathsf{E}[\![be_1]\!](\sigma), \ldots, \mathsf{E}[\![be_n]\!](\sigma)) \\
\mathsf{E}[\![\bigcirc Q]\!](\sigma) & = & \begin{array}{ll} \sigma_1(Q) & \text{if } |\sigma| > 0 \\ \text{choose-any-from(Bool)} & \text{otherwise} \end{array} \\
\mathsf{E}[\![\text{fin } Q]\!](\sigma) & = & \sigma_{|\sigma|}(Q) \\
\mathsf{M}[\![\text{true}]\!](\sigma) & = & \text{tt} \\
\mathsf{M}[\![h(e_1, \ldots, e_n)]\!](\sigma) = \text{tt} & \text{iff} & h(\mathsf{E}[\![e_1]\!](\sigma), \ldots, \mathsf{E}[\![e_n]\!](\sigma)) \\
\mathsf{M}[\![\neg f]\!](\sigma) = \text{tt} & \text{iff} & \text{not } (\mathsf{M}[\![f]\!](\sigma) = \text{tt}) \\
\mathsf{M}[\![f_1 \wedge f_2]\!](\sigma) = \text{tt} & \text{iff} & (\mathsf{M}[\![f_1]\!](\sigma) = \text{tt}) \text{ and } (\mathsf{M}[\![f_2]\!](\sigma) = \text{tt}) \\
\mathsf{M}[\![\text{skip}]\!](\sigma) = \text{tt} & \text{iff} & |\sigma| = 1 \\
\mathsf{M}[\![\forall v \bullet\ f]\!](\sigma) = \text{tt} & \text{iff} & (\text{for all } \sigma' \text{ s.t. } \sigma \sim_v \sigma', \mathsf{M}[\![f]\!](\sigma') = \text{tt}) \\
\mathsf{M}[\![f_1 \,;\, f_2]\!](\sigma) = \text{tt} & \text{iff} & \\
\quad (\text{exists } k, \text{ s.t. } \mathsf{M}[\![f_1]\!](\sigma_0 \ldots \sigma_k) = \text{tt and } \mathsf{M}[\![f_2]\!](\sigma_k \ldots \sigma_{|\sigma|}) = \text{tt}) & & \\
\mathsf{M}[\![f^*]\!](\sigma) = \text{tt} & \text{iff} & \\
\quad (\text{exist } l_0, \ldots, l_n \text{ s.t. } l_0 = 0 \text{ and } l_n = |\sigma| \text{ and} & & \\
\quad\ \text{for all } 0 \leq i < n, l_i < l_{i+1} \text{ and } \mathsf{M}[\![f]\!](\sigma_{l_i} \ldots \sigma_{l_{i+1}}) = \text{tt}) & &
\end{array}
$$

- $f_1 \,; f_2$ holds if the interval can be decomposed ("chopped") into a prefix and suffix interval, such that f_1 holds over the prefix interval and f_2 over the suffix interval.

- f^* holds if the interval is decomposable into a finite number of intervals such that for each of them f holds.

Let Σ^+ denote the set of all finite intervals.
Let Expressions denote the set of (integer or Boolean) expressions.
Let Val denote the set of integer or Boolean values ($\mathbb{Z} \cup$ Bool).
Let $\mathsf{E}[\![\ldots]\!](\ )$ denote the meaning function from Expressions $\times\, \Sigma^+$ to Val.
Let Formulae denote the set of ITL formulae.
Let $\mathsf{M}[\![\ldots]\!](\ )$ denote the meaning function from Formulae $\times\, \Sigma^+$ to Bool (set of Boolean values, $\{\text{tt}, \text{ff}\}$).
Let $\sigma = \sigma_0\sigma_1 \ldots$ denote an interval.
We write $\sigma \sim_v \sigma'$ if the intervals σ and σ' are identical with the possible exception of their mappings for the variable v.
Let choose-any-from(Val) denote the choice function that selects an arbitrary value from Val.
The formal semantics is listed in Table 2:

1.3 Derived Constructs

Frequently used derived constructs are listed in Table 3–6.

Table 3: Frequently used non-temporal derived constructs

false	$\triangleq$	$\neg$true	false value
$f_1 \vee f_2$	$\triangleq$	$\neg(\neg f_1 \wedge \neg f_2)$	or
$f_1 \supset f_2$	$\triangleq$	$\neg f_1 \vee f_2$	implies
$f_1 \equiv f_2$	$\triangleq$	$(f_1 \supset f_2) \wedge (f_2 \supset f_1)$	equivalent
$\exists v \bullet f$	$\triangleq$	$\neg \forall v \bullet \neg f$	exists

Table 4: Frequently used temporal derived constructs

$\bigcirc f$	$\triangleq$	skip ; f	next
more	$\triangleq$	$\bigcirc$true	non-empty interval
empty	$\triangleq$	$\neg$more	empty interval
$\Diamond f$	$\triangleq$	true ; f	sometimes
$\Box f$	$\triangleq$	$\neg \Diamond \neg f$	always
$\textcircled{w} f$	$\triangleq$	$\neg \bigcirc \neg f$	weak next
$\Diamond_i f$	$\triangleq$	f ; true	some initial subinterval
$\boxed{i} f$	$\triangleq$	$\neg(\Diamond_i \neg f)$	all initial subintervals
$\Diamond_a f$	$\triangleq$	true ; f ; true	some subinterval
$\boxed{a} f$	$\triangleq$	$\neg(\Diamond_a \neg f)$	all subintervals
$\boxed{s}(f)$	$\triangleq$	empty $\vee$ $(\boxed{i}(f)$; skip$)$	all strict initial intervals
$\Diamond_s(f)$	$\triangleq$	$\neg(\boxed{s}(\neg f))$	some strict initial interval
$\triangleright(f)$	$\triangleq$	$f \wedge \boxed{s}(\neg f)$	first occurrence

Table 5: Frequently used concrete derived constructs

if f_0 then f_1 else f_2	$\triangleq$	$(f_0 \wedge f_1) \vee (\neg f_0 \wedge f_2)$	if then else
if f_0 then f_1	$\triangleq$	if f_0 then f_1 else true	if then
fin f	$\triangleq$	$\Box(\mathsf{empty} \supset f)$	final state
halt f	$\triangleq$	$\Box(\mathsf{empty} \equiv f)$	terminate interval when
keep f	$\triangleq$	$\boxed{a}(\mathsf{skip} \supset f)$	all unit subintervals
keepnow f	$\triangleq$	$\Diamond_i(\mathsf{skip} \wedge f)$	initial unit subinterval
while f_0 do f_1	$\triangleq$	$(f_0 \wedge f_1)^* \wedge \mathsf{fin}\ \neg f_0$	while loop
repeat f_0 until f_1	$\triangleq$	f_0 ; (while $\neg f_1$ do f_0)	repeat loop
$f_1 \mapsto f_0$	$\triangleq$	$\boxed{i}(f_1 \supset \mathsf{fin}\ f_0)$	always followed by
$f_1 \leftrightarrow f_0$	$\triangleq$	$\boxed{i}(f_1 \equiv \mathsf{fin}\ f_0)$	strong followed by

Table 6: Frequently used derived constructs related to expressions

$Y := e$	$\triangleq$	$(\bigcirc Y) = e$	assignment
$Y \approx e$	$\triangleq$	$\Box(Y = e)$	equal in interval
$Y \leftarrow e$	$\triangleq$	$(\mathsf{fin}\ Y) = e$	temporal assignment
Y gets e	$\triangleq$	keep $(Y \leftarrow e)$	gets
stable Y	$\triangleq$	Y gets Y	stability
padded Y	$\triangleq$	(stable (Y) ; skip) $\vee$ empty	padded expression
$Y <\sim e$	$\triangleq$	$(Y \leftarrow e) \wedge$ padded Y	padded temporal assignment
intlen (e)	$\triangleq$	$\exists I \bullet (I = 0) \wedge (I \text{ gets } I + 1) \wedge (I \leftarrow e)$	interval length

1.4 Propositional proof system

In Table 7 we list the propositional axioms and rules for finite ITL.

Table 7: Propositional Axioms and Rules for finite ITL.

ChopAssoc	$\vdash$	$(f_0 ; f_1) ; f_2 \equiv f_0 ; (f_1 ; f_2)$
OrChopImp	$\vdash$	$(f_0 \lor f_1) ; f_2 \supset (f_0 ; f_2) \lor (f_1 ; f_2)$
ChopOrImp	$\vdash$	$f_0 ; (f_1 \lor f_2) \supset (f_0 ; f_1) \lor (f_0 ; f_2)$
EmptyChop	$\vdash$	$\mathrm{empty} ; f \equiv f$
ChopEmpty	$\vdash$	$f ; \mathrm{empty} \equiv f$
BiBoxChopImpChop	$\vdash$	$\boxed{i}(f_0 \supset f_1) \land \Box(f_2 \supset f_3) \supset (f_0 ; f_2) \supset (f_1 ; f_3)$
StateImpBi	$\vdash$	$p \supset \boxed{i}\, p$
NextImpNotNextNot	$\vdash$	$\bigcirc f \supset \neg\bigcirc\neg f$
BoxInduct	$\vdash$	$f \land \Box(f \supset \textcircled{w} f) \supset \Box f$
ChopStarEqv	$\vdash$	$f^* \equiv (\mathrm{empty} \lor ((f \land \mathrm{more}) ; f^*))$
MP	$\vdash$	$f_0 \supset f_1, \quad \vdash \quad f_0 \quad \Rightarrow \quad \vdash \quad f_1$
BoxGen	$\vdash$	$f_0 \quad \Rightarrow \quad \vdash \quad \Box f_0$
BiGen	$\vdash$	$f_0 \quad \Rightarrow \quad \vdash \quad \boxed{i}\, f_0$

1.5 First order proof system

Some axioms for the first order case are shown in Table 8.
Let v refer to both static and state variables.
We denote by $f\,[e\,/\,v]$ that in formula f expression e is substituted for variable v.

Table 8: Some First Order Axioms and Rules for finite ITL.

ForallSub	$\vdash \quad \forall v \bullet\ f \supset f\,[e\,/\,v]$, where the expression e has the same data and temporal type as the variable v and is free for v in f.
ForallImplies	$\vdash \quad \forall v \bullet (f_1 \supset f_2) \supset (f_1 \supset \forall v \bullet\ f_2)$, where v doesn't occur freely in f_1.
SubstAxiom	$\vdash \quad \Box(v_1 = v_2) \supset f \equiv f\,[v_2\,/\,v_1]$.
StaticWeakNext	$\vdash \quad w \supset \textcircled{w} w$, where w only contains static variables.
ExistsChopRight	$\vdash \quad \exists v \bullet (f_1 ; f_2) \supset (\exists v \bullet\ f_1) ; f_2$, where v doesn't occur freely in f_2.
ExistsChopLeft	$\vdash \quad \exists v \bullet (f_1 ; f_2) \supset f_1 ; (\exists v \bullet\ f_2)$, where v doesn't occur freely in f_1.
ForallGen	$\vdash \quad f \Rightarrow \ \vdash \quad \forall v \bullet\ f$, for any variable v.

Table 9: Syntax of finite and infinite ITL

Expressions	
$ie ::=$	$z \mid a \mid A \mid ig(ie_1, \ldots, ie_n) \mid \bigcirc A \mid \text{fin } A$
Boolean Expressions	
$be ::=$	$b \mid q \mid Q \mid bg(be_1, \ldots, be_n) \mid \bigcirc Q \mid \text{fin } Q$
Formulae	
$f ::=$	$\text{true} \mid h(e_1, \ldots, e_n) \mid \neg f \mid f_1 \wedge f_2 \mid \forall v \bullet\ f \mid \text{skip} \mid f_1 \mathbin{;} f_2 \mid f^*$

2 Finite and Infinite Interval Temporal Logic

The key notion of ITL is an *interval*. An interval σ is considered to be a (in)finite sequence of states $\sigma_0\sigma_1\ldots$.

2.1 Syntax

The syntax of ITL is defined in Table 9 where
z denotes an integer value,
a denotes a static integer variable,
A denotes a state integer variable,
b denotes a Boolean value,
q denotes a static propositional variable,
Q denotes a state propositional variable,
ig denotes a integer function symbol,
bg denotes a Boolean function symbol,
v denotes static or state integer variable,
e_i denotes a Boolean or integer expression,
h denotes a predicate symbol.

2.2 Semantics

Each state σ_i is the union of the mapping from the set of integer variables IntVar to the set of integer values $\mathbb{Z}$ and the mapping from propositional variables PropVar to set of Boolean values $\{\text{tt}, \text{ff}\}$.

Each interval has at least one state. The *length* $|\sigma|$ of an interval $\sigma_0\ldots\sigma_n$ is equal to n, one less than the number of states in the interval (this has always been a convention in ITL), i.e., a one state interval has length 0. Let $\sigma = \sigma_0\sigma_1\sigma_2\ldots$ be an interval then

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

The informal semantics of the most interesting constructs are as follows:

- $\bigcirc A$: if interval is non-empty then the value of A in the next state of that interval else an arbitrary value.
- fin A: if interval is finite then the value of A in the last state of that interval else an arbitrary value.

- skip unit interval (length 1).

- f_1 ; f_2 holds if the interval can be decomposed ("chopped") into a prefix and suffix interval, such that f_1 holds over the prefix and f_2 over the suffix, or if the interval is infinite and f_1 holds for that interval.

- f^* holds if the interval is decomposable into a finite number of intervals such that for each of them f holds, or the interval is infinite and can be decomposed into an infinite number of finite intervals for which f holds.

