Interval Temporal Logic

Antonio Cau, Ben Moszkowski and Hussein Zedan
Software Technology Research Laboratory

December 8, 2009

HTML version of the ITL home page
ITL-course: A not so short introduction to ITL

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[15]. Timing constraints are expressible and furthermore most imperative programming constructs can be viewed as formulas in a slightly modified version of ITL [25]. Tempura provides an executable framework for developing and experimenting with suitable ITL specifications. In addition, ITL and its mature executable subset Tempura [9] 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, various researchers have applied Tempura to hardware simulation and other areas where timing is important.

1 Syntax

The key notion of ITL is an *interval*. An interval σ is considered to be a (in)finite sequence of states $\sigma_0,\ \sigma_1 \ldots$, where a state σ_i is a mapping from the set of variables Var to the set of values Val. 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).

The syntax of ITL is defined in Table 1 where
z is an integer value,
a is a static integer variable (doesn't change within an interval),
A is a state integer variable (can change within an interval),
v a static or state integer variable,
g is a integer function symbol,

Table 1: Syntax of ITL

Expressions	
$e ::=$	$z \mid a \mid A \mid g(e_1, \ldots, e_n) \mid \bigcirc A \mid \mathsf{fin}\ A$
Formulae	
$f ::=$	$\mathsf{true} \mid q \mid Q \mid h(e_1, \ldots, e_n) \mid \neg f \mid f_1 \wedge f_2 \mid \forall \boldsymbol{v} \cdot f \mid$ $\mathsf{skip} \mid f_1 \,;\, f_2 \mid f^*$

q is a static Boolean variable (doesn't change within an interval),
Q is a state Boolean variable (can change within an interval),
h is a predicate symbol.

2 Semantics

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.
- $\mathsf{fin}\ A$: 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 $[\![\ldots]\!]^e$ be the "meaning" (semantic) function from $\mathit{Expressions} \times (\Sigma^+ \cup \Sigma^\omega)$ to Val and let $[\![\ldots]\!]$ be the "meaning" function from $\mathit{Formulae} \times (\Sigma^+ \cup \Sigma^\omega)$ to Bool (set of Boolean values, $\{\mathrm{tt}, \mathrm{ff}\}$) and let $\sigma = \sigma_0\sigma_1 \ldots$ be 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. The formal semantics is listed in Table 2:

3 Derived Constructs

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

Table 2: Semantics of ITL

$[\![z]\!]^e_\sigma$	$=$	$\hat{z}$
$[\![a]\!]^e_\sigma$	$=$	$\sigma_0(a)$ and for all $0 < i \leq \lvert\sigma\rvert, \sigma_i(a) = \sigma_0(a)$
$[\![A]\!]^e_\sigma$	$=$	$\sigma_0(A)$
$[\![g(e_1, \ldots, e_n)]\!]^e_\sigma$	$=$	$\hat{g}([\![e_1]\!]^e_\sigma, \ldots, [\![e_n]\!]^e_\sigma)$
$[\![\bigcirc A]\!]^e_\sigma$	$=$	$\begin{cases} \sigma_1(A) & \text{if } \lvert\sigma\rvert > 0 \\ \text{choose-any-from}(Val) & \text{otherwise} \end{cases}$
$[\![\mathsf{fin}\ A]\!]^e_\sigma$	$=$	$\begin{cases} \sigma_{\lvert\sigma\rvert}(A) & \text{if } \sigma \text{ is finite} \\ \text{choose-any-from}(Val) & \text{otherwise} \end{cases}$
$[\![\mathsf{true}]\!]_\sigma$	$=$	tt
$[\![q]\!]_\sigma$	$=$	$\sigma_0(q)$ and for all $0 < i \leq \lvert\sigma\rvert, \sigma_i(q) = \sigma_0(q)$
$[\![Q]\!]_\sigma$	$=$	$\sigma_0(Q)$
$[\![h(e_1, \ldots, e_n)]\!]_\sigma = \mathrm{tt}$	iff	$\hat{h}([\![e_1]\!]^e_\sigma, \ldots, [\![e_n]\!]^e_\sigma)$
$[\![\neg f]\!]_\sigma = \mathrm{tt}$	iff	not $([\![f]\!]_\sigma = \mathrm{tt})$
$[\![f_1 \wedge f_2]\!]_\sigma = \mathrm{tt}$	iff	$([\![f_1]\!]_\sigma = \mathrm{tt})$ and $([\![f_2]\!]_\sigma = \mathrm{tt})$
$[\![\mathsf{skip}]\!]_\sigma = \mathrm{tt}$	iff	$\lvert\sigma\rvert = 1$
$[\![\forall \boldsymbol{v} \cdot f]\!] = \mathrm{tt}$	iff	(for all σ' s.t. $\sigma \sim_v \sigma', [\![f]\!]_{\sigma'} = \mathrm{tt}$)
$[\![f_1 ; f_2]\!]_\sigma = \mathrm{tt}$	iff	

  (exists k, s.t. $[\![f_1]\!]_{\sigma_0 \ldots \sigma_k} = \mathrm{tt}$ and $[\![f_2]\!]_{\sigma_k \ldots \sigma_{\lvert\sigma\rvert}} = \mathrm{tt}$)
  or (σ is infinite and $[\![f_1]\!]_\sigma = \mathrm{tt}$)

$[\![f^*]\!] = \mathrm{tt}$ iff

if σ is finite then
  (exist $l_0, \ldots, l_n$ s.t. $l_0 = 0$ and $l_n = \lvert\sigma\rvert$ and
  for all $0 \leq i < n, l_i \leq l_{i+1}$ and $[\![f]\!]_{\sigma_{l_i} \cdots \sigma_{l_{i+1}}} = \mathrm{tt}$)
else
  (exist $l_0, \ldots, l_n$ s.t. $l_0 = 0$ and
  $[\![f]\!]_{\sigma_{l_n} \cdots \lvert\sigma\rvert} = \mathrm{tt}$ and
  for all $0 \leq i < n, l_i \leq l_{i+1}$ and $[\![f]\!]_{\sigma_{l_i} \cdots \sigma_{l_{i+1}}} = \mathrm{tt}$)
  or
  (exist an infinite number of l_i s.t. $l_0 = 0$ and
  for all $0 \leq i, l_i \leq l_{i+1}$ and $[\![f]\!]_{\sigma_{l_i} \cdots \sigma_{l_{i+1}}} = \mathrm{tt}$)

