In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP 2000)*, editors: Ugo Montanari, José Rolim and Emo Welzl. Geneva, Switzerland, July 9–15, 2000. *Lecture Notes in Computer Science*, vol. 1853, Springer-Verlag, pages 223–234.

An Automata-Theoretic Completeness Proof for Interval Temporal Logic (Extended Abstract)

B. C. Moszkowski*

Software Technology Research Lab.
SERCentre
Hawthorn Building
De Montfort University
The Gateway
Leicester LE1 9BH
Great Britain

benm@dmu.ac.uk
http://www.cms.dmu.ac.uk/~benm

Abstract. *Interval Temporal Logic* (ITL) is a formalism for reasoning about time periods. To date no one has proved completeness of a relatively simple ITL deductive system supporting infinite time and permitting infinite sequential iteration comparable to ω-regular expressions. We have developed a complete axiomatization for such a version of quantified ITL over finite domains and can show completeness by representing finite-state automata in ITL and then translating ITL formulas into them. Here we limit ourselves to finite time. The full paper (and another conference paper [15]) extends the approach to infinite time.

1 Introduction

Interval Temporal Logic (ITL) [6, 9, 10] is a temporal logic which includes a basic construct for the sequential composition of two formulas as well as an analog of Kleene star. Within ITL, one can express both finite-state automata and regular expressions. Its notation makes it suitable for logic-based modular reasoning involving periods of time, refinement [2], sequential composition using assumptions and commitments based on fixpoints of various temporal operators [12, 14] and for executable specifications [11]. Various imperative programming constructs are expressible in ITL and projection between time granularities is available (but not considered here). Zhou Chaochen, Hoare and Ravn [21] have developed an ITL extension called *Duration Calculus* for hybrid systems. Several researchers have looked at ITL decision procedures and axioms systems. However, previously there was no known complete axiom system for a version of ITL over both finite and ω-words having no artificial restrictions on interval constructs. We present a natural and complete axiomatization for a subset of quantified ITL

* Part of the research described here has been kindly supported by EPSRC research grant GR/K25922.

for finite time in which variables are limited to finite domains. The completeness proof describes an ITL decision procedure within ITL itself. In the full paper (and another conference paper [15]) infinite time is considered.

We build on the work of Siefkes [19] who proved the completeness of an axiomatization of the *Second-Order Theory of Successor* (S1S) and Kesten and Pnueli [7] who did likewise for *Quantified Propositional Temporal Logic* (QPTL) with past-time operators. Our approach follows Kesten and Pnueli's technique of reducing temporal formulas to finite-state automata as part of a decision procedure. These automata are themselves represented and manipulated in ITL. The ITL axiom system and completeness proof vary substantially from Kesten and Pnueli's. This reflects differences between conventional temporal logics and an interval-based one.

Our results offer a natural yet complete axiom system for a nontrivial subset of ITL and show how ITL can itself encode the decision procedure. Automata are more compositional than temporal logic tableaux which analyze several formulas in parallel. There is no need for Fischer-Ladner closures [4], first developed for a propositional version of Pratt's Dynamic Logic [17]. We also show that the ITL axiom system provides a logical framework for both finite-state automata and regular expressions.

2 Related Work

Let us now discuss other work on ITL axiom systems. Rosner and Pnueli [18] investigate an axiom system for quantifier-free propositional ITL (PITL) with finite and ω-intervals. The ITL subset also includes the *until* operator but not the operator *chop-star* which is like Kleene-star for regular expressions. A tableaux-based decision procedure underlies the completeness proof and uses an adaptation of the Fischer-Ladner closures. One inference rule requires detailed meta-reasoning about tableaux transitions.

Paech [16] investigates PITL with ω-intervals but includes a *chop-star* limited, like Kleene-star, to finitely many iterations and another operator *unless*. She gives a complete proof system with some nonconventional axioms for formulas already in a form like regular expressions and possibly involving complex meta-reasoning. A generalization of Fischer-Ladner closures is used.

Dutertre [3] gives two complete proof systems for first-order ITL without *chop-star* for finite time. One has a possible-worlds semantics of time and the other uses arbitrary linear orderings of states. Neither is complete for standard discrete-time intervals. Wang Hanpin and Xu Qiwen [20] generalize this to infinite time. Moszkowski [12] presents propositional and first-order ITL axiom systems for finite intervals. The former is claimed to be complete but only an outline of a proof is given. Axioms for temporal projection are given in [13]. Bowman and Thompson [1] have recently developed a tableaux-based completeness proof for an axiomatization of ITL with projection and finite time.