Let Σ^+ denote the set of all finite intervals and Σ^ω denotes the set of all infinite intervals.
Let Expressions denote the set of (integer or Boolean) expressions.
Let Val denote the set of integer or Boolean values ($\mathbb{Z} \cup \mathrm{Bool}$).
Let $\mathrm{E}[\![\ldots]\!](\ )$ denote the meaning function from $Expressions \times (\Sigma^+ \cup \Sigma^\omega)$ to Val.
Let Formulae denote the set of ITL formulae.
Let $\mathrm{M}[\![\ldots]\!](\ )$ denote the meaning function from $Formulae \times (\Sigma^+ \cup \Sigma^\omega)$ to Bool (set of Boolean values, $\{\mathrm{tt}, \mathrm{ff}\}$).
Let $\sigma = \sigma_0\sigma_1\ldots$ denote an interval.
We write $\sigma \sim_v \sigma'$ if the intervals σ and σ' are identical with the possible exception of their mappings for the variable v.
Let choose-any-from(Val) denote the choice function that selects an arbitrary value from Val.
The formal semantics is listed in Table 10:

Table 10: Semantics of finite and infinite ITL

$$
\begin{array}{lcl}
\mathsf{E}[\![z]\!](\sigma) & = & z \\
\mathsf{E}[\![a]\!](\sigma) & = & \sigma_0(a) \text{ and for all } 0 < i \le |\sigma|, \sigma_i(a) = \sigma_0(a) \\
\mathsf{E}[\![A]\!](\sigma) & = & \sigma_0(A) \\
\mathsf{E}[\![ig(ie_1, \ldots, ie_n)]\!](\sigma) & = & ig(\mathsf{E}[\![ie_1]\!](\sigma), \ldots, \mathsf{E}[\![ie_n]\!](\sigma)) \\
\mathsf{E}[\![\bigcirc A]\!](\sigma) & = & \begin{array}{ll} \sigma_1(A) & \text{if } |\sigma| > 0 \\ \text{choose-any-from}(\mathbb{Z}) & \text{otherwise} \end{array} \\
\mathsf{E}[\![\text{fin } A]\!](\sigma) & = & \begin{array}{ll} \sigma_{|\sigma|}(A) & \text{if } \sigma \text{ is finite} \\ \text{choose-any-from}(\mathbb{Z}) & \text{otherwise} \end{array} \\
\mathsf{E}[\![b]\!](\sigma) & = & b \\
\mathsf{E}[\![q]\!](\sigma) & = & \sigma_0(q) \text{ and for all } 0 < i \le |\sigma|, \sigma_i(q) = \sigma_0(q) \\
\mathsf{E}[\![Q]\!](\sigma) & = & \sigma_0(Q) \\
\mathsf{E}[\![bg(be_1, \ldots, be_n)]\!](\sigma) & = & bg(\mathsf{E}[\![be_1]\!](\sigma), \ldots, \mathsf{E}[\![be_n]\!](\sigma)) \\
\mathsf{E}[\![\bigcirc Q]\!](\sigma) & = & \begin{array}{ll} \sigma_1(Q) & \text{if } |\sigma| > 0 \\ \text{choose-any-from(Bool)} & \text{otherwise} \end{array} \\
\mathsf{E}[\![\text{fin } Q]\!](\sigma) & = & \begin{array}{ll} \sigma_{|\sigma|}(Q) & \text{if } \sigma \text{ is finite} \\ \text{choose-any-from(Bool)} & \text{otherwise} \end{array} \\
\mathsf{M}[\![\text{true}]\!](\sigma) & = & \text{tt} \\
\mathsf{M}[\![h(e_1, \ldots, e_n)]\!](\sigma) = \text{tt} & \text{iff} & h(\mathsf{E}[\![e_1]\!](\sigma), \ldots, \mathsf{E}[\![e_n]\!](\sigma)) \\
\mathsf{M}[\![\neg f]\!](\sigma) = \text{tt} & \text{iff} & \text{not } (\mathsf{M}[\![f]\!](\sigma) = \text{tt}) \\
\mathsf{M}[\![f_1 \wedge f_2]\!](\sigma) = \text{tt} & \text{iff} & (\mathsf{M}[\![f_1]\!](\sigma) = \text{tt}) \text{ and } (\mathsf{M}[\![f_2]\!](\sigma) = \text{tt}) \\
\mathsf{M}[\![\text{skip}]\!](\sigma) = \text{tt} & \text{iff} & |\sigma| = 1 \\
\mathsf{M}[\![\forall v \bullet\ f]\!](\sigma) = \text{tt} & \text{iff} & (\text{for all } \sigma' \text{ s.t. } \sigma \sim_v \sigma', \mathsf{M}[\![f]\!](\sigma') = \text{tt}) \\
\mathsf{M}[\![f_1\, ;\, f_2]\!](\sigma) = \text{tt} & \text{iff} &
\end{array}
$$

$\quad$ (exists k, s.t. $\mathsf{M}[\![f_1]\!](\sigma_0 \ldots \sigma_k) = \text{tt}$ and $\mathsf{M}[\![f_2]\!](\sigma_k \ldots \sigma_{|\sigma|}) = \text{tt}$)

$\quad$ or (σ is infinite and $\mathsf{M}[\![f_1]\!](\sigma) = \text{tt}$)

$\mathsf{M}[\![f^*]\!](\sigma) = \text{tt}$ iff

if σ is finite then

$\quad$ (exist $l_0, \ldots, l_n$ s.t. $l_0 = 0$ and $l_n = |\sigma|$ and

$\quad$ for all $0 \le i < n$, $l_i < l_{i+1}$ and $\mathsf{M}[\![f]\!](\sigma_{l_i} \ldots \sigma_{l_{i+1}}) = \text{tt}$)

else

$\quad$ (exist $l_0, \ldots, l_n$ s.t. $l_0 = 0$ and

$\quad$ $\mathsf{M}[\![f]\!](\sigma_{l_n} \ldots \sigma_{|\sigma|}) = \text{tt}$ and

$\quad$ for all $0 \le i < n$, $l_i < l_{i+1}$ and $\mathsf{M}[\![f]\!](\sigma_{l_i} \ldots \sigma_{l_{i+1}}) = \text{tt}$)

$\quad$ or

$\quad$ (exist an infinite number of l_i s.t. $l_0 = 0$ and

$\quad$ for all $0 \le i$, $l_i < l_{i+1}$ and $\mathsf{M}[\![f]\!](\sigma_{l_i} \ldots \sigma_{l_{i+1}}) = \text{tt}$)

2.3 Derived Constructs

Frequently used derived constructs are listed in Table 11–14.

Table 11: Frequently used non-temporal derived constructs

false	$\triangleq$	$\neg$true	false value
$f_1 \vee f_2$	$\triangleq$	$\neg(\neg f_1 \wedge \neg f_2)$	or
$f_1 \supset f_2$	$\triangleq$	$\neg f_1 \vee f_2$	implies
$f_1 \equiv f_2$	$\triangleq$	$(f_1 \supset f_2) \wedge (f_2 \supset f_1)$	equivalent
$\exists v \bullet f$	$\triangleq$	$\neg \forall v \bullet \neg f$	exists

Table 12: Frequently used temporal derived constructs

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

Table 13: Frequently used concrete derived constructs

if f_0 then f_1 else f_2	$\triangleq$	$(f_0 \wedge f_1) \vee (\neg f_0 \wedge f_2)$	if then else
if f_0 then f_1	$\triangleq$	if f_0 then f_1 else true	if then
fin f	$\triangleq$	$\Box(\text{empty} \supset f)$	final state
sfin f	$\triangleq$	$\neg(\text{fin}\,(\neg f))$	strong final state
halt f	$\triangleq$	$\Box(\text{empty} \equiv f)$	terminate interval when
shalt f	$\triangleq$	$\neg(\text{halt}\,(\neg f))$	strong terminate interval when
keep f	$\triangleq$	$\boxed{a}(\text{skip} \supset f)$	all unit subintervals
f^ω	$\triangleq$	$\text{isinf}\,(\text{isfin}\,(f)^*)$	infinite chopstar
fstar (f)	$\triangleq$	$\text{isfin}\,(\text{isfin}\,(f)^*) \vee$ $\text{isfin}\,(\text{isfin}\,(f)^*)\,;\,\text{isinf}\,(f)$	finite chopstar
while f_0 do f_1	$\triangleq$	$(f_0 \wedge f_1)^* \wedge \text{fin}\ \neg f_0$	while loop
repeat f_0 until f_1	$\triangleq$	$f_0\,;\,(\text{while}\ \neg f_1\ \text{do}\ f_0)$	repeat loop

Table 14: Frequently used derived constructs related to expressions

$Y := e$	$\triangleq$	$(\bigcirc Y) = e$	assignment
$Y \approx e$	$\triangleq$	$\Box(Y = e)$	equal in interval
$Y \leftarrow e$	$\triangleq$	$\text{finite} \wedge (\text{fin}\ Y) = e$	temporal assignment
Y gets e	$\triangleq$	$\text{keep}\,(Y \leftarrow e)$	gets
stable Y	$\triangleq$	Y gets Y	stability
padded Y	$\triangleq$	$(\text{stable}\,(Y)\,;\,\text{skip}) \vee \text{empty}$	padded expression
$Y <\sim e$	$\triangleq$	$(Y \leftarrow e) \wedge \text{padded}\ Y$	padded temporal assignment
intlen (e)	$\triangleq$	$\exists I \bullet (I = 0) \wedge (I\ \text{gets}\ I + 1) \wedge (I \leftarrow e)$	interval length

2.4 Propositional proof system

In Table 15 we list the propositional axioms and rules for finite and infinite ITL.

Table 15: Propositional Axioms and Rules for finite and infinite ITL.

ChopAssoc	$\vdash$	$(f_0 ; f_1) ; f_2 \equiv f_0 ; (f_1 ; f_2)$
OrChopImp	$\vdash$	$(f_0 \vee f_1) ; f_2 \supset (f_0 ; f_2) \vee (f_1 ; f_2)$
ChopOrImp	$\vdash$	$f_0 ; (f_1 \vee f_2) \supset (f_0 ; f_1) \vee (f_0 ; f_2)$
EmptyChop	$\vdash$	$\mathrm{empty} ; f \equiv f$
ChopEmpty	$\vdash$	$f ; \mathrm{empty} \equiv f$
BiBoxChopImpChop	$\vdash$	$\boxed{\mathrm{i}}(f_0 \supset f_1) \wedge \Box(f_2 \supset f_3) \supset (f_0 ; f_2) \supset (f_1 ; f_3)$
StateImpBi	$\vdash$	$p \supset \boxed{\mathrm{i}}\, p$
NextImpNotNextNot	$\vdash$	$\bigcirc f \supset \neg \bigcirc \neg f$
BoxInduct	$\vdash$	$f \wedge \Box(f \supset \textcircled{w} f) \supset \Box f$
InfChop	$\vdash$	$(f \wedge \mathrm{inf}) ; g \equiv (f \wedge \mathrm{inf})$
ChopStarEqv	$\vdash$	$f^* \equiv (\mathrm{empty} \vee ((f \wedge \mathrm{more}) ; f^*))$
ChopstarInduct	$\vdash$	$(\mathrm{inf} \wedge f \wedge \Box(f \supset (g \wedge \mathrm{fmore}) ; f)) \supset g^*$
MP	$\vdash$	$f_0 \supset f_1, \ \vdash f_0 \ \Rightarrow \ \vdash f_1$
BoxGen	$\vdash$	$f_0 \ \Rightarrow \ \vdash \Box f_0$
BiGen	$\vdash$	$f_0 \ \Rightarrow \ \vdash \boxed{\mathrm{i}}\, f_0$

2.5 First order proof system

Some axioms for the first order case are shown in Table 16.
Let v refer to both static and state variables.
We denote by $f\ [e \ / \ v]$ that in formula f expression e is substituted for variable v.

Table 16: Some First Order Axioms and Rules for ITL.

ForallSub	$\vdash$	$\forall v \bullet f \supset f\ [e \ / \ v]$, where the expression e has the same data and temporal type as the variable v and is free for v in f.
ForallImplies	$\vdash$	$\forall v \bullet (f_1 \supset f_2) \supset (f_1 \supset \forall v \bullet f_2)$, where v doesn't occur freely in f_1.
SubstAxiom	$\vdash$	$\Box(v_1 = v_2) \supset f \equiv f\ [v_2 \ / \ v_1]$.
StaticWeakNext	$\vdash$	$w \supset \textcircled{w} w$, where w only contains static variables.
ExistsChopRight	$\vdash$	$\exists v \bullet (f_1 ; f_2) \supset (\exists v \bullet f_1) ; f_2$, where v doesn't occur freely in f_2.
ExistsChopLeft	$\vdash$	$\exists v \bullet (f_1 ; f_2) \supset f_1 ; (\exists v \bullet f_2)$, where v doesn't occur freely in f_1.
ForallGen	$\vdash$	$f \Rightarrow \ \vdash \forall v \bullet f$, for any variable v.

3 Tools

3.1 (Ana)Tempura

Tempura, the C-Tempura interpreter version 2.7 developed originally by Roger Hale and now maintained by Antonio Cau and Ben Moszkowski, is an interpreter for executable Interval Temporal Logic formulae. The first Tempura interpreter was programmed in Prolog by Ben Moszkowski, and was operational around December 2, 1983. Subsequently he rewrote the interpreter in Lisp (mid Mar, 1984), and in late 1984 modified the program to handle a two-level memory and multi-pass scanning. The C-Tempura interpreter was written in early 1985 by Roger Hale at Cambridge University.

AnaTempura, which is built upon C-Tempura, is a tool for the runtime verification of systems using Interval Temporal Logic (ITL) and its executable subset Tempura. The runtime verification technique uses assertion points to check whether a system satisfies timing, safety or security properties expressed in ITL. The assertion points are inserted in the source code of the system and will generate a sequence of information (system states), like values of variables and timestamps of value change, while the system is running. Since an ITL property corresponds to a set of sequences of states (intervals), runtime verification is just checking whether the sequence generated by the system is a member of the set of sequences corresponding to the property we want to check. The Tempura interpreter is used to do this membership test.