Table 3: Frequently used non-temporal derived constructs

false	$\widehat{=}$	$\neg\mathsf{true}$	false value
$f_1 \lor f_2$	$\widehat{=}$	$\neg(\neg f_1 \land \neg f_2)$	or
$f_1 \supset f_2$	$\widehat{=}$	$\neg f_1 \lor f_2$	implies
$f_1 \equiv f_2$	$\widehat{=}$	$(f_1 \supset f_2) \land (f_2 \supset f_1)$	equivalent
$\exists \boldsymbol{v} \bullet f$	$\widehat{=}$	$\neg \forall \boldsymbol{v} \bullet \neg f$	exists

Table 4: Frequently used temporal derived constructs

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

4 Propositional proof system

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

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_v^e that in formula f expression e is substituted for variable v.

Table 5: Frequently used concrete derived constructs

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

Table 6: Frequently used derived constructs related to expressions

$A := exp$	$\widehat{=}$	$\bigcirc A = exp$	assignment
$A \approx exp$	$\widehat{=}$	$\Box(A = exp)$	equal in interval
$A \leftarrow exp$	$\widehat{=}$	finite $\wedge$ (fin A) $= exp$	temporal assignment
A gets exp	$\widehat{=}$	keep $(A \leftarrow exp)$	gets
stable A	$\widehat{=}$	A gets A	stability
padded A	$\widehat{=}$	(stable (A) ; skip) $\vee$ empty	padded expression
$A \lessdot\!\sim exp$	$\widehat{=}$	$(A \leftarrow exp) \wedge$ padded A	padded temporal assignment
goodindex A	$\widehat{=}$	keep $(A \leftarrow A \vee A \leftarrow A + 1)$	increasing index
intlen (exp)	$\widehat{=}$	$\exists \boldsymbol{I} \cdot (I = 0) \wedge (I$ gets $I + 1) \wedge (I \leftarrow exp)$	interval length

Table 7: Propositional Axioms and Rules for 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	$\vdash\ \mathsf{empty}; f_1 \ \equiv\ f_1$
ChopEmpty	$\vdash\ f_1; \mathsf{empty} \ \equiv\ f_1$
BiBoxChopImpChop	$\vdash\ \boxed{\scriptstyle i}(f_0 \ \supset\ f_1) \wedge \Box(f_2 \supset f_3) \ \supset\ (f_0; f_2) \ \supset\ (f_1; f_3)$
StateImpBi	$\vdash\ p \ \supset\ \boxed{\scriptstyle i}\, p$
NextImpNotNextNot	$\vdash\ \bigcirc f_0 \ \supset\ \neg \bigcirc \neg f_0$
KeepnowImpNotKeepnowNot	$\vdash\ \mathsf{keepnow}\,(f_0) \ \supset\ \neg\, \mathsf{keepnow}\,(\neg f_0)$
BoxInduct	$\vdash\ f_0 \wedge \Box(f_0 \ \supset\ \textcircled{w}\, f_0) \ \supset\ \Box\, f_0$
InfChop	$\vdash\ (f_0 \wedge \mathsf{inf})\ ;\ f_1 \ \equiv\ (f_0 \wedge \mathsf{inf})$
ChopStarEqv	$\vdash\ f_0^* \ \equiv\ (\mathsf{empty} \vee ((f_0 \wedge \mathsf{more})\ ;\ f_0^*))$
ChopstarInduct	$\vdash\ (\mathsf{inf} \wedge f_0 \wedge \Box(f_0 \ \supset\ (f_1 \wedge \mathsf{fmore})\ ;\ f_0)) \ \supset\ f_1^*$
MP	$\vdash\ f_0 \ \supset\ f_1,\ \vdash\ f_0 \ \Rightarrow \vdash\ f_1$
BoxGen	$\vdash\ f_0 \ \Rightarrow \vdash\ \Box\, f_0$
BiGen	$\vdash\ f_0 \ \Rightarrow \vdash\ \boxed{\scriptstyle i}\, f_0$

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

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

6 Tools

6.1 (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 Stable Version:

- 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.
  ```

To run the graphical interface you can either

- use the pre-compiled binary:

  anatempura_linux (Ubuntu 9.10)

  anatempura_solaris (Solaris 10u8)

  anatempura.exe (Windows XP)

- or use anatempura.tcl:

  you need to compile tempura, install Tcl/Tk (at least 8.5) and Expect. You can get Tcl/Tk from Tcl/Tk site and you can get Expect from the Expect homepage. A convenient way of installing these is using the ActiveTcl package which includes both. ActiveTcl is the complete, ready-to-install Tcl/Tk distribution for Windows, Mac OS X, Linux, Solaris, AIX and HP-UX. Newer versions of the ActiveTcl do not have the Expect package installed. But you can get Expect with the teacup command:

  > (For Windows you have to do this in a DOS shell.)
  > Type the following command:
  > teacup install Expect

  Tempura can be compiled using the Gnu C compiler under a Unix like operating system like Sunos 4.1.3, Solaris 2.5.1 (and higher), Linux etc. For Windows you need to install the MinGW (Minimalist GNU for Windows). For convenience the pre-compiled binary for Windows (Windows XP) tempura.exe, for Solaris (Solaris 10u8) tempura_solaris and for Linux (Ubuntu 9.10) are included.

Contact: Email `tempura@dmu.ac.uk` 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.

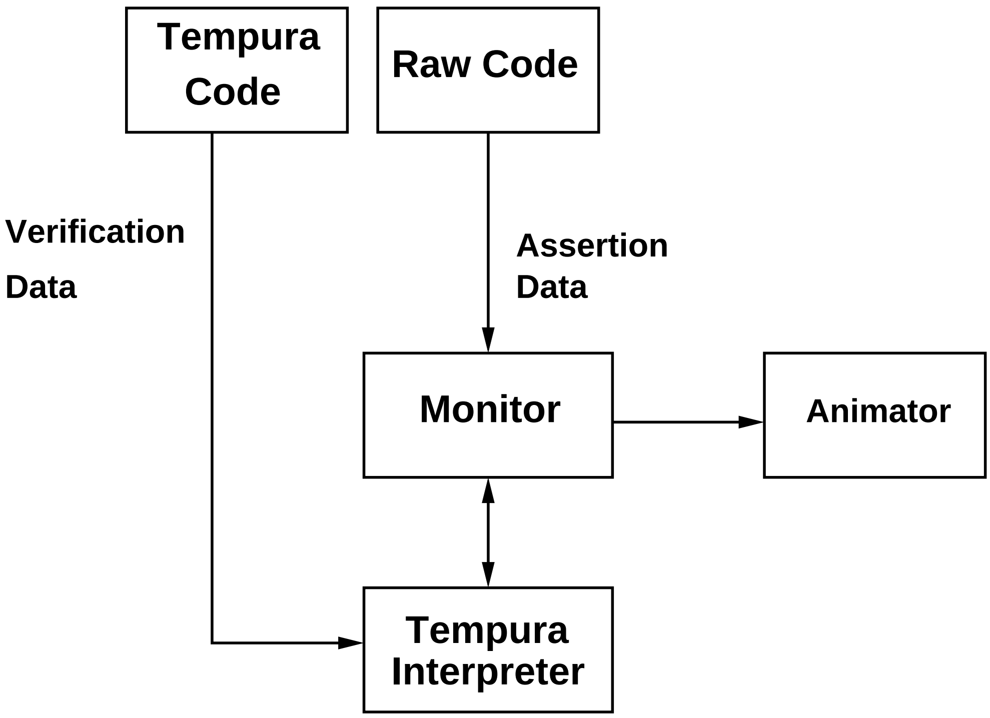

Figure 1: Overview of AnaTempura

Figure 2 shows the interface of AnaTempura.

Figure 3 shows a graphical snapshot of a simulation of the ep/3 microprocessor specified in Tempura.

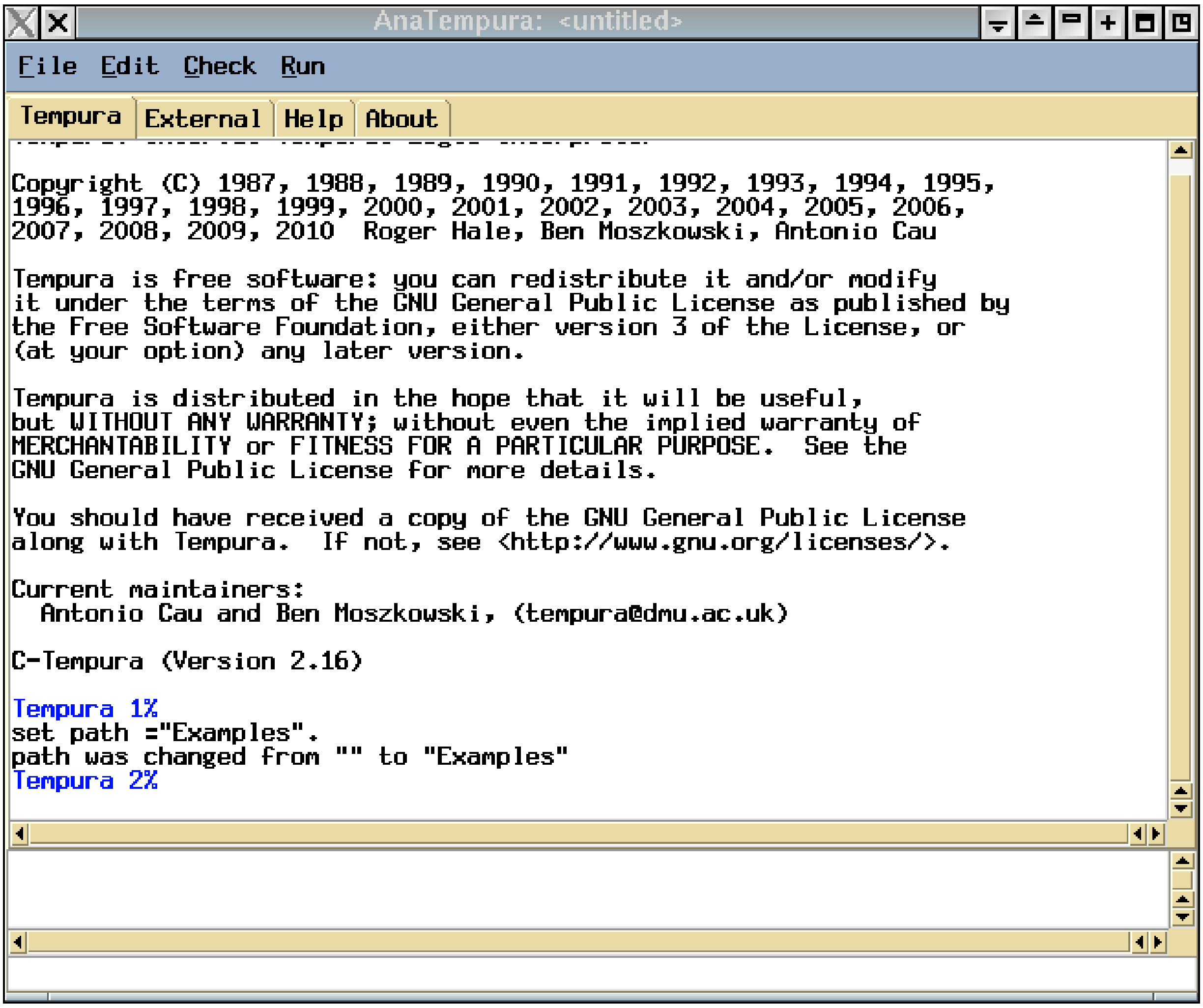

Figure 2: Interface of AnaTempura

6.2 ITL Theorem Prover based on Prover9

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
```