3 Overview of Interval Temporal Logic

We now briefly describe ITL for finite time. More details are in [6, 8–12, 14]. Basic ITL uses discrete, linear time. An interval σ has a length $|\sigma| \geq 0$ and a finite, nonempty

sequence of $|\sigma|+1$ states $\sigma_0, \ldots, \sigma_{|\sigma|}$. A state σ_i maps a variable such as A to a value $\sigma_i(A)$. Lower-case *static* variables $a, b, \ldots$ do not vary over time.

Here are permitted constructs using variable v, terms t and t' and formulas P and Q:

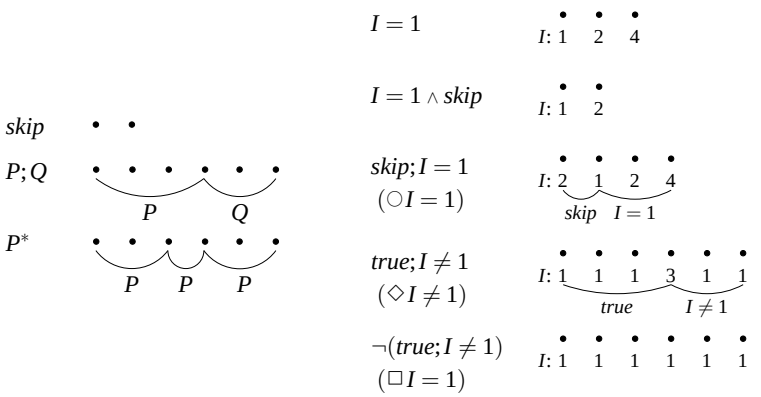
Terms: v (for numerical v), 0, 1, 2, ... (natural numbers), *if* P *then* t *else* t'
Formulas: v (for boolean v), $t = t'$, $\forall v.P$, $\neg P$, $P \wedge Q$, *skip*, $P;Q$, P^*

A variable v's values in an interval range over the finite, nonempty set $domain(v)$ which here is either $\{false, true\}$ or some initial subsequence of the natural numbers. Finite data domains ensure we have a decision procedure for our completeness result. We can readily extend *domain* to all ITL constructs. As in several temporal logics, the formula $I = 2$ is true on σ iff I's value in σ_0 equals 2.

There are three primitive temporal operators:

$$skip \qquad P;Q \ \ (chop) \qquad P^* \ \ (chop\text{-}star) \ ,$$

where P and Q are themselves formulas. The formula *skip* is true on a two-state interval. A formula $P;Q$ is true on σ iff σ can be chopped into two subintervals sharing a state σ_k for some $k \leq |\sigma|$ with P true on $\sigma_0 \ldots \sigma_k$ and Q true on $\sigma_k \ldots \sigma_{|\sigma|}$. Thus the formula $skip;I = J$ is true on σ iff σ has at least two states and $I = J$ is true in σ_1. A formula P^* is true on σ iff σ can be chopped into zero or more parts with P true on each. Any formula P^* (including $false^*$) is true on a one-state interval (see §3.2). Figure 1a pictorially illustrates the semantics of *skip*, *chop*, and *chop-star*. Some simple ITL formulas together with intervals which satisfy them are shown in Fig. 1b.



(a) Informal semantics (b) Some examples

Fig. 1. Informal ITL semantics and examples

For natural numbers i, j with $i \leq j \leq |\sigma|$, let $\sigma_{i:j}$ denotes the subinterval of length $j - i$ (i.e., $j - i + 1$ states) with starting state σ_i and final state σ_j. Below is the syntax

and semantics of the basic ITL constructs used here. We denote the semantics of a term t and formula P on interval σ as $\mathcal{M}_\sigma[\![t]\!]$ and $\mathcal{M}_\sigma[\![P]\!]$.