Download Version:

- Version 3.4 (released 19/12/2017) gzipped tar file or zip file.

```
- Initial support for monitoring rmi based Java programs.
- Use internal variable runid to assign id to external processes.
- Tempura commands stable and output now allows lists of variables,
  i.e., stable(V,W) and output(V,W).
- Use kitcreator to generate the anatempura binaries. The windows
  binaries are generated using the mingw-w64 cross-compiler.
- Various other bug fixes, see ChangeLog for more details.
```

- Version 3.3 (released 07/06/2016) gzipped tar file or zip file.

```
- The Tcl/Tk graphical user interface does not depend on Expect anymore.
- The lexer/parser now throws an error on encountering a unknown
  character instead of silently discarding it.
- The read-only State_number variable can be used in Tempura programs
  to determine the current state number.
- The monitoring of C# programs does not need a Java wrapper anymore.
- Support for monitoring of Java programs in the form of a jar file.
- Support of time-stamps in the form of seconds and microseconds.
- The GUI has now a different look/layout, history of commands window is
  gone, the most commonly used menu entries are now buttons and one interacts
  with Tempura using a shell-like interface.
- Added template .anatempurarc .
- Integers are now 64bits regardless of the machine architecture.
- Windows binaries are compiled using the msys2-mingw32/64 system.
```

```
- For Windows we have 32 bit (XP and beyond) and 64 bit (7 and beyond)
  Tempura/AnaTempura binaries.
- Random numbers are now generated using xorshift128plus.
- Added support for monitoring Scala programs.
- Added option -stdio to start anatempura in cmdline mode.
- Various other bug fixes, see ChangeLog for more details.
```

- Version 3.2 (released 30/11/2015) gzipped tar file or zip file.

```
- Rewritten the Tempura output capture/processing routine.
- Rewritten the external program data capture/processing routine.
- New implementation of memory/framed variables, existsf V : f, is now used
  to indicate that V is memory/framed variable within f.
  The previous syntax mem(V) is now deprecated.
- Fix bugs in the implementation of prev(L) where L is a list.
- Added Tempura binary for Arduino-Yun and Raspberry Pi, i.e.,
  tempura_mips_openwrt_linux and tempura_arm_linux_gnueabihf.
- Added elapsed time in the statistics output at the end of a run.
- Added plc and sql injection detection examples.
- Various other bug fixes see ChangeLog for more details.
```

- Version 3.1 (released 22/05/2015) gzipped tar file or zip file.

```
- Changed contact email address.
- Added pre-compiled MacOSX binary.
- Added frame(V) as alias for mem(V)
- Fixed some bugs in implementation of mem(V).
- Fixed some bugs in the help command.
```

- Version 3.0 (released 04/07/2013) gzipped tar file or zip file.

```
- Dropped pre-compiled Solaris binary.
- flex/bison based parser backward compatible with previous hard coded
  version but with stricter syntactic checks
- changed from cvs to svn as version control system
- improved syntax error messages
- fixed memory leaks
- tidy up format command
- startup file .anatempurarc can also be in the current directory
- use kbs-0.4.4 to generate anatempura binaries
- anatempura binaries are using Tcl/Tk 8.6
- works again under Windows XP
- program assertions can have any symbol except control characters and !
```

- Version 2.18 (released 01/11/2012): gzipped tar file or zip file.

```
- Dropped tempura_macosx binary but added tempura_linux64 and
  anatempura_linux64 binaries.
- fixed some small bugs
- fixed memory leaks
- added command ‘winput’ that will wait for input from a file instead
  of switching to the keyboard.
```

- Version 2.17 (released 04/10/2011): gzipped tar file or zip file.

```
- initial support for MACOSX
- fixed gui bugs
- fixed some tempura bugs
```

- Version 2.16 (released 08/12/2009): gzipped tar file or zip file.

```
- added floats. Floats have the form $2.3e+10$ in Tempura. For output:
  output($2.3$) will be $2.30000e+00$ , i.e., precision is 5 digits
  after the '.'.  One can set this via the precision variable. With
  precision of 2 one gets $2.30e+00$. The format command can output
  floats in two forms: %f output will be of the form 2.33333, e output
  will be of the form 2.33333e+01. The following operations on floats
  are defined: unary, +, -; binary: +, -, div, mod, /, *, **, ceil,
  floor, sqrt, itof, exp, log, log10, sin, cos, tan, asin, acos, atan,
  atan2, sinh, cosh, tanh, fabs.
- anatempura is now using the new Tile interface
- when setting system variables with set, output both old and new
  values
- Added 'frandom' and 'fRandom' for float random number between
  [0.0,1.0)
- Added defaults command, X defaults 1 denotes when X is undefined
  then take as value for X the value 1.
- Added prev(X) operator, the value of X in the previous state.
- Added mem(X) operator, X is a 'memory' variable, i.e., when
  undefined take the value in the previous state.
- Added #n history operator, used as option to exists when declaring a
  variable, it will keep a history of n previous values of a variable.
- Added nprev(X,n) operator, nprev(X,3) for instance is an
  abbreviation of prev(prev(prev(X))).
- When setting debug_level to 6 more usefull information is displayed
  like the state of a variable and reduction rule being applied.
- Included tempura executables tempura_linux for Linux (compiled on
  Ubuntu 9.10),  tempura_solaris for Solaris (compiled on Sparc
  Solaris 10u8), and tempura.exe for Windows (compiled on Windows XP SP3).
- Included anatempura executables anatempura_solaris, anatempura_linux
  and anatempura.exe. These were built using the Tclkit Kitgen build
  system (http://wiki.tcl.tk/18146). Now no need anymore to install
  tcl/tk and expect in order to run anatempura.
- changed copyright license to GPLv3.0
```

- Version 2.15 (released 14/08/2008): gzipped tar file or zip file.

```
*******************2.15****************************************************
- introduced various node accessor macros so that if one changes the node
  structure we only have to change the macro.
- if formula can't be reduced in the final state of the
  prefix of a chop then we will evaluate ((prefix and empty);true)
  and (suffix). This feature can be switched on/off with hopchop.
  The default of hopchop is true.
- added integer overflow tests.
- unified/cleaned up the various node data structures.
```

- Version 2.14 (released 29/11/2007): gzipped tar file or zip file.

```
*******************2.14****************************************************
- work around a recent misfeature of windows when started an external program.
- added the io redirections, set infile="some file name", set
  outfile="some file name", where stdin and stdout can be used to
  redirect to standard keyboard and screen i/o.
- added the infinite and randlen constructs for respectively an
  infinite interval and a random length interval (less or equal
  to max_randlen).
```

- Version 2.13 (released 28/08/2007): gzipped tar file or zip file.

```
*******************2.13****************************************************
- added reset in file menu to restart tempura.
- open and reload now also load the file into Tempura.
- added showstate Tempura command. This will display what is
  (un)defined in the current state.
- changed contact email address to tempura@dmu.ac.uk
```

- Version 2.12 (released 04/05/2007): gzipped tar file or zip file.

```
******************2.12****************************************************
This version is the first version that compiles both under
Windows and Unix/Linux type of machines. See Changelog for detailed
news/changes.
```

How to run AnaTempura?

- Use the pre-compiled binary:

  anatempura.exe: 32 bit binary and will run on Windows Xp, 7 and beyond, this will use tempura.exe in the current directory

  anatempura-win64.exe: 64 bit binary and will run on Windows 7 and beyond, this will use tempura-win64.exe in the current directory

  anatempura_linux: For 32bit Linux OS, this will use tempura_linux in the current directory

  anatempura_linux64: For 64bit Linux OS, this will use tempura_linux64 in the current directory

  anatempura_macosx: For MacOSX, this will use tempura_macosx in the current directory

- Use anatempura.tcl:

  you need to install Tempura, and Tcl/Tk (at least 8.5). Get Tcl/Tk and from Tcl/Tk site or use the ActiveTcl package.

  Tempura can be compiled using the Gnu C compiler under a Unix like operating system like Ubuntu, GNU Debian, etc. For Windows you can use the MSYS2/MinGW-32/64 system, see their website for downloading and installation. Install the MSYS2 system, and then, using pacman, install the mingw-w64-i686-toolchain for compiling 32bit binaries and the mingw-w64-x86_64-toolchain for compiling 64bit binaries.

  Compile Tempura using the following commands

  ```
  ./configure
  make
  ```

  For convenience the following pre-compiled Tempura binaries are included in the distribution:

32 bit Windows binary	tempura.exe,
64 bit Windows binary	tempura-win64.exe,
32 bit Linux binary	tempura_linux,
64 bit Linux binary	tempura_linux64,
MacOSX binary	tempura_macosx,
Arduino-Yun binary	tempura_mips_openwrt_linux,
Raspberry Pi binary	tempura_arm_linux_gnueabihf.

Contact: Email cau.researcher@gmail.com in case of problems.

Publications:

- Analysing C programs is discussed in:
  A Framework For Analysing The Effect of 'Change' In Legacy Code, S. Zhou, H. Zedan and A. Cau. In IEEE Proc. of ICSM'99, 1999.

- Analysing Verilog programs is discussed in:
  A logic-based Approach for Hardware/Software Co-design, H. Zedan and A. Cau. Digest of IEE event Hardware-Software Co-design, 8 Dec., 2000.

- A paper describing the run-time verification method used in AnaTempura:
  Run-time analysis of time-critical systems. S. Zhou and H. Zedan and A. Cau. Journal of System Architecture, 51(5):331-345, 2005.

- Slides of seminar talk about AnaTempura:
  AnaTempura, A. Cau, S. Zhou and H. Zedan.

Overview: Figure 1 shows an overview of AnaTempura.
Figure 2 shows the interface of AnaTempura.
Figure 3 shows a graphical snapshot of a simulation of the EP/3 microprocessor specified in Tempura.