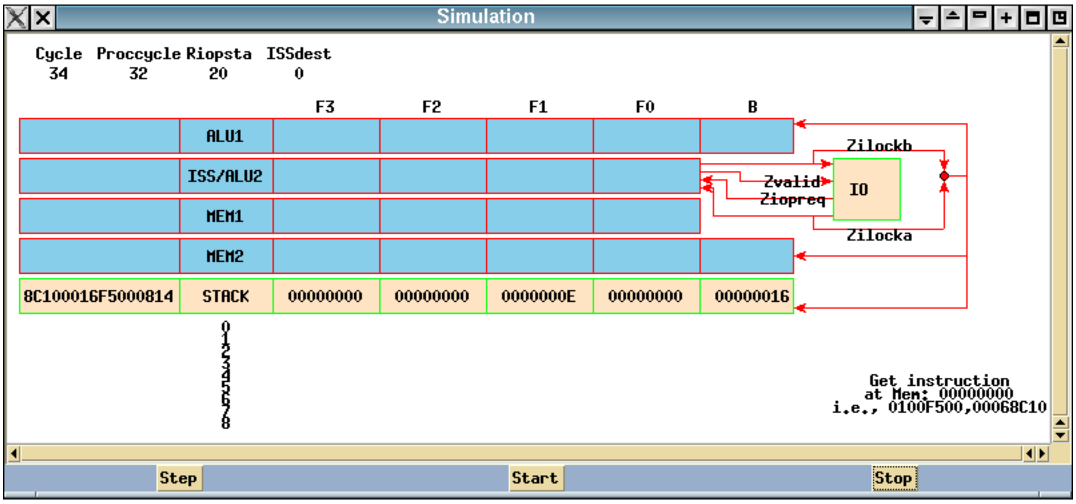

Figure 3: Graphical snapshot of simulation of the EP/3 microprocessor

```
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.

6.3 ITL Proof Checker based on PVS

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.

6.4 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.

7 Tempura Book

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. Any comments are welcome. Here is how to reach the author:

Dr. Ben Moszkowski
Software Technology Research Laboratory
Gateway House (GH4.55)
The Gateway
Leicester LE1 9BH
Great Britain
email: Ben's email
URL: Dr. Ben Moszkowski www page

8 STRL Publications using ITL

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

[2] B. Moszkowski. *Reasoning about Digital Circuits*. PhD thesis, Department of Computer Science, Stanford University, 1983. Technical report STAN–CS–83–970.

[3] B. Moszkowski and Z. Manna. Reasoning in Interval Temporal Logic. In Edmund Clarke and Dexter Kozen, editors, *Proceedings of the Workshop on Logics of Programs*, volume 164 of *LNCS*, pages 371–382, Pittsburgh, PA, June 1983. Springer Verlag.

[4] J. Halpern, Z. Manna, and B. Moszkowski. A hardware semantics based on temporal intervals. In J. Diaz, editor, *Proceedings of the 10-th International Colloquium on Automata, Languages and Programming*, volume 154 of *LNCS*, pages 278–291, Berlin, 1983. Springer Verlag.

[5] B. Moszkowski. Executing temporal logic programs. Technical Report 55, Computer Laboratory, University of Cambridge, 1984.

[6] B. Moszkowski. A temporal logic for multilevel reasoning about hardware. *IEEE Computer*, 18(2):10–19, 1985.

[7] B. Moszkowski. Executing temporal logic programs (preliminary version). In S. D. Brookes, A. W. Roscose, and G. Winskel, editors, *Seminary on Concurrency*, volume 197 of *LNCS*, pages 111–130, Berlin, 1985. Springer Verlag.

[8] B. Moszkowski. A temporal analysis of some concurrent systems. In B. T. Denvir, W. T .Harwood, M. I. Jackson, and M. J. Wray, editors, *The Analysis of Concurrent Systems*, volume 207 of *LNCS*, pages 359–364, Berlin, 1985. Springer Verlag.

[9] B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, Cambridge, England, 1986. online version Tempura Book.

[10] R. Hale. Temporal logic programming. In A. Galton, editor, *Temporal Logics and Their Applications*, pages 91–119. Academic Press, London, 1987.

[11] R. Hale and B. C. Moszkowski. Parallel programming in temporal logic. In J. W. de Bakker, A. J. Nijman, and Philip C. Treleaven, editors, *PARLE, Parallel Architectures and Languages Europe, Volume II: Parallel Languages*, volume 259 of *LNCS*, pages 277–296, Eindhoven, The Netherlands, June 1987. Springer Verlag.

[12] R. W. S. Hale. *Programming in Temporal Logic*. PhD thesis, Computer Laboratory, Cambridge University, Cambridge, England, October 1988. Appeared as technical report 173 in year 1989.

[13] R. Hale. Using temporal logic for prototyping: The design of a lift controller. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, volume 398 of *LNCS*, pages 375–408, Berlin, 1989. Springer Verlag. Proceedings (Altrincham, UK, April, 1987).

[14] B. Moszkowski. Some very compositional temporal properties. Technical Report 466, Dept. of Computing Science, University of Newcastle, December 1993.

[15] B. Moszkowski. Some very compositional temporal properties. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi*, volume A-56 of *IFIP Transactions*, pages 307–326. IFIP, Elsevier Science B.V. (North–Holland), 1994.

[16] R. W. S. Hale. Program compilation. In J. Bowen, editor, *Towards Verified Systems*, chapter 7, pages 131–130. Elsevier Science B.V. (North–Holland), Amsterdam, 1994.

[17] R. W. S. Hale and He Jifeng. A real-time programming language. In J. Bowen, editor, *Towards Verified Systems*, chapter 6, pages 115–130. Elsevier Science B.V. (North–Holland), Amsterdam, 1994.

[18] B. Moszkowski. Compositional reasoning about projected and infinite time. Technical Report EE/0495/M1, Dept. of Elec. and Elec. Eng., Univ. of Newcastle, UK, Newcastle upon Type, UK, April 1995. Some preliminary ideas presented at 2nd ProCoS-WG meeting, Oxford, 10–12 Jan 1995.

[19] B. 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)*, pages 238–245. IEEE Computer Society Press, 1995. STRL Publication 1995-2.

[20] B. Moszkowski. Embedding imperative constructs in Interval Temporal Logic. Technical Report Internal memorandum EE/0895/M1, Dept. of Elec. and Elec. Eng., Univ. of Newcastle, UK, Newcastle upon Type, UK, August 1995. Presented at Third ProCoS WG meeting at Vedbaek, Denmark, 21-23 August 1995.

[21] B. Moszkowski. Using temporal fixpoints to compositionally reason about liveness. In He Jifeng, John Cooke, and Peter Wallis, editors, *BCS-FACS 7th Refinement Workshop*, electronic Workshops in Computing, London, 1996. BCS-FACS, Springer-Verlag and British Computer Society. STRL Publication 1996-4.

[22] A. Cau, H. Zedan, N. Coleman, and B. Moszkowski. Using ITL and TEMPURA for large scale specification and simulation. In *Proc. of the 4th Euromicro Workshop on Parallel and Distributed Processing*, pages 493–500. IEEE Computer Society Press, 1996. STRL Publication 1996-1.

[23] A. Cau and B. Moszkowski. Using PVS for Interval Temporal Logic Proofs. Part 1: The syntactic and semantic encoding. Technical monograph 14, SERCentre, De Montfort University, Leicester, 1996. STRL Technical Monograph 7.

[24] B. Moszkowski. The programming language Tempura. *Journal of Symbolic Computation*, 22(5/6):730–733, November/December 1996. STRL Publication 1996-3.