3.1 Semantics of Terms

- Numerical static or state variable: $\mathcal{M}_\sigma[\![v]\!]=\sigma_0(v)$.
  A numerical variable's value for an interval σ is the value in σ's initial state σ_0.
- Numerical constant: $\mathcal{M}_\sigma[\![c]\!] = c$.
- Conditional term: $\mathcal{M}_\sigma[\![\mathit{if}\ P\ \mathit{then}\ t\ \mathit{else}\ t']\!] = \begin{cases} \mathcal{M}_\sigma[\![t]\!], \text{ if } \mathcal{M}_\sigma[\![P]\!] = \mathit{true} \\ \mathcal{M}_\sigma[\![t']\!], \text{ otherwise.} \end{cases}$

3.2 Semantics of Formulas

- Boolean static or state variable: $\mathcal{M}_\sigma[\![v]\!] = \sigma_0(v)$.
  A boolean variable's value for an interval σ is the value in σ's initial state σ_0.
- Equality: $\mathcal{M}_\sigma[\![t = t']\!] = \mathit{true}$ iff $\mathcal{M}_\sigma[\![t]\!] = \mathcal{M}_\sigma[\![t']\!]$.
- Negation: $\mathcal{M}_\sigma[\![\neg P]\!] = \mathit{true}$ iff $\mathcal{M}_\sigma[\![P]\!] = \mathit{false}$.
- Conjunction: $\mathcal{M}_\sigma[\![P \wedge Q]\!] = \mathit{true}$ iff $\mathcal{M}_\sigma[\![P]\!] = \mathcal{M}_\sigma[\![Q]\!] = \mathit{true}$.
- Universal quantification: $\mathcal{M}_\sigma[\![\forall v.\, P]\!] = \mathit{true}$ iff $\mathcal{M}_{\sigma'}[\![P]\!] = \mathit{true}$,
  for every interval σ' identical to σ except possibly for variable v's behavior.
- Unit interval: $\mathcal{M}_\sigma[\![\mathit{skip}]\!] = \mathit{true}$ iff $|\sigma| = 1$.
- Chop: $\mathcal{M}_\sigma[\![P;Q]\!] = \mathit{true}$ iff $\mathcal{M}_{\sigma'}[\![P]\!] = \mathit{true}$ and $\mathcal{M}_{\sigma''}[\![Q]\!] = \mathit{true}$,
  where $\sigma' = \sigma_{0:i}$ and $\sigma'' = \sigma_{i:|\sigma|}$ for some $i \le |\sigma|$. Intervals σ' and σ'' share state σ_i.
- Chop-star: $\mathcal{M}_\sigma[\![P^*]\!] = \mathit{true}$ iff $\mathcal{M}_{\sigma_{l_i:l_{i+1}}}[\![P]\!] = \mathit{true}$, for each $i : 0 \le i < n$,
  for some $n \ge 0$ and finite sequence of natural numbers $l_0 \le l_1 \le \cdots \le l_n$ where $l_0 = 0$ and $l_n = |\sigma|$. Every one-state interval satisfies P^* since we can trivially choose $n = 0$.

If a formula P is true on an interval σ, then σ *satisfies* P, denoted $\sigma \models P$. A formula P satisfied by all intervals is *valid*, denoted $\models P$.

We view formulas as boolean terms to avoid, for example, distinct theorems for quantified boolean and numerical variables. Hence $P = Q$ and $P \equiv Q$ are identical.

3.3 Some Definable Constructs

Constructs like *true*, $P \vee Q$ and $\exists v.\, P$ are definable as are $\Diamond P$ ("sometimes P"), $\Box P$ ("always P") and $\bigcirc P$ ("next P"):

$$\Diamond P \stackrel{\text{def}}{\equiv} \mathit{true}; P \qquad \Box P \stackrel{\text{def}}{\equiv} \neg \Diamond \neg P \qquad \bigcirc P \stackrel{\text{def}}{\equiv} \mathit{skip}; P \ .$$

We refer to the quantifier-free ITL subset built from no temporal operators but $\Diamond$ and $\bigcirc$ as *simple temporal logic*. Here are more operators expressible in this:

$\circledcirc P \stackrel{\text{def}}{\equiv} \neg \bigcirc \neg P$	(Weak next)	$\mathit{fin}\ P \stackrel{\text{def}}{\equiv} \Box(\mathit{empty} \supset P)$	(Final state)
$\mathit{more} \stackrel{\text{def}}{\equiv} \bigcirc \mathit{true}$	(More states)	$\mathit{halt}\ P \stackrel{\text{def}}{\equiv} \Box(P \equiv \mathit{empty})$	(Just last)
$\mathit{empty} \stackrel{\text{def}}{\equiv} \neg \mathit{more}$	(One state)	$\circledast P \stackrel{\text{def}}{\equiv} \Box(\mathit{more} \supset P)$	(Mostly)