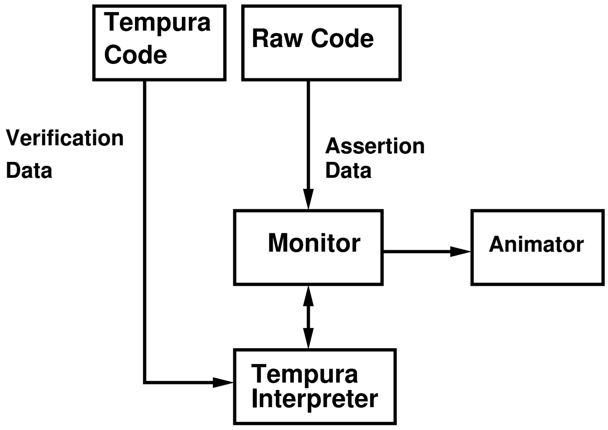

Figure 1: Overview of AnaTempura

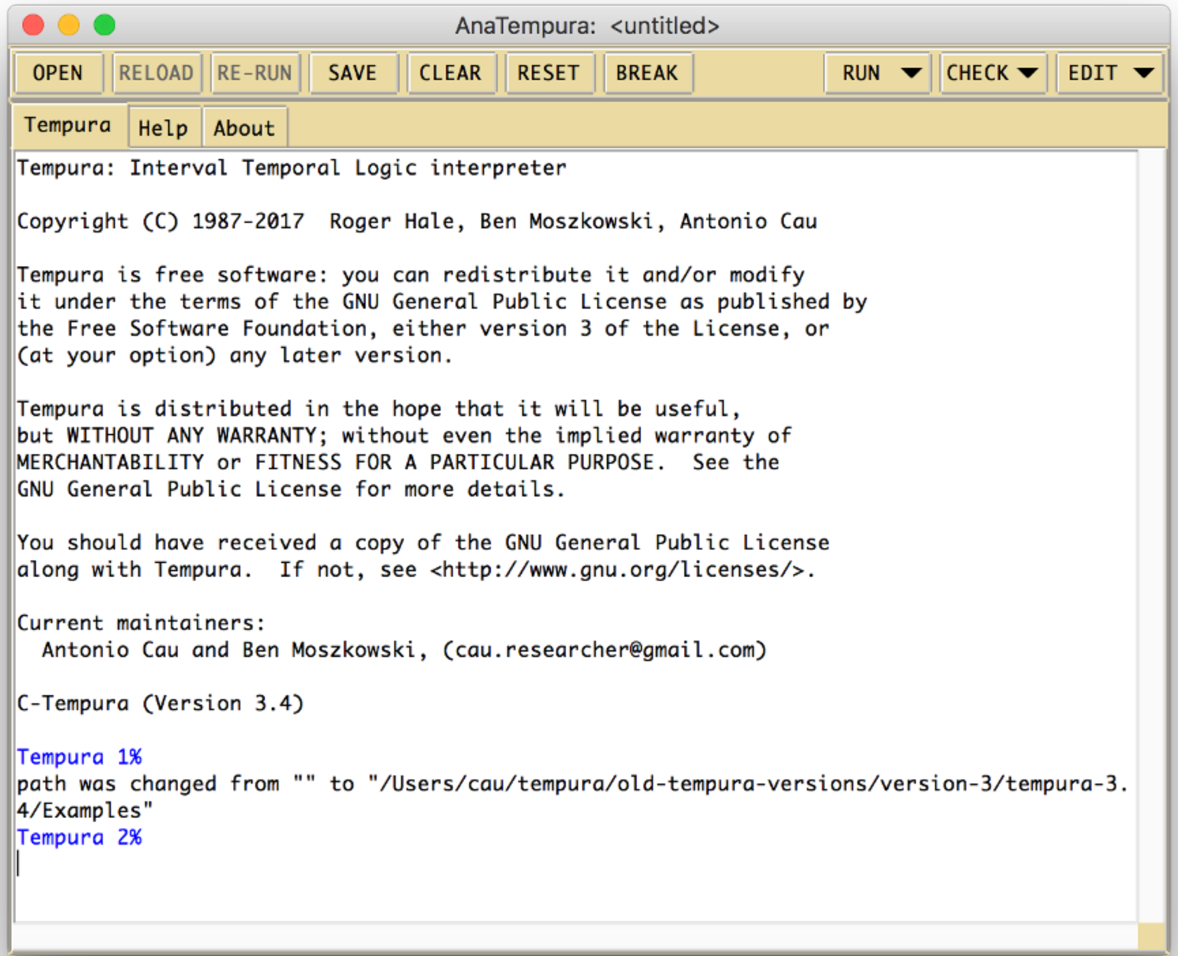

Figure 2: Interface of AnaTempura

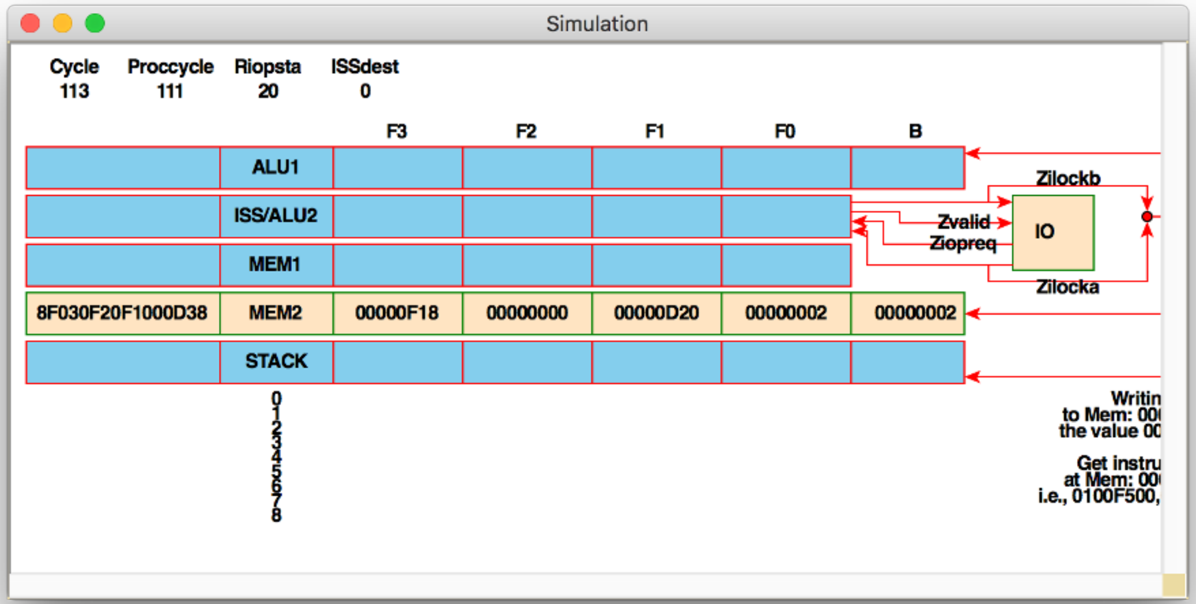

Figure 3: Graphical snapshot of simulation of the EP/3 microprocessor

3.2 FLCheck: Fusion Logic decision Procedure

Fusion Logic augments conventional Propositional Temporal Logic (PTL) with the fusion operator. Note: the fusion-operator is basically a "chop" that does not have an explicit negation on the left hand (for right fusion logic) side (as fusion expression) or the right hand (for left fusion logic). The negation is implicit, i.e., the negation is a derived fusion expression operator. The expressiveness of Fusion Logic is the same as Propositional Interval Temporal Logic. The main differences concern computational complexity, naturalness of expression for analytical purposes, and succinctness. Fusion Logic is closely related to Propositional Dynamic Logic (PDL).

We have implemented above decision procedure for Fusion Logic in Tcl/Tk and the CUDD BDD library. The tool allows one to check the validity or satisfiability of a Fusion Logic formula. If a formula is not valid it will produce a counter example and if a formula is satisfiable it will produce an example model. Figure 4 gives a screen dump of our tool which is available at

- FLCHECK version 1.2 (released 22/05/2015).

  Main Changes:

  - Added support for MacOSX.
  - Added pre-compiled binary for MacOSX.

- FLCHECK version 1.1 (released 04/07/2013).

  Main Changes:

  - Drop support for Solaris Sparc.
  - Added pre-compiled binaries for Windows (XP and 7) and Linux (Ubuntu 12.04, 32 and 64 bits).

- FLCHECK version 1.0 (released 23/01/2013).

  Main Changes:

  - Simplified 'left always followed by' operator.
  - Added more examples.

- FLCHECK version 0.9 (released 21/02/2012).

  Main Changes:

  - Introduction of left and right Fusion Logic which makes the specification of access control policies much simpler
  - Use of time reversal to rewrite left fusion logic formulae into right fusion logic formulae
  - Enforcement of policies expressed in left Fusion Logic
  - All examples come now with comments

- FLCHECK version 0.8 (released 26/03/2010).

  Initial release.

Publications:

- Antonio Cau, Helge Janicke, and Ben Moszkowski. Verification and enforcement of access control policies. Formal Methods in System Design, Springer, 2013.
- A Note on Expressing Policy Rules in Fusion Logic, A. Cau. Technical report, STRL, De Montfort University.

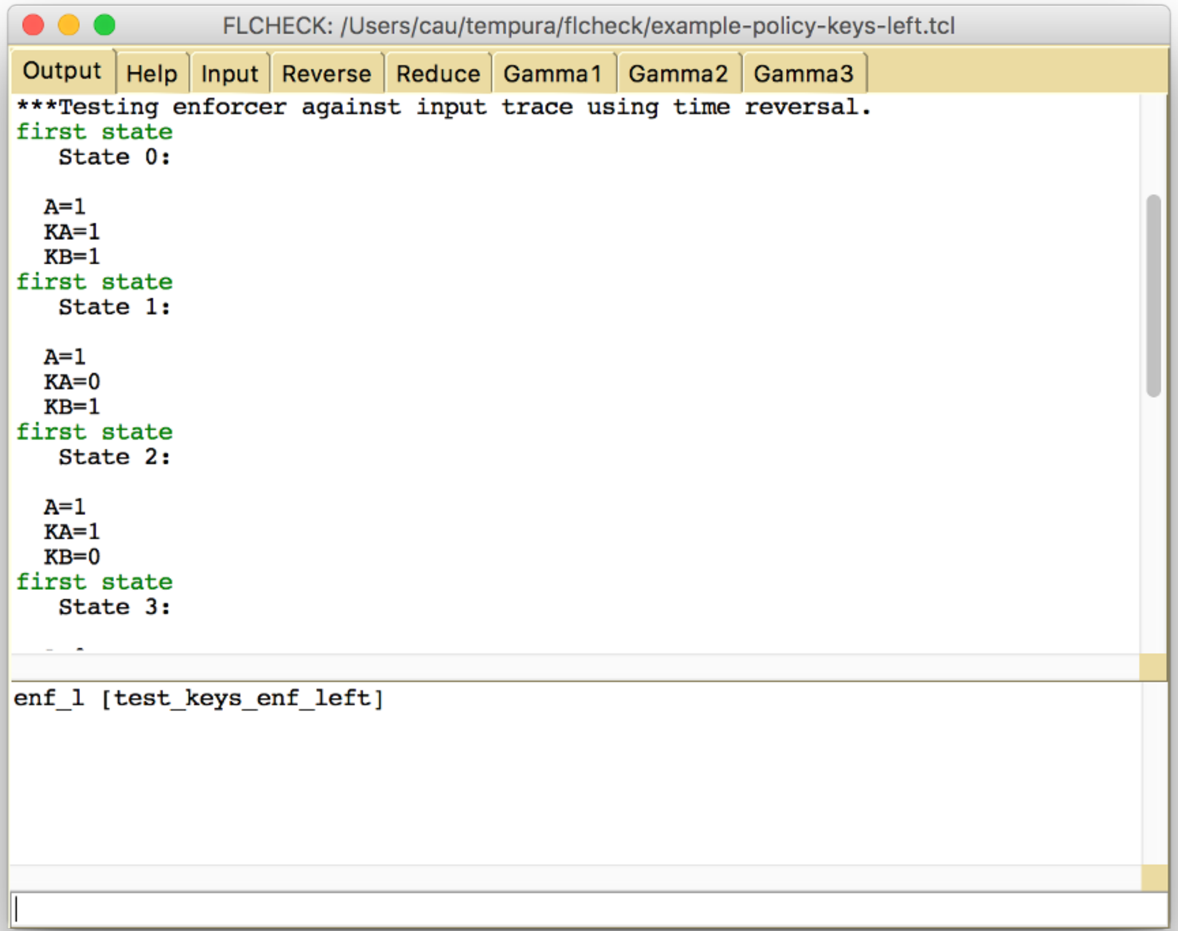

Figure 4: FLCHECK fusion logic decision procedure

3.3 ITL library for Isabelle/HOL

Isabelle/HOL is a generic proof assistant. It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus.

We have given a deep embedding of propositional ITL and a shallow embedding of first order ITL in Isabelle/HOL. A shallow embedding represents ITL using Isabelle/HOL predicates, while a *deep* embedding would represent ITL formulas as mutually inductive datatypes. See, e.g., [96] for a discussion about deep vs. shallow embeddings in Isabelle/HOL. The shallow embedding of first order ITL uses techniques developed by Stefan Merz [175, 20] for the shallow embedding of Temporal Logic of Actions (TLA) in Isabelle/HOL.

- *Deep embedding*: version 1.8 (08/12/2018, first public release), gzipped tar file.

  Contains:

  - Propositional ITL, syntax, semantics (finite intervals), and axioms and proof rules.
  - The lemmas from Imperative Reasoning in Interval Temporal Logic by Ben Moszkowski.
  - Time Reversal operator and an extensive list of lemmas.
  - First operator and Monitors. All key theorems/lemmas/semantics in David Smallwood PhD thesis "ITL Monitor: Compositional Runtime Analysis with Interval Temporal Logic" have been verified.
  - Link with Georg Struth's work on Kleene Algebra. This work is based on Interval Temporal Algebra (Section 3.4) but now re-encoded in Isabelle/HOL. So the work on ITL and Prover9 and PVS has now been superseded by the Isabelle/HOL library.

- *Shallow embedding*: version 2.2 (08/12/2018, warning this is work in progress), gzipped tar file.

  Contains:

  - First Order ITL, both quantification over state and static variables, added temporal variables (current, next, final, penultimate), syntax, semantics, and axioms and proof rules. The technique used is similar to that of Stephan Merz [175, 20] used for encoding TLA in Isabelle/HOL.
  - The lemmas from Imperative Reasoning in Interval Temporal Logic by Ben Moszkowski.
  - Time Reversal operator and an extensive list of lemmas included. Time reversal operator also works for temporal variables and quantification.
  - First operator and Monitors. All key theorems/lemmas/semantics in David Smallwood PhD thesis "ITL Monitor: Compositional Runtime Analysis with Interval Temporal Logic" have been verified.
  - Some concrete monitors have been specified and properties verified.
  - Some examples using quantification.

3.4 ITL Theorem Prover based on Prover9

Note: this work has been re-encoded in the deep embedding of propositional ITL in Isabelle/HOL.

Prover9 is a resolution/paramodulation automated theorem prover for first-order and equational logic developed by William McCune.

We have given an algebraic axiom system for Propositional Interval Temporal Logic (PITL): Interval Temporal Algebra. The axiom system is a combination of a variant of Kleene algebra and Omega algebra plus axioms for linearity and confluence.

This algebraic axiom system for PITL has been encoded in Prover9. So we can use Prover9 to prove the validity of various PITL theorems. The Prover9 encoding of PITL plus examples of more than 300 PITL theorems are available for download as

- Version 1.8 (released 27/08/2009): gzipped tar file.

```
- documentation updated to new semantics for
chopstar and chop omega algebraic operators
```

- Version 1.7 (released 15/05/2009): gzipped tar file.

```
- updated documentation in doc, to use new ITL semantics
```

- Version 1.6 (released 12/12/2008): gzipped tar file.

```
- changed copyright license to GPLv3.0 and added the notice to all files
```

- Version 1.5 (first public release: 05/12/2008): gzipped tar file.

The README in this tar file contains instructions how to use Prover9 for proving PITL theorems.

3.5 ITL Proof Checker based on PVS

Note: this work has been superseded by the deep and shallow embedding of ITL in Isabelle/HOL.

PVS is an interactive environment, developed at SRI, for writing formal specifications and checking formal proofs. The specification language used in PVS is a strongly typed higher order logic. The powerful interactive theorem prover/proof checker of PVS has a large set of basic deductive steps and the facility to combine these steps into proof strategies. PVS is implemented in Common Lisp –with ancillary functions provided in C, Tcl/TK and LaTeX– and uses GNU Emacs for its interface. PVS is freely available for IBM RS6000 machines as well as Sun Sparcs under license from SRI. See PVS homepage for more information.

- The ITL library for PVS 4.0.
- The ITL library for PVS 3.2.
- The ITL library for PVS 2.4 patchlevel 1.
- The ITL library for PVS 2.3.
- The ITL library for PVS 2.2.
- The ITL library for PVS 2.1 patchlevel 2.417.

Publications:

- Technical report.

3.6 Automatic Verification of Interval Temporal Logic

Shinji Kono has developed an automatic theorem prover for propositional ITL (LITE). The implementation is in Prolog. Further information can be gathered at Shinji Kono's Interval Temporal Logic page. Shinji Kono has also a Java version of LITE see CVS repository of JavaLite.

4 ITL Related Publications

Download BibTeX file

The book Executing Temporal Logic Programs by Dr. B. C. Moszkowski was originally published by Cambridge University Press in 1986. The publishers have kindly given the copyright back to the author. The pdf version of the book has now been made available.

4.1 Articles

[1] Ben Moszkowski and Dimitar Guelev. "An application of temporal projection to interleaving concurrency". In: *Formal Aspects of Computing* 29.4 (July 2017), pp. 705–750.
URL: http://dx.doi.org/10.1007/s00165-017-0417-3.

[2] Brijesh Dongol, Ian J. Hayes, and Georg Struth. "Convolution As a Unifying Concept: Applications in Separation Logic, Interval Calculi, and Concurrency". In: *ACM Trans. Comput. Logic* 17.3 (Feb. 2016), 15:1–15:25. ISSN: 1529-3785.
URL: http://doi.acm.org/10.1145/2874773.

[3] Stéphane Demri and Morgan Deters. "Two-Variable Separation Logic and Its Inner Circle". In: *ACM Trans. Comput. Logic* 16.2 (Apr. 2015), 15:1–15:36. ISSN: 1529-3785.
URL: http://dx.doi.org/10.1145/2724711.

[4] Helge Janicke, Andrew Nicholson, Stuart Webber, and Antonio Cau. "Runtime-Monitoring for Industrial Control Systems". In: *Electronics* 4.4 (Dec. 2015). Ed. by Dhananjay Phatak. Open Access, pp. 995–1017.
URL: http://dx.doi.org/10.3390/electronics4040995.

[5] Qian Ma, Zhenhua Duan, Nan Zhang, and Xiaobing Wang. "Verification of distributed systems with the axiomatic system of MSVL". In: *Formal Aspects of Computing* 27.1 (2015), pp. 103–131. ISSN: 0934-5043.
URL: http://dx.doi.org/10.1007/s00165-014-0303-1.