[25] A. Cau and H. Zedan. Refining Interval Temporal Logic specifications. In M. Bertran and T. Rus, editors, *Transformation-Based Reactive Systems Development*, volume 1231 of *LNCS*, pages 79–94. AMAST, Springer Verlag, 1997. STRL Publication 1997-2.

[26] X. Li, A. Cau, B. Moszkowski, and N. Coleman. Proving the correctness of the interlock mechanism in processor design. In Hon F. Li and David K. Probst, editors, *Advances in Hardware Design and Verification*, pages 2–22, London, 1997. IFIP/Chapman and Hall. STRL Publication 1997-3.

[27] A. Cau, C. Czarnecki, and H. Zedan. Designing a Provably Correct Robot Control System using a 'Lean' Formal Method. In Anders P. Ravn and Hans Rischel, editors, *Proceedings 5th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'98)*, volume 1486 of *LNCS*, pages 123–132. Springer Verlag, 1998. STRL Publication 1998-2.

[28] B. Moszkowski. Compositional reasoning using Interval Temporal Logic and Tempura. In Willem-Paul de Roever, Hans Langmaack, and Amir Pnueli, editors, *Compositionality: The Significant Difference*, volume 1536 of *LNCS*, pages 439–464, Berlin, 1998. Springer Verlag. STRL Publication 1998-1.

[29] S. Zhou, H. Zedan, and A. Cau. A framework for analysing the effect of 'change' in legacy code. In *IEEE Proc. of ICSM'99*. IEEE, 1999. STRL Publication 1999-1.

[30] H. Zedan, A. Cau, Z. Chen, and H. Yang. Atom: An object-based formal method for real-time systems. *Annals of Software Engineering*, 7:235–256, 1999. STRL Publication 1999-3.

[31] Z. Chen, H. Zedan, A. Cau, and H. Yang. A wide-spectrum language for object-based development of real-time systems. *Journal of Information Sciences*, 118:15–35, 1999. STRL Publication 1999-4.

[32] H. Zedan, A. Cau, and B.C. Moszkowski. Compositional modelling: The formal perspective. In David Bustard, editor, *Proc. of Workshop on Systems Modelling for Business Process Improvement*, pages 333–354. Artech House, 2000. STRL Publication 2000-1.

[33] J. Dimitrov. Compositional reasoning about events in interval temporal logic. In *Proc. of The Fifth International Conference on Computer Science and Informatics*, 2000. STRL Publication 2000-3.

[34] A.C. Rao, A. Cau, and H. Zedan. Visualization of interval temporal logic. In *Proc. of the Fifth International Conference on Computer Science and Informatics*, 2000. STRL Publication 2000-4.

[35] H. Zedan, A. Cau, and S. Zhou. A calculus for evolution. In *Proc. of the Fifth International Conference on Computer Science and Informatics*, 2000. STRL Publication 2000-6.

[36] B. 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)*, pages 242–251. IEEE Computer Society Press, June 2000. STRL Publication 2000-21.

[37] B. Moszkowski. An automata-theoretic completeness proof for Interval Temporal Logic (extended abstract). In Ugo Montanari, José Rolim, and Emo Welzl, editors, *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP 2000)*, volume 1853 of *LNCS*, pages 223–234, Geneva, Switzerland, July 2000. Springer Verlag. STRL Publication 2000-20.

[38] A. Cau and H. Zedan. *Systems Engineering for Business Process Change, Peter Henderson, editor*, chapter The Systematic Construction of Information Systems, pages 264–278. Springer Verlag, 2000. STRL Publication 2000-8.

[39] H. Yang, X. Liu, and H. Zedan. Abstraction: A key notion for reverse engineering in a reengineering approach. *Journal of Software Maintenance: Research and Practice*, 12(4):197–228, 2000.

[40] R. Hale. Using itl for codesign. In *Proceedings of the Verification Workshop part of the International Joint Conference on Automated Reasoning IJCAR'2001*, 2001. STRL Publication 2001-12.

[41] H. Zedan and A. 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*, pages 194–200. IEEE Computer Society, 2001.

[42] H. Zedan, S. Zhou, N. Sampat, X. Chen, A. Cau, and H. Yang. K-mediator: Towards evolving information systems. In *ICSM*, pages 520–527, 2001. STRL Publication 2001-11.

[43] A. Cau, R. Hale, J. Dimitrov, H. Zedan, B. Moszkowski, M. Manjunathaiah, and M. Spivey. A compositional framework for hardware/software co-design. *Design Automation for Embedded Systems*, 6(4):367–399, 2002. STRL Publication 2002-4.

[44] Jordan Dimitrov. *Formal Compositional Design of Mixed Hardware/Software Systems with semantics of Verilog HDL*. PhD thesis, Software Technology Research Laboratory, De Montfort University, 2002. STRL Thesis 10.

[45] B. Moszkowski. A hierarchical completeness proof for propositional temporal logic. In Nachum Dershowitz, editor, *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), volume 2772 of *LNCS*, pages 480–523, Berlin, 2003. Springer Verlag.

[46] B. 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)*)*, pages 41–65, Vienna, August 2003.

[47] M. Solanki, A. Cau, and H. Zedan. Introducing compositionality in webservice descriptions. In *proceedings of the 3rd International Anwire Workshop on Adaptable Service Provision*. Springer-Verlag, 2003.

[48] F. Siewe, A. Cau, and H. Zedan. A compositional framework for access control policies enforcement. In *proceedings of the ACM workshop on Formal Methods in Security Engineering: From Specifications to Code*, 2003. STRL Publication 2003-16.

[49] M. Solanki, A. Cau, and H. Zedan. Augmenting semantic web service descriptions with compositional specification. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*, pages 544–552. ACM, 2004. STRL Publication 2004-7.

[50] Ben Moszkowski. A hierarchical completeness proof for Propositional Interval Temporal Logic with finite time. *Journal of Applied Non-Classical Logics*, 14(1–2):55–104, 2004. Special issue on Interval Temporal Logics and Duration Calculi.