The conventional temporal operator *until*, though definable (with $\exists$), is not needed. A version of $\bigcirc$ for numerical terms is expressible using conditional terms. The formula t *gets* t' is true iff for each pair of adjacent states, the value of term t' at the first state (i.e., on the suffix subinterval starting from it) equals term t's value at the second state:

$$t \textit{ gets } t' \quad \stackrel{\text{def}}{\equiv} \quad \boxed{\text{m}}\big((\bigcirc t) = t'\big) \ .$$

Below are operators for examining *initial* and *arbitrary* subintervals:

$$\boxed{\text{i}}\, P \stackrel{\text{def}}{\equiv} P; \mathit{true} \qquad \boxed{\text{i}}\, P \stackrel{\text{def}}{\equiv} \neg \boxed{\text{i}} \neg P \qquad \boxed{\text{a}}\, P \stackrel{\text{def}}{\equiv} \mathit{true}; P; \mathit{true} \qquad \boxed{\text{a}}\, P \stackrel{\text{def}}{\equiv} \neg \boxed{\text{a}} \neg P \ .$$

4 A Proof System

We now present a proof system for ITL. Our experience with hundreds of proofs has helped refine it. There is a quantifier-free part and another for quantifiers.

4.1 Quantifier-Free Axioms and Inference Rules.

We use some of Rosner and Pnueli's axioms for *chop* [18] and ours for $\boxed{\text{i}}$ and *chop-star* [12]. Let w be a *state formula*, i.e., without temporal operators.

Basic $\vdash$ Substitution instances of all valid quantifier-free state formulas.
P2 $\vdash (P;Q);R \equiv P;(Q;R)$
P3 $\vdash (P \vee P');Q \supset (P;Q) \vee (P';Q)$
P4 $\vdash P;(Q \vee Q') \supset (P;Q) \vee (P;Q')$
P5 $\vdash \mathit{empty};P \equiv P$
P6 $\vdash P;\mathit{empty} \equiv P$
P7 $\vdash w \supset \boxed{\text{i}}\, w$
P8 $\vdash w \supset \Box w$, where variables in w are static.
P9 $\vdash \boxed{\text{i}}(P \supset P') \wedge \Box(Q \supset Q') \supset (P;Q) \supset (P';Q')$
P10 $\vdash \bigcirc P \supset \boxed{\text{w}} P$
P11 $\vdash P \wedge \Box(P \supset \boxed{\text{w}} P) \supset \Box P$
P12 $\vdash P^* \equiv \mathit{empty} \vee (P \wedge \mathit{more});P^*$

MP $\vdash P \supset Q, \ \vdash P \ \Rightarrow \ \vdash Q$
$\Box$Gen $\vdash P \ \Rightarrow \ \vdash \Box P$
$\boxed{\text{i}}$Gen $\vdash P \ \Rightarrow \ \vdash \boxed{\text{i}}\, P$

These axioms and inference rules do not have quantifiers but w, P, etc. can. In Axiom **Basic**, term t substitutes into variable v only if $\mathit{domain}(t) \subseteq \mathit{domain}(v)$. Axiom **P11** enables induction over time.

A formula P deduced from the axiom system is called an *ITL theorem*, denoted $\vdash P$. Below are a few theorems. The full paper has more with some proofs.

T1 $\vdash \Box(P \supset Q) \supset \Box P \supset \Box Q$
T2 $\vdash \bigcirc(P \supset Q) \supset \bigcirc P \supset \bigcirc Q$
T3 $\vdash \Diamond \mathit{empty}$
T4 $\vdash (w \wedge P);Q \equiv w \wedge (P;Q)$
T5 $\vdash P^{**} \equiv P^*$
T6 $\vdash \mathit{skip}^*$

4.2 Axioms and Inference Rules for Quantifiers

In the axioms and inference rules for quantifiers, v is an arbitrary variable:

Q1 $\vdash \forall v.P \quad \supset \quad P^t_v$,
where v is free for t in P and $domain(t) \subseteq domain(v)$. (The full paper describes substitution into temporal contexts.)

Q2 $\vdash \forall v.(P \supset Q) \supset (P \supset \forall v.Q)$, where v does not occur freely in P.