[6] Ben Moszkowski. "Compositional reasoning using intervals and time reversal". In: *Annals of Mathematics and Artificial Intelligence* 71.1-3 (2014), pp. 175–250. ISSN: 1012-2443.
URL: http://dx.doi.org/10.1007/s10472-013-9356-8.

[7] Ben Moszkowski, Dimitar Guelev, and Martin Leucker. "Guest editors' preface to special issue on interval temporal logics". In: *Ann. Math. Artif. Intell.* 71.1-3 (2014), pp. 1–9.
URL: http://dx.doi.org/10.1007/s10472-014-9417-7.

[8] Gerhard Schellhorn, Bogdan Tofan, Gidon Ernst, Jörg Pfähler, and Wolfgang Reif. "RGITL: A temporal logic framework for compositional reasoning about interleaved programs". In: *Annals of Mathematics and Artificial Intelligence* 71.1-3 (2014), pp. 131–174. ISSN: 1012-2443.
URL: http://dx.doi.org/10.1007/s10472-013-9389-z.

[9] Bogdan Tofan, Oleg Travkin, Gerhard Schellhorn, and Heike Wehrheim. "Two approaches for proving linearizability of multiset". In: *Science of Computer Programming* 96, Part 3.0 (2014), pp. 297–314. ISSN: 0167-6423.
URL: http://dx.doi.org/10.1016/j.scico.2014.04.001.

[10] Antonio Cau, Helge Janicke, and Ben Moszkowski. "Verification and enforcement of access control policies". In: *Formal Methods in System Design* 43.3 (2013). Download pdf, pp. 450–492. ISSN: 0925-9856.
URL: http://dx.doi.org/10.1007/s10703-013-0187-3.

[11] Zhenhua Duan, Hanna Klaudel, and Maciej Koutny. "ITL semantics of composite Petri nets". In: *The Journal of Logic and Algebraic Programming* 82.2 (2013), pp. 95–110. ISSN: 1567-8326.
URL: http://dx.doi.org/10.1016/j.jlap.2012.12.001.

[12] Zhenhua Duan, Nan Zhang, and Maciej Koutny. "A complete proof system for propositional projection temporal logic". In: *Theoretical Computer Science* 497.0 (2013), pp. 84–107. ISSN: 0304-3975.
URL: http://dx.doi.org/10.1016/j.tcs.2012.01.026.

[13] Helge Janicke, Antonio Cau, François Siewe, and Hussein Zedan. "Dynamic Access Control Policies: Specification and Verification". In: *The Computer Journal* 56.4 (2013). Download pdf, pp. 440–463.
URL: http://dx.doi.org/10.1093/comjnl/bxs102.

[14] Ben Moszkowski. "Interconnections between classes of sequentially compositional temporal formulas". In: *Information Processing Letters* 113.9 (2013), pp. 350–353. ISSN: 0020-0190.
URL: http://dx.doi.org/10.1016/j.ipl.2013.02.005.

[15] Nan Zhang, Zhenhua Duan, and Cong Tian. "A cylinder computation model for many-core parallel computing". In: *Theoretical Computer Science* 497.0 (2013), pp. 68–83. ISSN: 0304-3975.
URL: http://dx.doi.org/10.1016/j.tcs.2012.02.011.

[16] Sulaiman Al Amro and Antonio Cau. "Behavioural API based Virus Analysis and Detection". In: *International Journal of Computer Science and Information Security* 10.5 (May 2012). Download pdf, pp. 14–22.
URL: https://sites.google.com/site/ijcsis/vol-10-no-5-may-2012.

[17] Ben C. Moszkowski. "A Complete Axiom System for Propositional Interval Temporal Logic with Infinite Time". In: *Logical Methods in Computer Science Journal* 8.3 (2012).
URL: http://dx.doi.org/10.2168/LMCS-8(3:10)2012.

[18] Turki Alghamdi, Hussein Zedan, and Ali Alzahrani. "Enforcing Learning Activities Policies in Runtime Monitoring System for E-learning Environments". In: *International Journal of Computer Science and Information Security (IJCSIS)* 9.8 (2011), pp. 45–53.
URL: https://sites.google.com/site/ijcsis/vol-9-no-8-aug-2011.

[19] Simon Bäumler, Gerhard Schellhorn, Bogdan Tofan, and Wolfgang Reif. "Proving linearizability with temporal logic". In: *Formal Aspects of Computing* 23.1 (2011), pp. 91–112. ISSN: 0934-5043.
URL: http://dx.doi.org/10.1007/s00165-009-0130-y.

[20] Gudmund Grov and Stephan Merz. "A Definitional Encoding of TLA* in Isabelle/HOL". In: *Archive of Formal Proofs* (2011). https://www.isa-afp.org/entries/TLA.html, Formal proof development. ISSN: 2150-914x.

[21] Alan Burns and Ian J. Hayes. "A timeband framework for modelling real-time systems". In: *Real-Time Systems* 45.1-2 (2010), pp. 106–142. ISSN: 0922-6443.
URL: http://dx.doi.org/10.1007/s11241-010-9094-5.

[22] Ben Moszkowski. "Using Temporal Logic to Analyse Temporal Logic: A Hierarchical Approach Based on Intervals". In: *Journal of Logic and Computation* 17.2 (2007). Download pdf, pp. 333–409.
URL: http://dx.doi.org/10.1093/logcom/exm006.

[23] Dimitar P. Guelev and Dang Van Hung. "On the Completeness and Decidability of Duration Calculus with Iteration". In: *Theoretical Computer Science* 337 (2005), pp. 278–304.
URL: http://dx.doi.org/10.1016/j.tcs.2005.01.017.

[24] Shikun Zhou, Hussein Zedan, and Antonio Cau. "Run-time Analysis of Time-critical Systems". In: *Journal of System Architecture* 51.5 (2005). Download pdf, pp. 331–345.
URL: http://dx.doi.org/10.1016/j.sysarc.2004.12.003.

[25] Zhen-Hua Duan and Maciej Koutny. "A framed temporal logic programming language". In: *Journal of Computer Science and Technology* 19.3 (2004), pp. 341–351. ISSN: 1000-9000.
URL: http://dx.doi.org/10.1007/BF02944904.

[26] Rodolfo Gomez and Howard Bowman. "PITL2MONA: Implementing a Decision Procedure for Propositional Interval Temporal Logic". In: *Journal of Applied Non-Classical Logics* 14.1–2 (2004), pp. 105–148.
URL: http://dx.doi.org/10.3166/jancl.14.105-148.

[27] Wang Hanpin and Xu Qiwen. "Completeness of temporal logics over infinite intervals". In: *Discrete Applied Mathematics* 136.1 (2004). Discrete Mathematics and Theoretical Computer Science (DMTCS), pp. 87–103. ISSN: 0166-218X.
URL: http://dx.doi.org/10.1016/S0166-218X(03)00201-4.

[28] Ben Moszkowski. "A Hierarchical Completeness Proof for Propositional Interval Temporal Logic with Finite Time". In: *Journal of Applied Non-Classical Logics* 14.1–2 (2004), pp. 55–104.
URL: http://dx.doi.org/10.3166/jancl.14.55-104.

[29] Howard Bowman and Simon J. Thompson. "A Decision Procedure and Complete Axiomatization of Finite Interval Temporal Logic with Projection". In: *Journal of Logic and Computation* 13.2 (Apr. 2003), pp. 195–239.
URL: http://dx.doi.org/10.1093/logcom/13.2.195.

[30] Antonio Cau et al. "A Compositional Framework for Hardware/Software Co-Design". In: *Design Automation for Embedded Systems* 6.4 (2002). Ed. by Raul Camposano, Wayne Wolf, and Bashir Al-Hashimi. Download pdf, pp. 367–399. ISSN: 0929-5585.
URL: http://dx.doi.org/10.1023/A:1016507527035.

[31] R. Mattolini and P. Nesi. "An Interval Logic for Real-Time System Specification". In: *Transactions on Software Engineering, IEEE* 27.3 (Mar. 2001), pp. 208–227.
URL: http://dx.doi.org/10.1109/32.910858.

[32] Hongji Yang, Xiaodong Liu, and Hussein Zedan. "Abstraction: A Key Notion for Reverse Engineering in A Reengineering Approach". In: *Journal of Software Maintenance: Research and Practice* 12.4 (2000), pp. 197–228.
URL: http://dx.doi.org/10.1002/1096-908X(200007/08)12:4%3C197::AID-SMR211%3E3.0.CO;2-X.

[33] Zhiqiang Chen, Hussein Zedan, Antonio Cau, and Hongji Yang. "A Wide-Spectrum Language for Object-Based Development of Real-time Systems". In: *Journal of Information Sciences* 118 (1999). Download pdf, pp. 15–35.
URL: http://dx.doi.org/10.1016/S0020-0255(99)00039-0.

[34] Hussein Zedan, Antonio Cau, Zhiqiang Chen, and Hongji Yang. "ATOM: An Object-based Formal Method for Real-time Systems". In: *Annals of Software Engineering* 7 (1999). Download pdf, pp. 235–256.
URL: http://dx.doi.org/10.1023/A:1018942406449.

[35] Ben Moszkowski. "Executable Temporal Logic Systems: The programming language Tempura". In: *Journal of Symbolic Computation* 22.5-6 (1996). Download pdf, pp. 730–733. ISSN: 0747-7171.
URL: http://dx.doi.org/10.1006/jsco.1996.0073.

[36] Y. Srinivas Ramakrishna, Peter M. Melliar-Smith, Louise E. Moser, Laura K. Dillon, and George Kutty. "Interval logics and their decision procedures. Part II: A real-time interval logic". In: *Theoretical Computer Science* 170.1–2 (Dec. 1996), pp. 1–46.
URL: http://dx.doi.org/10.1016/S0304-3975(96)80701-8.

[37] Y. Srivinas Ramakrishna, Peter M. Melliar-Smith, Louise E. Moser, Laura K. Dillon, and George Kutty. "Interval logics and their decision procedures. Part I: An interval logic". In: *Theoretical Computer Science* 166.1–2 (Oct. 1996), pp. 1–47.
URL: http://dx.doi.org/10.1016/0304-3975(95)00254-5.

[38] Laura K. Dillon, George Kutty, Louise E. Moser, Peter M. Melliar-Smith, and Y. Srinivas Ramakrishna. "A Graphical Interval Logic for Specifying Concurrent Systems". In: *ACM Transactions on Software Engineering and Methodology* 3.2 (Apr. 1994), pp. 131–165.
URL: http://dx.doi.org/10.1145/192218.192226.

[39] R. Dowsing, E. Elliot, and I. Marshall. "Automated technique for high-level circuit synthesis from temporal logic specifications". In: *IEE Proceedings–Computers and Digital Techniques* 141.3 (May 1994), pp. 145–152.
URL: http://dx.doi.org/10.1049/ip-cdt:19941005.

[40] Michael R. Hansen. "Model-Checking Discrete Duration Calculus". In: *Formal Aspects of Computing* 6.1 (1994), pp. 826–845.
URL: http://dx.doi.org/10.1007/BF01213605.

[41] Masahiro Fujita and Shinji Kono. "Synthesis of Controllers from Interval Temporal Logic Specification". In: *International Workshop on Logic Synthesis* (1993).

[42] Kiyoharu Hamaguchi, Hiromi Hiraishi, and Shuzo Yajima. "Infinity-Regular Temporal Logic and its Model Checking Problem". In: *Theor. Comput. Sci.* 103.2 (1992), pp. 191–204.
URL: http://dx.doi.org/10.1016/0304-3975(92)90012-5.

[43] Hiromi Hiraishi, Kiyoharu Hamaguchi, Hiroshi Fujii, and Shuzo Yajima. "Regular Temporal Logic Expressively Equivalent to Finite Automata and its Application to Logic Design Verification". In: *Journal of Information Processing* 15.1 (1992), pp. 129–138.
URL: http://ci.nii.ac.jp/naid/110002673621/en/.

[44] Antonio Nunez, Don Fay, R.D. Dowsing, and R. Elliott. "A higher level of behavioural specification: An example in Interval Temporal Logic". In: *Microprocessing and Microprogramming* 32.1 (1991), pp. 517–524. ISSN: 0165-6074.
URL: http://dx.doi.org/10.1016/0165-6074(91)90395-A.

[45] Chaochen Zhou, Charles Antony Richard Hoare, and Anders P. Ravn. "A Calculus of Durations". In: *Information Processing Letters* 40.5 (1991), pp. 269–276.
URL: http://dx.doi.org/10.1016/0020-0190(91)90122-X.

[46] Miriam Leeser. "Reasoning About the Function and Timing of Integrated Circuits with Interval Temporal Logic". In: *IEEE Transactions on Computer-Aided Design* 8.12 (Dec. 1989), pp. 1233–1246.
URL: http://dx.doi.org/10.1109/43.44505.

[47] Masahiro Fujita, Shinji Kono, Hidehiko Tanaka, and Tatsuhiko Moto-Oka. "Aid to hierarchial and structured logic design using temporal logic and Prolog". In: *IEE Proceedings E (Computers and Digital Techniques)* 133 (5 Sept. 1986), pp. 283–294. ISSN: 0143-7062.
URL: http://dx.doi.org/10.1049/ip-e.1986.0035.

[48] Ben Moszkowski. "A Temporal Logic for Multilevel Reasoning about Hardware". In: *IEEE Computer* 18.2 (1985), pp. 10–19. ISSN: 0018-9162.
URL: http://dx.doi.org/10.1109/MC.1985.1662795.

4.2 In Collections

[49] Vladimir Valkanov et al. "AjTempura – First Software Prototype of C3A Model". In: *Intelligent Systems'2014: Proceedings of the 7th IEEE International Conference Intelligent Systems IS'2014, September 24-26, 2014, Warsaw, Poland, Volume 1: Mathematical Foundations, Theory, Analyses*. Ed. by P. Angelov et al. Cham: Springer International Publishing, 2015, pp. 427–435. ISBN: 978-3-319-11313-5.
URL: https://doi.org/10.1007/978-3-319-11313-5_38.

[50] Hanna Klaudel, Maciej Koutny, and Zhenhua Duan. "Interval Temporal Logic Semantics of Box Algebra". In: *Language and Automata Theory and Applications*. Ed. by Adrian-Horia Dediu, Carlos Martín-Vide, José-Luis Sierra-Rodríguez, and Bianca Truthe. Vol. 8370. Lecture Notes in Computer Science. Springer Verlag, 2014, pp. 441–452. ISBN: 978-3-319-04920-5.
URL: http://dx.doi.org/10.1007/978-3-319-04921-2_36.