[51] Ben Moszkowski. A hierarchical completeness proof for propositional temporal logic. In Nachum Dershowitz, editor, *Verification: Theory and Practice: Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, volume 2772 of *LNCS*, pages 480–523. Springer Verlag, Heidelberg, 2004.

[52] M. Solanki, A. Cau, and H. 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*, pages 14–20. IEEE Computer Society, 2004. STRL Publication 2004-8.

[53] S. Zhou, H. Zedan, and A. Cau. Run-time analysis of time-critical systems. *Journal of System Architecture*, 51(5):331–345, 2005. STRL Publication 2005-1.

[54] François Siewe. *A Compositional Framework for the Development of Secure Access Control Systems*. PhD thesis, Software Technology Research Laboratory, Department of Computer Science and Engineering, De Montfort University, Leicester, 2005. STRL Thesis 20.

[55] Monika Solanki. *A Compositional Framework for the Specification, Verification and Runtime Validation of Reactive Web Services*. PhD thesis, Software Technology Research Laboratory, Department of Computer Science and Engineering, De Montfort University, Leicester, 2005. STRL Thesis 21.

[56] Helge Janicke, François Siewe, Kevin Jones, Antonio Cau, and Hussein Zedan. Analysis and Run-time Verification of Dynamic Security Policies. In Robert Ghanea-Hercock and Mark Greaves and Nick Jennings and Simon Thompson, editor, *In Proceedings of The First Workshop on Defence Applications for Multi-Agent Systems (DAMAS'05)*, volume 3890 of *Lecture Notes in Computer Science*, pages 92–103, Utrecht, The Netherlands, July 2005. Springer. STRL Publication 2005-3.

[57] F. Siewe, H. Janicke, and K. Jones. Dynamic access control policies and web-service composition. In *Proceedings of the 1st Young Researchers Workshop on Service Oriented Computing (YR-SOC 05)*, 2005. STRL Publication 2005-4.

[58] Monika Solanki, Antonio Cau, and Hussein Zedan. *Semantic Web Services, Processes and Applications*, chapter Temporal Specifications for Semantic Web services. Springer, 2005. Link to publication on Springer website.

[59] 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*, volume 140, 2005. STRL Publication 2005-13.

[60] Ben Moszkowski. A hierarchical analysis of propositional temporal logic based on intervals. In Sergei Artemov, Howard Barringer, Artur S. d'Avila Garcez, Luis C. Lamb, and John Woods, editors, *We Will Show Them: Essays in Honour of Dov Gabbay*, volume 2, pages 371–440. College Publications (formerly KCL Publications), King's College, London, 2005.

[61] Helge Janicke, Antonio Cau, François Siewe, Hussein Zedan, and Kevin Jones. A Compositional Event & Time-based Policy Model. In *Procceedings of POLICY2006, London, Ontario, Canada*, pages 173–182, London, Ontario Canada, June 2006. IEEE Computer Society. STRL Publication 2006-3.

[62] 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*, Edinburgh, Scotland, 2006. ACM. STRL Publication 2006-4.

[63] Helge T. Janicke. *The Development of Secure Multi-Agent Systems*. PhD thesis, De Montfort University, February 2007. STRL Thesis 31.

[64] Ben Moszkowski. Using Temporal Logic to Analyse Temporal Logic: A Hierarchical Approach Based on Intervals. *Journal of Logic and Computation*, 17(2):333–409, 2007. STRL Publication 2007-8.

[65] Helge Janicke, Antonio Cau, Francois Siewe, and Hussein Zedan. Deriving Enforcement Mechanisms from Policies. In *In Proceedings of the 8th IEEE international Workshop on Policies for Distributed Systems*, 2007. STRL Publication 2007-6.

[66] Helge Janicke, Antonio Cau, Francois Siewe, and Hussein Zedan. A note on the formalisation of UCON. In *In Proceedings of SACMAT07*, 2007. STRL Publication 2007-7.

[67] 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. STRL Publication 2007-9.

[68] Helge Janicke, Antonio Cau, Francois Siewe, and Hussein Zedan. Concurrent Enforcement of Usage Control Policies. In *In Proceedings of the 9th IEEE international Workshop on Policies for Distributed Systems*, 2008. STRL Publication 2008-4.

9 Non-STRL Publications using ITL

[69] J. Bowen, editor. *Towards Verified Systems*. Elsevier Science B.V. (North–Holland), Amsterdam, 1994. About the SAFEMOS project.

[70] J. Bowen, He Jifeng, Roger Hale, and John Herbert. Towards verified systems: The safemos project. In C. Mitchell and V. Stavridou, editors, *The Mathematics of Dependable Systems*, The Institute of Mathematics and its Applications Conference Series, Oxford, 1995. Oxford University Press.

[71] H. Bowman, H. Cameron, P. King, and S. Thompson. Mexitl: Multimedia in Executable Interval Temporal Logic. Technical Report 3-97, Computing Laboratory, University of Kent at Canterbury, May 1997.

[72] H. Bowman, H. Cameron, P. King, and S. Thompson. Specification and prototyping of structured multimedia documents using Interval Temporal Logic. In *International Conference on Temporal Logic*, Applied Logic Series. Kluwer Academic Publishers, July 1997.

[73] H. Bowman and S. J. Thompson. A tableaux method for Interval Temporal Logic with projection. In *TABLEAUX'98, International Conference on Analytic Tableaux and Related Methods*, volume 1397 of *Lecture Notes in AI*, pages 108–123. Springer Verlag, May 1998.

[74] XiaoShan Li. Formal semantics of Verilog HDL. In *CDROM: Proceedings of the Second Forum on Design Languages (FDL '99)*, pages 127–136, Lyon, France, Aug. 30 – Sep. 3 1999. ECSI Verlag, Gières. August 30 - September 3, 1999, Ecole Normale Supérieure de Lyon, Lyon, France.

[75] 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*, pages 197–204, Los Angeles, California, May 1999. IEEE Computer Society.