[51] Xu Lu, Zhenhua Duan, Cong Tian, and Hongjin Liu. "Integrating Separation Logic with PPTL". In: *Structured Object-Oriented Formal Language and Method*. Ed. by Shaoying Liu and Zhenhua Duan. Lecture Notes in Computer Science. Springer Verlag, 2014, pp. 35–47. ISBN: 978-3-319-04914-4.
URL: http://dx.doi.org/10.1007/978-3-319-04915-1_3.

[52] Ajesh Babu and Paritosh K. Pandya. "Chop Expressions and Discrete Duration Calculus". In: *Modern Applications of Automata Theory*. World Scientific Publishing, 2012, pp. 229–256.
URL: http://dx.doi.org/10.1142/9789814271059_0008.

[53] Kamal Lodaya. "A Language-Theoretic View of Verification". In: *Modern Applications of Automata Theory*. World Scientific Publishing, 2012, pp. 149–170.
URL: http://dx.doi.org/10.1142/9789814271059_0005.

[54] Markus Holzer and Sebastian Jakobi. "Chop Operations and Expressions: Descriptional Complexity Considerations". In: *Developments in Language Theory*. Ed. by Giancarlo Mauri and Alberto Leporati. Vol. 6795. Lecture Notes in Computer Science. Springer Verlag, 2011, pp. 264–275.
URL: http://dx.doi.org/10.1007/978-3-642-22321-1_23.

[55] Monika Solanki, Antonio Cau, and Hussein Zedan. "Temporal Reasoning Of Reactive Web Services". In: *Semantic Web Services, Processes and Applications*. Ed. by Jorge Cardoso and AmitP. Sheth. Vol. 3. Semantic Web and Beyond. Springer US, 2006, pp. 107–136. ISBN: 978-0-387-30239-3.
URL: http://dx.doi.org/10.1007/978-0-387-34685-4_5.

[56] Ben Moszkowski. "A Hierarchical Analysis of Propositional Temporal Logic based on Intervals". In: *We Will Show Them: Essays in Honour of Dov Gabbay*. Ed. by Sergei Artemov, Howard Barringer, Artur S. d'Avila Garcez, Luis C. Lamb, and John Woods. Vol. 2. King's College, London: College Publications (formerly KCL Publications), 2005, pp. 371–440. ISBN: 1-904987-12-5.

[57] Howard Bowman, Helen Cameron, Peter King, and Simon Thompson. "Specification and Prototyping of Structured Multimedia Documents using Interval Temporal Logic". In: *Advances in Temporal Logic*. Ed. by Howard Barringer, Michael Fisher, Dov Gabbay, and Graham Gough. Vol. 16. Applied Logic Series. Springer Verlag, 2000, pp. 435–453. ISBN: 978-90-481-5389-3.
URL: http://dx.doi.org/10.1007/978-94-015-9586-5_22.

[58] Antonio Cau and Hussein Zedan. "The Systematic Construction of Information Systems". In: *Systems Engineering for Business Process Change*. Ed. by Peter Henderson. Download pdf. Springer Verlag, 2000, pp. 264–278. ISBN: 978-1-4471-1146-7.
URL: http://dx.doi.org/10.1007/978-1-4471-0457-5_21.

[59] Wolfgang Grieskamp and Markus Lepper. "Encoding Temporal Logics in Executable Z: A Case Study for the ZETA System". In: *Logic for Programming and Automated Reasoning*. Ed. by Michel Parigot and Andrei Voronkov. Vol. 1955. Lecture Notes in Artificial Intelligence. Springer Verlag, 2000, pp. 43–53. ISBN: 978-3-540-41285-4.
URL: http://dx.doi.org/10.1007/3-540-44404-1_4.

[60] Howard Bowman and Simon Thompson. "A Tableau Method for Interval Temporal Logic with Projection". In: *Automated Reasoning with Analytic Tableaux and Related Methods*. Ed. by Harrie de Swart. Vol. 1397. Lecture Notes in Computer Science. Springer Verlag, 1998, pp. 108–123. ISBN: 978-3-540-64406-4.
URL: http://dx.doi.org/10.1007/3-540-69778-0_17.

[61] Peter King, Helen Cameron, Howard Bowman, and Simon Thompson. "Synchronization in multimedia documents". In: *Electronic Publishing, Artistic Imaging, and Digital Typography*. Ed. by Roger D. Hersch, Jacques André, and Heather Brown. Vol. 1375. Lecture Notes in Computer Science. Springer Verlag, 1998, pp. 355–369. ISBN: 978-3-540-64298-5.
URL: http://dx.doi.org/10.1007/BFb0053283.

[62] Robert Büssow and Wolfgang Grieskamp. "Combining Z and temporal interval logics for the formalization of properties and behaviors of embedded systems". In: *Advances in Computing Science - ASIAN'97*. Ed. by R.K. Shyamasundar and K. Ueda. Vol. 1345. Lecture Notes in Computer Science. Springer Verlag, 1997, pp. 46–56. ISBN: 978-3-540-63875-9.
URL: http://dx.doi.org/10.1007/3-540-63875-X_42.

[63] Shinji Kono. “A combination of clausal and non clausal temporal logic programs”. In: *Executable Modal and Temporal Logics*. Ed. by Michael Fisher and Richard Owens. Vol. 897. Lecture Notes in Computer Science. Springer Verlag, 1995, pp. 40–57. ISBN: 978-3-540-58976-1.
URL: http://dx.doi.org/10.1007/3-540-58976-7_3.

[64] Zhenhua Duan, Maciej Koutny, and Chris Holt. “Projection in temporal logic programming”. In: *Logic Programming and Automated Reasoning*. Ed. by Frank Pfenning. Vol. 822. Lecture Notes in Computer Science. Springer Verlag, 1994, pp. 333–344. ISBN: 978-3-540-58216-8.
URL: http://dx.doi.org/10.1007/3-540-58216-9_48.

[65] Roger W.S. Hale. “Program Compilation”. In: *Towards Verified Systems*. Ed. by Jonathan Bowen. Vol. 2. Real-Time Safety Critical Systems. Elsevier, 1994. Chap. 7, pp. 131–146.
URL: http://dx.doi.org/10.1016/B978-0-444-89901-9.50016-1.

[66] Roger W.S. Hale and He Jifeng. “A Real-time Programming Language”. In: *Towards Verified Systems*. Ed. by Jonathan Bowen. Vol. 2. Real-Time Safety Critical Systems. Elsevier, 1994, pp. 115–130.
URL: http://dx.doi.org/10.1016/B978-0-444-89901-9.50015-X.

[67] Eric C. R. Hehner. “Abstractions of time”. In: *A Classical Mind*. Ed. by A. W. Roscoe. London: Prentice-Hall Int’l., 1994. Chap. 12.

[68] Stephen M. Brien. “A time-interval calculus”. In: *Mathematics of Program Construction*. Ed. by R.S. Bird, C.C. Morgan, and J.C.P. Woodcock. Vol. 669. Lecture Notes in Computer Science. Springer Verlag, 1993, pp. 67–79. ISBN: 978-3-540-56625-0.
URL: http://dx.doi.org/10.1007/3-540-56625-2_8.

[69] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. “Towards refining temporal specifications into hybrid systems”. In: *Hybrid Systems I*. Ed. by R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel. Vol. 736. Lecture Notes in Computer Science. Springer Verlag, 1993, pp. 60–76.
URL: http://dx.doi.org/10.1007/3-540-57318-6_24.

[70] Roger Hale. “Temporal Logic Programming”. In: *Temporal Logics and Their Applications*. Ed. by Antony Galton. London: Academic Press, 1987, pp. 91–119. ISBN: 0-12-274060-2.

[71] Ben Moszkowski. “Executing temporal logic programs”. In: *Seminar on Concurrency*. Ed. by Stephen D. Brookes, Andrew William Roscoe, and Glynn Winskel. Vol. 197. Lecture Notes in Computer Science. Springer Verlag, 1985, pp. 111–130. ISBN: 978-3-540-15670-3.
URL: http://dx.doi.org/10.1007/3-540-15670-4_6.

4.3 Conference Papers

[72] Hanna Klaudel, Maciej Koutny, and Ben Moszkowski. “From Petri Nets with Shared Variables to ITL”. In: *16th International Conference on Application of Concurrency to System Design (ACSD)*. June 2016, pp. 11–18.
URL: http://dx.doi.org/10.1109/ACSD.2016.12.

[73] Ben C. Moszkowski and Dimitar P. Guelev. “An Application of Temporal Projection to Interleaving Concurrency”. In: *Dependable Software Engineering: Theories, Tools, and Applications - First*

International Symposium, SETTA 2015, Nanjing, China, November 4-6, 2015, Proceedings. 2015, pp. 153–167.
URL: http://dx.doi.org/10.1007/978-3-319-25942-0_10.

[74] Andrew Nicholson, Helge Janicke, and Antonio Cau. "Position Paper: Safety and Security Monitoring in ICS/SCADA Systems". In: *Proceedings of the 2nd International Symposium for ICS & SCADA Cyber Security Research 2014*. Download pdf. BCS, 2014, pp. 61–66.
URL: http://dx.doi.org/10.14236/ewic/ics-csr2014.9.

[75] Meng Han, Zhenhua Duan, and Xiaobing Wang. "Time constraints with temporal logic programming". In: *Proceedings of the 14th international conference on Formal Engineering Methods: formal methods and software engineering*. ICFEM'12. Kyoto, Japan: Springer Verlag, 2012, pp. 266–282. ISBN: 978-3-642-34280-6.
URL: http://dx.doi.org/10.1007/978-3-642-34281-3_20.

[76] Amin El-kustaban, Ben Moszkowski, and Antonio Cau. "Formalising of Transactional Memory using Interval Temporal Logic (ITL)". In: *Proceedings of the Spring World Congress on Engineering and Technology (SCET 2012)*. Download pdf. 2012.
URL: http://dx.doi.org/10.1109/SCET.2012.6342060.

[77] Amin El-kustaban, Ben Moszkowski, and Antonio Cau. "Specification Analysis of Transactional Memory using ITL and AnaTempura". In: *Proceedings of International MultiConference of Engineers and Computer Scientists 2012 (IMECS 2012)*. Download pdf. 2012.
URL: https://doaj.org/article/373d1feae6e04b2c90b7457634ddfe4d.

[78] Emad Shafie and Antonio Cau. "A Framework for the Detection and Prevention of SQL Injection". In: *Proceedings of the 11th European Conference on Information Warfare and Security ECIW-2012*. Download pdf. 2012.

[79] Xiaoxiao Yang, Yu Zhang, Ming Fu, and Xinyu Feng. "A concurrent temporal programming model with atomic blocks". In: *Proceedings of the 14th international conference on Formal Engineering Methods: formal methods and software engineering*. ICFEM'12. Kyoto, Japan: Springer Verlag, 2012, pp. 22–37. ISBN: 978-3-642-34280-6.
URL: http://dx.doi.org/10.1007/978-3-642-34281-3_5.

[80] Sulaiman Al Amro and Antonio Cau. "Behaviour-based Virus Detection System using Interval Temporal Logic". In: *CRiSIS 2011, Proceedings of the Sixth International Conference on Risks and Security of Internet and Systems, Timişoara, Romania, September 26-28, 2011*. 2011, pp. 106–111.
URL: http://dx.doi.org/10.1109/CRiSIS.2011.6061544.

[81] Ben C. Moszkowski. "Compositional Reasoning Using Intervals and Time Reversal". In: *Proceedings of the Eighteenth International Symposium on Temporal Representation and Reasoning TIME2011*. 2011, pp. 107–114.
URL: http://dx.doi.org/10.1109/TIME.2011.25.

[82] Jim Woodcock, Marcel Oliveira, Alan Burns, and Kun Wei. "Modelling and Implementing Complex Systems with Timebands". In: *Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI), 2010*. June 2010, pp. 1–13.
URL: http://dx.doi.org/10.1109/SSIRI.2010.7.

[83] Mohamed Sarrab, Helge Janicke, and Antonio Cau. "Interactive runtime monitoring of information flow policies". In: *In proceedings of Second international conference of Creativity and Innovation in Software Engineering*. Download pdf. 2009.

[84] Helge Janicke, Antonio Cau, François Siewe, and Hussein Zedan. "Concurrent Enforcement of Usage Control Policies". In: *9th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2008), 2-4 June 2008, Palisades, New York, USA*. Download pdf. 2008, pp. 111–118.
URL: http://dx.doi.org/10.1109/POLICY.2008.44.

[85] Kamal Lodaya, Paritosh K. Pandya, and Simoni S. Shah. "Marking the chops: an unambiguous temporal logic". In: *Fifth IFIP International Conference On Theoretical Computer Science - TCS 2008, IFIP 20th World Computer Congress, TC 1, Foundations of Computer Science, September 7-10, 2008, Milano, Italy*. 2008, pp. 461–476.
URL: http://dx.doi.org/10.1007/978-0-387-09680-3_31.

[86] Helge Janicke, Antonio Cau, François Siewe, and Hussein Zedan. "A note on the formalisation of UCON". In: *SACMAT 2007, 12th ACM Symposium on Access Control Models and Technologies, Sophia Antipolis, France, June 20-22, 2007, Proceedings*. Download pdf. 2007, pp. 163–168.
URL: http://dx.doi.org/10.1145/1266840.1266867.

[87] Helge Janicke, Antonio Cau, François Siewe, and Hussein Zedan. "Deriving Enforcement Mechanisms from Policies". In: *8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2007), 13-15 June 2007, Bologna, Italy*. Download pdf. 2007, pp. 161–172.
URL: http://dx.doi.org/10.1109/POLICY.2007.15.

[88] Helge Janicke and Linda Finch. "The Role of Dynamic Security Policy in Military Scenarios". In: *In Proceedings of the 6th European Conference on Information Warfare and Security*. 2007.

[89] Helge Janicke, Antonio Cau, François Siewe, Hussein Zedan, and Kevin Jones. "A Compositional Event & Time-Based Policy Model". In: *7th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2006), 5-7 June 2006, London, Ontario, Canada*. Download pdf. 2006, pp. 173–182.
URL: http://dx.doi.org/10.1109/POLICY.2006.2.

[90] Monika Solanki, Antonio Cau, and Hussein Zedan. "ASDL: A Wide Spectrum Language for Designing Web services". In: *Proceedings of 15th International World Wide Web Conference WWW2006*. Download pdf. Edinburgh, Scotland: ACM, 2006.
URL: http://dx.doi.org/10.1145/1135777.1135878.

[91] Helge Janicke, François Siewe, Kevin Jones, Antonio Cau, and Hussein Zedan. "Analysis and Runtime Verification of Dynamic Security Policies". In: *In Proceedings of The First Workshop on Defence Applications for Multi-Agent Systems (DAMAS'05)*. Ed. by Robert Ghanea-Hercock and Mark Greaves and Nick Jennings and Simon Thompson. Vol. 3890. Lecture Notes in Computer Science. Download pdf. Utrecht, The Netherlands: Springer Verlag, July 2005, pp. 92–103.
URL: http://dx.doi.org/10.1007/11683704_8.

[92] François Siewe, Helge Janicke, and Kevin Jones. "Dynamic Access Control Policies and Web-Service Composition". In: *Proceedings of the 1st Young Researchers Workshop on Service Oriented Com-*

puting (YR-SOC 05). 2005.

[93] Monika Solanki, Antonio Cau, and Hussein Zedan. "Semantically annotating reactive web services with temporal specifications". In: *Proceedings of the IEEE ICWS 2005, Second International Workshop on Semantic and Dynamic Web Processes*. Vol. 140. Download pdf. 2005.

[94] Monika Solanki, Antonio Cau, and Hussein Zedan. "Augmenting semantic web service descriptions with compositional specification". In: *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*. Ed. by Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills. Download pdf. ACM, 2004, pp. 544–552. ISBN: 1-58113-844-X.
URL: http://dx.doi.org/10.1145/988672.988746.

[95] Monika Solanki, Antonio Cau, and Hussein Zedan. "Introducing Compositionality in Web Service Descriptions". In: *10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004), 26-28 May 2004, Suzhou, China*. Download pdf. IEEE Computer Society, 2004, pp. 14–20. ISBN: 0-7695-2118-5.
URL: http://dx.doi.org/10.1109/FTDCS.2004.1316588.

[96] Martin Wildmoser and Tobias Nipkow. "Certifying Machine Code Safety: Shallow versus Deep Embedding". In: *Theorem Proving in Higher Order Logics (TPHOLs 2004)*. Ed. by K. Slind, A. Bunker, and G. Gopalakrishnan. Vol. 3223. LNCS. Springer, 2004, pp. 305–320.

[97] Ben Moszkowski. "A Hierarchical Completeness Proof for Interval Temporal Logic with Finite Time (Preliminary Version)". In: *Proc. of the Workshop on Interval Temporal Logics and Duration Calculi (part of* 15th European Summer School in Logic Language and Information (ESSLLI-2003)*)*. Vienna, Aug. 2003, pp. 41–65.

[98] Ben Moszkowski. "A Hierarchical Completeness Proof for Propositional Temporal Logic". In: *Verification–Theory and Practice: Proceedings of an International Symposium in Honor of Zohar Manna's 64th Birthday (Taormina, Sicily, Italy, June 29 - July 4, 2003)*. Ed. by Nachum Dershowitz. Vol. 2772. Lecture Notes in Computer Science. Berlin: Springer Verlag, 2003, pp. 480–523. ISBN: 3-540-21002-4.
URL: http://dx.doi.org/10.1007/978-3-540-39910-0_22.

[99] François Siewe, Antonio Cau, and Hussein Zedan. "A Compositional Framework for Access Control Policies Enforcement". In: *Proceedings of the 2003 ACM Workshop on Formal Methods in Security Engineering*. FMSE '03. Download pdf. Washington, D.C.: ACM, 2003, pp. 32–42. ISBN: 1-58113-781-8.
URL: http://dx.doi.org/10.1145/1035429.1035433.

[100] Monika Solanki, Antonio Cau, and Hussein Zedan. "Introducing compositionality in webservice descriptions". In: *proceedings of the 3rd International Anwire Workshop on Adaptable Service Provision*. Download pdf. Springer Verlag, 2003.

[101] Roger Hale. "Using ITL for Codesign". In: *Proceedings of the Verification Workshop part of the International Joint Conference on Automated Reasoning IJCAR'2001*. 2001.

[102] Hussein Zedan and Antonio Cau. "Voice Over IP: Correct Hardware/Software Co-design". In: *8th IEEE Workshop on Future Trends of Distributed Computer Systems (FTDCS 2001), 31 October, 2 November 2001, Bologna, Italy, Proceedings*. IEEE Computer Society, 2001, pp. 194–200. ISBN: 0-7695-1384-0.
URL: http://dx.doi,org/10.1109/FTDCS.2001.969641.

[103] Hussein Zedan et al. "K-Mediator: Towards Evolving Information Systems". In: *Software Maintenance, 2001. Proceedings. IEEE International Conference on*. Download pdf. 2001, pp. 520–527.
URL: http://dx.doi.org/10.1109/ICSM.2001.972765.

[104] Jordan Dimitrov. "Compositional Reasoning about Events in Interval Temporal Logic". In: *Proc. of The Fifth International Conference on Computer Science and Informatics*. 2000.

[105] Wang Jianzhong, Xu Qiwen, and Ma Huadong. "Modelling and Verification of a Network Player System with DCValid". In: *Proc. of the First Asia-Pacific Conference on Quality Software (APAQS 2OOO)*. Ed. by T. H. Tse and T. Y. Chen. Hong Kong: IEEE Computer Society Press, Oct. 2000, pp. 44–49.
URL: http://dx.doi.org/10.1109/APAQ.2000.883777.

[106] Kamal Lodaya. "Sharpening the Undecidability of Interval Temporal Logic". In: *Advances in Computing Science - ASIAN 2000: Proc. of Sixth Asian Computing Science Conference*. Ed. by He Jifeng and Masahiko Sato. Vol. 1961. Lecture Notes in Computer Science. Penang, Malaysia: Springer Verlag, Nov. 2000. ISBN: 3-540-41428-2.
URL: http://dx.doi.org/10.1007/3-540-44464-5_21.

[107] Ben Moszkowski. "A Complete Axiomatization of Interval Temporal Logic with Infinite Time (Extended Abstract)". In: *Proc. of the 15th Annual IEEE Symposium on Logic in Computer Science (LICS 2000)*. Download pdf. IEEE Computer Society Press, June 2000, pp. 242–251. ISBN: 0–7695–0725–5.
URL: http://dx.doi.org/10.1109/LICS.2000.855773.

[108] Ben Moszkowski. "An Automata-Theoretic Completeness Proof for Interval Temporal Logic (Extended Abstract)". In: *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP 2000)*. Ed. by Ugo Montanari, José Rolim, and Emo Welzl. Vol. 1853. Lecture Notes in Computer Science. Download pdf. Geneva, Switzerland: Springer Verlag, July 2000, pp. 223–234. ISBN: 3-540-67715-1.
URL: http://dx.doi.org/10.1007/3-540-45022-X_19.

[109] Arun C. Rao, Antonio Cau, and Hussein Zedan. "Visualization of Interval Temporal Logic". In: *Proc. of the Fifth International Conference on Computer Science and Informatics*. Download pdf. 2000.

[110] Hussein Zedan, Antonio Cau, and Ben C. Moszkowski. "Compositional Modelling: The Formal Perspective". In: *Proc. of Workshop on Systems Modelling for Business Process Improvement*. Ed. by

David Bustard. Download pdf. Artech House, 2000, pp. 333–354.

[111] Hussein Zedan, Antonio Cau, and Shikun Zhou. "A Calculus for Evolution". In: *Proc. of the Fifth International Conference on Computer Science and Informatics*. Download pdf. 2000.

[112] XiaoShan Li. "Formal Semantics of Verilog HDL". In: *CDROM: Proceedings of the Second Forum on Design Languages (FDL '99)*. Lyon, France: ECSI Verlag, Gières, Aug. 1999, pp. 127–136. ISBN: 2-84010-033-9.

[113] XiaoShan Li. "Specification and Simulation of a Concurrent Real-Time System". In: *IEEE Proceedings of International Symposium on Software Engineering for Parallel and Distributed Systems*. Los Angeles, California: IEEE Computer Society, May 1999, pp. 197–204.
URL: http://dx.doi.org/10.1109/PDSE.1999.779752.

[114] Matthew J. Morley. "Semantics of Temporal *e*". In: Banff'99 *Higher Order Workshop: Formal Methods in Computation, Ullapool, Scotland, 9–11 Sept. 1999*. Ed. by T. F. Melham and F. G. Moller. University of Glasgow, Department of Computing Science Technical Report. 1999, pp. 138–142.

[115] Markus Müller-Olm. "A Modal Fixpoint Logic with Chop". In: *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*. Ed. by Christoph Meinel and Sophie Tison. Vol. 1563. Lecture Notes in Computer Science. Trier, Germany: Springer Verlag, Mar. 1999, pp. 510–520.
URL: http://dx.doi.org/10.1007/3-540-49116-3_48.

[116] Thomas M. Rasmussen. "Signed Interval Logic". In: *Computer Science Logic, 13th International Workshop, CSL '99, 8th Annual Conference of the EACSL*. Ed. by Jörg Flum and Mario Rodríguez-Artalejo. Vol. 1683. Lecture Notes in Computer Science. Berlin, 1999, pp. 157–171. ISBN: 3-540-66536-6.
URL: http://dx.doi.org/10.1007/3-540-48168-0_12.

[117] Shikun Zhou, Hussein Zedan, and Antonio Cau. "A Framework For Analysing The Effect of 'Change' In Legacy Code". In: *IEEE Proc. of ICSM'99*. Download pdf. IEEE. 1999.
URL: http://dx.doi,org/10.1109/ICSM.1999.792639.

[118] Antonio Cau, Chris Czarnecki, and Hussein Zedan. "Designing a Provably Correct Robot Control System using a 'Lean' Formal Method". In: *Proceedings 5th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'98)*. Ed. by Anders P. Ravn and Hans Rischel. Vol. 1486. Lecture Notes in Computer Science. Download pdf. Springer Verlag, 1998, pp. 123–132.
URL: http://dx.doi.org/10.1007/BFb0055342.

[119] Ben Moszkowski. "Compositional reasoning using Interval Temporal Logic and Tempura". In: *Compositionality: The Significant Difference*. Ed. by Willem-Paul de Roever, Hans Langmaack, and Amir Pnueli. Vol. 1536. Lecture Notes in Computer Science. Download pdf. Berlin: Springer Verlag, 1998, pp. 439–464. ISBN: 3-540-65493-3.
URL: http://dx.doi.org/10.1007/3-540-49213-5_17.

[120] Antonio Cau and Hussein Zedan. "Refining Interval Temporal Logic specifications". In: *Transformation-Based Reactive Systems Development*. Ed. by M. Bertran and T. Rus. Vol. 1231. Lecture Notes in Computer Science. Download pdf. AMAST. Springer Verlag, 1997, pp. 79–94.
URL: http://dx.doi.org/10.1007/3-540-63010-4_6.

[121] XiaoShan Li, Antonio Cau, Ben Moszkowski, and Nick Coleman. "Proving the correctness of the interlock mechanism in processor design". In: *Advances in Hardware Design and Verification*. Ed. by Hon F. Li and David K. Probst. Download pdf. London: IFIP/Chapman and Hall, 1997, pp. 2–22.

[122] Antonio Cau, Hussein Zedan, Nick Coleman, and Ben Moszkowski. "Using ITL and TEMPURA for large scale specification and simulation". In: *Proc. of the 4th Euromicro Workshop on Parallel and Distributed Processing*. Download pdf. IEEE Computer Society Press, 1996, pp. 493–500.
URL: http://dx.doi.org/10.1109/EMPDP.1996.500624.

[123] Ben Moszkowski. "Using temporal fixpoints to compositionally reason about liveness". In: *BCS-FACS 7th Refinement Workshop*. Ed. by He Jifeng, John Cooke, and Peter Wallis. electronic Workshops in Computing. Download pdf. BCS-FACS. London: Springer Verlag and British Computer Society, 1996. ISBN: 3-540-76104-7.

[124] Jonathan Bowen, He Jifeng, Roger Hale, and John Herbert. "Towards Verified Systems: The SAFEMOS Project". In: *The Mathematics of Dependable Systems*. Ed. by C. Mitchell and V. Stavridou. The Institute of Mathematics and its Applications Conference Series. Oxford: Oxford University Press, 1995.

[125] Bruno Dutertre. "Complete proof systems for first order interval temporal logic". In: *Proc. of the 10th Annual IEEE Symposium on Logic in Computer Science*. Los Alamitos, Calif., USA: IEEE Computer Society Press, June 1995, pp. 36–43.
URL: http://dx.doi.org/10.1109/LICS.1995.523242.

[126] Ben Moszkowski. "Compositional reasoning about projected and infinite time". In: *Proceedings of the First IEEE Int'l Conf. on Engineering of Complex Computer Systems (ICECCS'95)*. Download pdf. Los Alamitos, California: IEEE Computer Society Press, 1995, pp. 238–245.
URL: http://dx.doi.org/10.1109/ICECCS.1995.479336.

[127] R. Elliot. "An exercise in formally based circuit synthesis from a behavioural specification in Interval Temporal Logic". In: *Proc. EUROMICRO '94*. 1994.
URL: http://dx.doi.org/10.1109/EURMIC.1994.390402.

[128] Arjun Kapur, Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. "Proving safety properties of hybrid systems". In: *Formal Techniques in Real-Time and Fault-Tolerant Systems*. Ed. by Hans Langmaack, Willem-Paul de Roever, and Jan Vytopil. Vol. 863. Lecture Notes in Computer Science. Springer Verlag, 1994, pp. 431–454. ISBN: 978-3-540-58468-1.
URL: http://dx.doi.org/10.1007/3-540-58468-4_177.

[129] Ben Moszkowski. "Some very compositional temporal properties". In: *Programming Concepts, Methods and Calculi*. Ed. by E.-R. Olderog. Vol. A-56. IFIP Transactions. IFIP. Elsevier Science B.V. (North–Holland), 1994, pp. 307–326.