[76] H. Bowman and S. J. Thompson. A complete axiomatization of Interval Temporal Logic with projection. Technical Report 6-00, Computing Laboratory, University of Kent, Canterbury, Great Britain, January 2000.

[77] H. Bowman and S. J. Thompson. A decision procedure and complete axiomatization of finite Interval Temporal Logic with projection. *Journal of Logic and Computation*, 13(2):195–239, April 2003.

[78] S. M. Brien. A relational calculus of intervals. Master's thesis, Programming Research Group, Oxford University, 12 1990.

[79] R. Buessow and W. Grieskamp. Combining Z and temporal interval logics for the formalization of properties and behaviors of embedded systems. In *Advances in Computing Science - ASIAN '97*, volume 1345 of *LNCS*, pages 46–56, 1997.

[80] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.

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

[82] H. Cameron, P. King, H. Bowman, and S. Thompson. Synchronization in multimedia documents. In R.D. Hersch, J. Andr/'e, and H. Brown, editors, *Electronic Publishing, Artistic Imaging, and Digital Typography: 7th International Conference on Electronic Publishing (EP'98)*, volume 1375 of *LNCS*, Berlin, 1998. Springer Verlag.

[83] L. K. Dillon, G. Kutty, L. E. Moser, P. M. Melliar-Smith, and Y. S. Ramakrishna. A graphical interval logic for specifying concurrent systems. *ACM Transactions on Software Engineering and Methodology*, 3(2):131–165, April 1994.

[84] R. D. Dowsing and R. Elliot. A higher level of behaviorial specification: An example in interval temporal logic. In *Microprocessing and Microprogramming (Proceedings of EUROMICRO '91)*, volume 32, pages 517–524, 1991.

[85] R. Dowsing, E. Elliot, and I. Marshall. Automated technique for high-level circuit synthesis from temporal logic specifications. *IEE Proceedings–Computers and Digital Techniques*, 141(3):145–152, May 1994.

[86] Z. 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. Technical report 556.

[87] Z. Duan, M. Koutny, and C. Holt. Projection in temporal logic programming. In Frank Pfenning, editor, *Proc. of Logic Programming and Automated Reasoning (LPAR '94)*, volume 822 of *LNCS*, pages 333–344, Berlin, 1994. Springer Verlag.

[88] Z. Duan and M. Koutny. A framed temporal logic programming language. *Journal of Computer Science and Technology*, 19(3):341–351, 2004.

[89] B. Dutertre. On first order interval temporal logic. Technical Report CSD-TR-94-3, Dept. of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England, 1994.

[90] B. Dutertre. Complete proof systems for first order interval temporal logic. In *Proc. of the 10th Annual IEEE Symposium on Logic in Computer Science*, pages 36–43, Los Alamitos, Calif., USA, June 1995. IEEE Computer Society Press.

[91] R. Elliot. An exercise in formally based circuit synthesis from a behavioural specification in Interval Temporal Logic. In *Proc. EUROMICRO '94*, 1994.

[92] M. Fujita, S. Kono, and H. Tanaka. Aid to Hierarchical and Structured Logic Design Using Temporal Logic and Prolog. In *Prodeedings.Pt.E*, pages 283–294. IEE, 1986.

[93] M. Fujita, S. Kono, H. Tanaka, and T. Moto-Oka. Tokio: Logic programming language based on temporal logic and its compilation to Prolog. In *Proc. 3rd Int'l. Conf. on Logic Programming*, volume 225 of *LNCS*, pages 695–709, Berlin, 1986. Springer-Verlag.

[94] M. Fujita, M. Ishisone, H. Nakamura, H. Tanaka, and T. Mto-oka. Using the temporal logic programming language tokio for algorithm description and automatic cmos gate array synthesis. In Eiiti Wada, editor, *Logic Programming '85*, volume 221 of *LNCS*, pages 246–255, Berlin, 1986. Springer Verlag.

[95] M. Fujita and S. Kono. Synthesis of controllers from Interval Temporal Logic specification. *International Workshop on Logic Synthesis*, 1993.

[96] M. Fujita and S. 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.

[97] R. Gomez and H. Bowman. PITL2MONA: Implementing a Decision Procedure for Propositional Interval Temporal Logic. *Journal of Applied Non-Classical Logics*, 14(1–2):105–148, 2004. Special issue on Interval Temporal Logics and Duration Calculi. V. Goranko and A. Montanari guest eds.

[98] V. Goranko and A. Montanari, editors. *Special issue on Interval Temporal Logics and Duration Calculi*, volume 14 of *Journal of Applied Non-Classical Logics*. Lavoisier, 2004.

[99] W. Grieskamp and M. Lepper. Encoding temporal logics in executable Z: A case study for the ZETA system. In *Logic for Programming and Automated Reasoning: Proc. 7th International Conference (LPAR 2000)*, volume 1955 of *LNCS*, pages 43–53, Reunion Island, France, November 2000. Springer Verlag.

[100] D. P. Guelev and D. Van Hung. On the completeness and decidability of duration calculus with iteration. To appear in *Theoretical Computer Science*.

[101] K. Hamaguchi, H. Hiraishi, and S. Yajima. Infinity-regular temporal logic and its model checking problem. *Theoretical Computer Science*, 103(2):191–204, 1992.

[102] M. R. Hansen. Model-checking discrete duration calculus. *Formal Aspects of Computing*, 6(6A):826–845, 1994.

[103] E. C. R. Hehner. Abstractions of time. In A. W. Roscoe, editor, *A Classical Mind*, chapter 12. Prentice-Hall Int'l., London, 1994.

[104] T. A. Henzinger, Z. Manna, and A. Pnueli. Towards refining temporal specifications into hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, volume 736 of *LNCS*, pages 60–76. Springer-Verlag, 1993.

[105] H. Hiraishi, K. Hamaguchi, H. Fujii, and S. Yajima. Regular temporal logic expressively equivalent to finite automata and its application to logic design verification. *Journal of Information Processing*, 15(1):129–138, 1992.