[130] Masahiro Fujita and Shinji Kono. "Synthesis of Controllers from Interval Temporal Logic Specification". In: *International Conference on Computer Design: VLSI in Computers and Processors*. IEEE Computer Society Press, 1993.
URL: http://dx.doi.org/10.1109/ICCD.1993.393373.

[131] Alastair R. Ruddle. "Formal Methods in the Specification of Real-Time, Safety-Critical Control Systems". In: *Z User Workshop, London 1992*. Ed. by J. P. Bowen and J. E. Nicholls. Workshops in Computing. Springer Verlag, 1993, pp. 131–146. ISBN: 3-540-19818-0.
URL: http://dx.doi.org/10.1007/978-1-4471-3556-2_10.

[132] He JiFeng and Jonathan Bowen. "Time Interval Semantics and Implementation of a Real-Time Programming Language". In: *Proceedings of the 4-th Euromicro Workshop on Real-Time Systems*. IEEE Computer Society Press, 1992, pp. 110–115.
URL: http://dx.doi.org/10.1109/EMWRT.1992.637480.

[133] Shinji Kono. "Automatic Verification of Interval Temporal Logic". In: *8th British Colloquium For Theoretical Computer Science*. 1992.

[134] Y. Srivinas Ramakrishna, Laura K. Dillon, Louise E. Moser, Peter M. Melliar-Smith, and George Kutty. "An Automata-theoretic decision procedure for Future Interval Logic". In: *Proc. 12th FST& TCS*. Vol. 652. Lecture Notes in Computer Science. Springer Verlag, Dec. 1992, pp. 51–67.
URL: http://dx.doi.org/10.1007/3-540-56287-7_94.

[135] Roger Hale. "Using temporal logic for prototyping: The design of a lift controller". In: *Temporal Logic in Specification*. Ed. by B. Banieqbal, H. Barringer, and A. Pnueli. Vol. 398. Lecture Notes in Computer Science. Berlin: Springer Verlag, 1989, pp. 375–408.
URL: http://dx.doi.org/10.1007/3-540-51803-7_35.

[136] Danny Kilis, Albert C. Esterline, and James R. Slagle. "Specification and verification of network protocols using executable temporal logic". In: *Proceedings of IFIP Congress '89*. North Holland Publishing Co. Amsterdam, 1989, pp. 845–850.

[137] Barbara Paech. "Gentzen-systems for propositional temporal logics". In: *Proceedings of the 2nd Workshop on Computer Science Logic, Duisburg (FRG)*. Ed. by E. Börger, H. Kleine Büning, and M. M. Richter. Vol. 385. Lecture Notes in Computer Science. Springer Verlag, Oct. 1988, pp. 240–253.
URL: http://dx.doi.org/10.1007/BFb0026305.

[138] Roger Hale and Ben C. Moszkowski. "Parallel Programming in Temporal Logic". In: *PARLE, Parallel Architectures and Languages Europe, Volume II: Parallel Languages*. Ed. by J. W. de Bakker, A. J. Nijman, and Philip C. Treleaven. Vol. 259. Lecture Notes in Computer Science. Eindhoven, The Netherlands: Springer Verlag, June 1987, pp. 277–296. ISBN: 3-540-17945-3.
URL: http://dx.doi.org/10.1007/3-540-17945-3_16.

[139] Masahiro Fujita, Makoto Ishisone, Hiroshi Nakamura, Hidehiko Tanaka, and Tatsuhiko Moto-oka. "Using the temporal logic programming language Tokio for algorithm description and automatic CMOS gate array synthesis". In: *Logic Programming '85*. Ed. by Eiiti Wada. Vol. 221. Lecture Notes in Computer Science. Springer Verlag, 1986, pp. 246–255. ISBN: 978-3-540-16479-1.
URL: http://dx.doi.org/10.1007/3-540-16479-0_24.

[140] Masahiro Fujita, Shinji Kono, Hidehiko Tanaka, and Tatsuhiko Moto-oka. "Tokio: Logic programming language based on temporal logic and its compilation to Prolog". In: *Third International Conference on Logic Programming*. Ed. by Ehud Shapiro. Vol. 225. Lecture Notes in Computer Science. Springer Verlag, 1986, pp. 695–709. ISBN: 978-3-540-16492-0.
URL: http://dx.doi.org/10.1007/3-540-16492-8_119.

[141] Roni Rosner and Amir Pnueli. "A choppy logic". In: *First Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, June 1986, pp. 306–313. ISBN: 0-8186-0720-3.

[142] Shinji Kono, Tatsuya Aoyagi, Masahiro Fujita, and Hidehiko Tanaka. "Implementation of Temporal Logic Programming Language Tokio". In: *Logic Programming '85*. Ed. by Eiiti Wada. Vol. 221. Lecture Notes in Computer Science. Springer Verlag, 1985, pp. 138–147.
URL: http://dx.doi.org/10.1007/3-540-16479-0_14.

[143] Ben Moszkowski. "A temporal analysis of some concurrent systems". In: *The Analysis of Concurrent Systems*. Ed. by B. T. Denvir, W. T .Harwood, M. I. Jackson, and M. J. Wray. Vol. 207. Lecture Notes in Computer Science. Berlin: Springer Verlag, 1985, pp. 359–364.
URL: http://dx.doi.org/10.1007/3-540-16047-7_58.

[144] Joseph Halpern, Zohar Manna, and Ben Moszkowski. "A hardware semantics based on temporal intervals". In: *Proceedings of the 10-th International Colloquium on Automata, Languages and Programming*. Ed. by Josep Diaz. Vol. 154. Lecture Notes in Computer Science. Berlin: Springer Verlag, 1983, pp. 278–291.
URL: http://dx.doi.org/10.1007/BFb0036915.

[145] Ben Moszkowski. "A temporal logic for multi-level reasoning about hardware". In: *Proceedings of the 6-th International Symposium on Computer Hardware Description Languages*. Pittsburgh, Pennsylvania: North-Holland Pub. Co., May 1983, pp. 79–90.

[146] Ben Moszkowski and Zohar Manna. "Reasoning in Interval Temporal Logic". In: *Proceedings of the Workshop on Logics of Programs*. Ed. by Edmund Clarke and Dexter Kozen. Vol. 164. Lecture Notes in Computer Science. Pittsburgh, PA: Springer Verlag, June 1983, pp. 371–382. ISBN: 3-540-12896-4.
URL: http://dx.doi.org/10.1007/3-540-12896-4_374.

[147] Richard L. Schwartz, Peter M. Melliar-Smith, and Friedrich H. Vogt. "An Interval Logic for Higher-Level Temporal Reasoning". In: *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. Aug. 1983, pp. 173–186.
URL: http://dx.doi.org/10.1145/800221.806720.

4.4 Books

[148] Ernst-Rüdiger Olderog and Henning Dierks. *REAL-TIME SYSTEMS: Formal Specification and Automatic Verification*. Cambridge University Press, 2008.

[149] Valentin Goranko and Angelo Montanari, eds. *Special issue on Interval Temporal Logics and Duration Calculi*. Vol. 14. Journal of Applied Non-Classical Logics 1–2. Lavoisier, 2004.
URL: http://www.tandfonline.com/toc/tncl20/14/1-2.

[150] Chaochen Zhou and Michael R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. Monographs in Theoretical Computer Science (An EATCS series). Springer Verlag, 2004.

[151] Jonathan Bowen, ed. *Towards Verified Systems*. Vol. 2. Real-Time Safety Critical Systems. Amsterdam: Elsevier Science B.V. (North–Holland), 1994. ISBN: 978-0-444-89901-9.
URL: http://www.sciencedirect.com/science/bookseries/15725960/2.

[152] Ben Moszkowski. *Executing Temporal Logic Programs*. Download pdf. Cambridge, England: Cambridge University Press, 1986.

4.5 Theses

[153] Bader Alouffi. "Run Time verification of Hybrid Systems". PhD thesis. De Montfort University, 2016.
URL: https://www.dora.dmu.ac.uk/handle/2086/12490.

[154] Sami Alsarhani. "Reasoning about History Based Access Control Policy Using Past Time Operators of Interval Temporal Logic". PhD thesis. De Montfort University, 2014.
URL: https://www.dora.dmu.ac.uk/handle/2086/10406.

[155] Sulaiman Al Amro. "Behaviour-based Virus Analysis and Detection". PhD thesis. De Montfort University, 2013.
URL: https://www.dora.dmu.ac.uk/handle/2086/9488.

[156] Emad Shafie. "Runtime Detection and Prevention for Structure Query Language Injection Attacks". PhD thesis. De Montfort University, 2013.
URL: https://www.dora.dmu.ac.uk/handle/2086/10076.

[157] Turki Mohammed Alghamdi. "Policy-based Runtime Tracking for E-learning Environments". PhD thesis. De Montfort University, 2012.
URL: https://www.dora.dmu.ac.uk/handle/2086/8237.

[158] Amin Mohammed El-kustaban. "Studying and Analysing Transactional Memory Using Interval Temporal Logic and AnaTempura". PhD thesis. De Montfort University, 2012.
URL: https://www.dora.dmu.ac.uk/handle/2086/6900.

[159] Helge T. Janicke. "The Development of Secure Multi-Agent Systems". PhD thesis. De Montfort University, 2007.
URL: https://www.dora.dmu.ac.uk/xmlui/handle/2086/4809.

[160] Frederick V. Ramsey. "A General Algebra of Business Rules for Heterogeneous Systems". PhD thesis. De Montfort University, 2007.
URL: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.438868.

[161] François Siewe. "A Compositional Framework for the Development of Secure Access Control Systems". PhD thesis. De Montfort University, 2005.
URL: http://www.cse.dmu.ac.uk/~fsiewe/papers/fsiewephd.pdf.

[162] Monika Solanki. "A Compositional Framework for the Specification, Verification and Runtime Validation of Reactive Web Services". PhD thesis. De Montfort University, 2005.
URL: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.422585.

[163] Shikun Zhou. "Compositional Framework for the Guided Evolution of Time-Critical Systems". PhD thesis. De Montfort University, 2003.
URL: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.250763.

[164] Jordan Dimitrov. "Formal Compositional Design of Mixed Hardware/Software Systems with semantics of Verilog HDL". PhD thesis. De Montfort University, 2002.
URL: http://www.cse.dmu.ac.uk/~jordan/root.pdf.

[165] Arun Chakrapnani Rao. "A Visual Framework for Formal Systems Development Using Interval Temporal Logic". PhD thesis. De Montfort University, 2002.
URL: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.393497.

[166] Xiaodong Liu. "Abstraction: A Notion For Reverse Engineering". PhD thesis. De Montfort University, 1999.
URL: https://www.dora.dmu.ac.uk/handle/2086/4214.

[167] Arjun Kapur. "Interval and Point-Based Approaches to Hybrid System Verification". PhD thesis. Dept. of Computer Science, Stanford University, Sept. 1997.
URL: http://theory.stanford.edu/people/arjun/thesis.ps.

[168] Zhenhua H. Duan. "An Extended Interval Temporal Logic and a Framing Technique for Temporal Logic Programming". PhD thesis. Dept. of Computing Science, University of Newcastle Upon Tyne, May 1996.
URL: https://theses.ncl.ac.uk/dspace/handle/10443/2075.

[169] Stephen M. Brien. "A relational calculus of intervals". MA thesis. Programming Research Group, Oxford University, Dec. 1990.

[170] Roger W. S. Hale. "Programming in Temporal Logic". PhD thesis. Cambridge University, England, Oct. 1988.

[171] Randall W. Lichota. "Evaluating Hardware Architectures for Real-Time Parallel Algorithms using Temporal Specifications". PhD thesis. Computer Science Department, University of California at Los Angeles, 1988.
URL: http://www.osti.gov/scitech/biblio/5213399.

[172] Ben Moszkowski. "Reasoning about Digital Circuits". PhD thesis. Department of Computer Science, Stanford University, 1983.
URL: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA136634.

4.6 Technical Reports

[173] Howard Bowman and Simon J. Thompson. *A Complete Axiomatization of Interval Temporal Logic with Projection*. Technical Report 6-00. Canterbury, Great Britain: Computing Laboratory, Univer-

sity of Kent, Jan. 2000, p. 52.
URL: https://kar.kent.ac.uk/22055/.

[174] Wang Hanpin and Xu Qiwen. *Temporal logics over infinite intervals*. Tech. rep. 158. Macau: UNU/IIST, 1999.

[175] Stephan Merz. "An Encoding of TLA in Isabelle". https://members.loria.fr/SMerz/projects/isabelle-tla/index.html. Part of the Isabelle distribution. 1998.

[176] Howard Bowman, Helen Cameron, Peter King, and Simon Thompson. *Mexitl: Multimedia in Executable Interval Temporal Logic*. Tech. rep. 3-97. Computing Laboratory, University of Kent at Canterbury, May 1997.
URL: https://kar.kent.ac.uk/14016/.

[177] Antonio Cau and Ben Moszkowski. *Using PVS for Interval Temporal Logic Proofs. Part 1: The syntactic and semantic encoding*. Technical monograph 14. Download pdf. Leicester: SERCentre, De Montfort University, 1996.

[178] Ben Moszkowski. *Compositional Reasoning about Projected and Infinite time*. Tech. rep. EE/0495/M1. Newcastle upon Type, UK: Dept. of Elec. and Elec. Eng., Univ. of Newcastle, UK, Apr. 1995.

[179] Ben Moszkowski. *Embedding Imperative Constructs in Interval Temporal Logic*. Tech. rep. Internal memorandum EE/0895/M1. Newcastle upon Type, UK: Dept. of Elec. and Elec. Eng., Univ. of Newcastle, UK, Aug. 1995.

[180] Bruno Dutertre. *On first order interval temporal logic*. Tech. rep. CSD-TR-94-3. Egham, Surrey TW20 0EX, England: Dept. of Computer Science, Royal Holloway, University of London, 1994.

[181] Ben Moszkowski. *Some very compositional temporal properties*. Tech. rep. 466. Dept. of Computing Science, University of Newcastle, Dec. 1993.

[182] Ben Moszkowski. *Executing Temporal Logic Programs*. Tech. rep. 55. Computer Laboratory, University of Cambridge, 1984.