[106] H. JiFeng and J. Bowen. Time interval semantics and implementation of a real-time programming language. In *Proceedings of the 4-th Euromicro Workshop on Real-Time Systems*, pages 110–115. IEEE Computer Society Press, 1992.

[107] A. Kapur. *Interval and Point-Based Approaches to Hybrid System Verification*. PhD thesis, Dept. of Computer Science, Stanford University, September 1997. Technical report CS-TR-97-1594.

[108] A. Kapur, T. A. Henzinger, Z. Manna, and A. Pnueli. Proving safety properties of hybrid systems. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT 94: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 863, pages 431–454. Springer-Verlag, 1994. Uses Hybrid temporal logic.

[109] D. Kilis, A. C. Esterline, and J. R. Slagle. Specification and verification of network protocols using executable temporal logic. In *Proceedings of IFIP Congress '89*, pages 845–850, Amsterdam, 1989. North Holland Publishing Co.

[110] S. Kono, T. Aoyagi, M. Fujita, and H. Tanaka. Implementation of Temporal Logic Programming Language Tokio. In *Logic Programming Conference '85*, pages 138–147. ICOT, 1985.

[111] S. Kono, T. Aoyagi, M. Fujita, and H. Tanaka. Implementation of Temporal Logic Programming Language Tokio. In *Logic Programming '85*, volume LNCS-221. Springer-Verlag, 1985. Lecture Notes in Computer Science.

[112] S. Kono, T. Aoyagi, M. Fujita, and H. Tanaka. Verification of Temporal Logic Programming Language Tokio. In *Logic Programming Conference '86*, 1986. (in Japanese).

[113] S. Kono. Automatic Verification of Interval Temporal Logic. In *8th British Colloquium For Theoretical Computer Science*, 1992. version: 24-04-1994.

[114] S. Kono. A Combination of Clausal and Non Clausal Temporal Logic Program. *IJCAI-93 Workshop on Executable Modal and Temporal Logics*, 1993.

[115] S. Kono. A combination of clausal and non-clausal temporal logic programs. In Michael Fisher and Richard Owens, editors, *Executable Modal and Temporal Logics*, volume 897 of *Lecture Notes in Artificial Intelligence*, pages 40–57, Cambery, France, February 1995. Springer Verlag.

[116] M. Leeser. Reasoning about the function and timing of integrated circuits with Interval Temporal Logic. *IEEE Transactions on Computer-Aided Design*, 8(12):1233–1246, December 1989.

[117] R. 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.

[118] K. Lodaya. Sharpening the undecidability of Interval Temporal Logic. In He Jifeng and Masahiko Sato, editors, *Advances in Computing Science - ASIAN 2000: Proc. of Sixth Asian Computing Science Conference*, volume 1961 of *LNCS*, Penang, Malaysia, November 2000. Springer Verlag.

[119] R. Mattolini and P. Nesi. An interval logic for real-time system specification. *Transactions on Software Engineering, IEEE*, 27(3):208–227, March 2001.

[120] M. J. Morley. Semantics of temporal ***e***. In T. F. Melham and F. G. Moller, editors, Banff'99 *Higher Order Workshop: Formal Methods in Computation, Ullapool, Scotland, 9–11 Sept. 1999*, pages 138–142. University of Glasgow, Department of Computing Science Technical Report, 1999.

[121] M. Müller-Olm. A modal fixpoint logic with chop. In Christoph Meinel and Sophie Tison, editors, *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of *lncs*, pages 510–520, Trier, Germany, MAR 1999. springer.

[122] B. Paech. Gentzen-systems for propositional temporal logics. In E. Börger, H. Kleine Büning, and M. M. Richter, editors, *Proceedings of the 2nd Workshop on Computer Science Logic, Duisburg (FRG)*, volume 385 of *LNCS*, pages 240–253. Springer Verlag, October 1988.

[123] Y. S. Ramakrishna, L. K. Dillon, L. E. Moser, P. M. Melliar-Smith, and G. Kutty. An automata-theoretic decision procedure for future interval logic. In *Proc. 12th FST& TCS*, volume 652 of *LNCS*, pages 51–67. Springer Verlag, December 1992.

[124] Y. S. Ramakrishna, P. M. Melliar-Smith, L. E. Moser, L. K. Dillon, and G. Kutty. Interval logics and their decision procedures. Part I: An interval logic. *Theoretical Computer Science*, 166(1–2):1–47, 20 October 1996. Fundamental study.

[125] Y. S. Ramakrishna, P. M. Melliar-Smith, L. E. Moser, L. K. Dillon, and G. Kutty. Interval logics and their decision procedures. Part II: A real-time interval logic. *Theoretical Computer Science*, 170(1–2):1–46, 15 December 1996. Fundamental study.

[126] T. M. Rasmussen. Signed interval logic. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic, 13th International Workshop, CSL '99, 8th Annual Conference of the EACSL*, volume 1683 of *LNCS*, pages 157–171, Berlin, 1999.

[127] R. Rosner and A. Pnueli. A choppy logic. In *First Annual IEEE Symposium on Logic in Computer Science*, pages 306–313. IEEE Computer Society Press, June 1986.

[128] A. R. Ruddle. Formal methods in the specification of real-time, safety-critical control systems. In J. P. Bowen and J. E. Nicholls, editors, *Z User Workshop, London 1992*, Workshops in Computing, pages 131–146. Springer Verlag, 1993.

[129] R. L. Schwartz, P. M. Melliar-Smith, and F. 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*, pages 173–186, 1983.

[130] Wang Hanpin and Xu Qiwen. Temporal logics over infinite intervals. Technical Report 158, UNU/IIST, Macau, 1999.

[131] Wang Jianzhong, Xu Qiwen, and Ma Huadong. Modelling and verification of a network player system with DCValid. In T. H. Tse and T. Y. Chen, editors, *Proc. of the First Asia-Pacific Conference on Quality Software (APAQS 2OOO)*, pages 44–49, Hong Kong, October 2000. IEEE Computer Society Press.