Q3 $\vdash \exists v.(P;Q) \quad \equiv \quad (\exists v.P);Q$, where v does not occur freely in Q.

Q4 $\vdash \exists v.(P;Q) \quad \equiv \quad P;(\exists v.Q)$, where v does not occur freely in P.

Q5 $\vdash (\exists v.P);\bigcirc(\exists v.Q) \supset \exists v.(P;\bigcirc Q)$, where v is a state variable.

$\forall$**Gen** $\vdash P \quad \Rightarrow \quad \vdash \forall v.P$, for any variable v.

The next theorem expresses *chop-star* using a fresh boolean variable B:

T7 $\vdash P^* \quad \equiv \quad \exists B.\Big(B \wedge \boxplus\big(B \supset \diamondsuit(P \wedge \bigcirc halt\, B)\big)\Big)$.

Here is one to construct a hidden state variable v always equaling t:

T8 $\vdash \exists v.\Box(v = t)$,
where $domain(t) \subseteq domain(v)$ and v does not occur freely in t.

The one below creates a hidden state variable v which is initialized and then incrementally assigned a term which can depend on v's current value:

T9 $\vdash \exists v.(v = t \wedge v\ gets\ t')$,
where $domain(t) \subseteq domain(v)$ and $domain(t') \subseteq domain(v)$. Also v does not occur freely in t or within the temporal operators in t'.

Thus, the boolean variable B below initially equals *false* and always flips:

$$\vdash \quad \exists B.(B = false \wedge B\ gets\ \neg B) \ .$$

One can easily show that the axiom system is *sound*, that is, $\vdash P$ implies $\models P$. Our main goal is conversely to establish *completeness*, that is, $\models P$ implies $\vdash P$:

Theorem 4.1 (Completeness) *Any valid ITL formula is also a theorem.*

5 Overview of the Proof of Completeness

The basic completeness proof assumes formulas contain no static variables:

Lemma 5.1 (Relative completeness for static variables) *If all valid formulas without static variables are theorems, so are those with them.*

Proof (Outline) Let P be a valid formula and let P' be $\forall u_1 \ldots \forall u_n.P$, where $u_1, \ldots, u_n$ are the free static variables in P. Now P' is also valid and provably implies P (i.e., $\vdash P' \supset P$). For each static variable u in P', replace any subformula $\forall u.Q$ using the theorem $\vdash \forall u.Q \equiv \bigwedge_{c \in domain(u)} Q^c_u$. The new formula P'' is valid and provably equivalent to P' (i.e., $\vdash P' \equiv P''$). By our assumption, P'' is a theorem so we can deduce P. $\Box$

The proof of Theorem 4.1 reduces formulas to equivalent ones in a *normal form*:

Lemma 5.2 (Normal form) *For each formula we can deduce an equivalent one having no new variables and in which each equality is of the form $v = c$, where v is numerical and $c \in domain(v)$. If the original formula is the simple temporal logic defined in §3.3, so is the normalized one.*

Lemma 5.3 (Completeness for simple temporal logic) *Any valid formula in the simple temporal logic subset is a theorem.*

Proof (Outline) We start with a complete axiom system for conventional linear-time temporal logic easily altered for finite intervals and finite domains. All of the axioms and inference rules are provable from our ITL axiom system. They are Axioms **Basic**, **P8**, **P10** and **P11**, Inference Rules **MP** and $\Box$**Gen** and Theorems **T1**, **T2** and **T3**. Therefore, ITL theoremhood of any valid formula in the subset can be deduced. □

5.1 Automata

We now adapt the approach of Kesten and Pnueli [7] and utilize a variant of finite-state automata called here *chop-automata* which selectively accept intervals. All normalized ITL formulas can be built from a few constructs, each having a translation into an automaton. In effect, we embed a decision procedure for ITL in ITL itself and use the logic to express the procedure's correctness. This helps to show completeness.

The behavior of an automaton $\mathcal{A}$ can be expressed in ITL as the formula denoted $\chi^{\mathcal{A}}$ (defined later) which is true on an interval iff $\mathcal{A}$ accepts that interval. We then construct from a formula P an automaton $\mathcal{A}^P$ accepting the intervals satisfying P. The formula $\chi^{\mathcal{A}^P}$ represents $\mathcal{A}^P$'s accepting runs and is provably equivalent to P in the axiom system (i.e., $\vdash P \equiv \chi^{\mathcal{A}^P}$). The shorter form χ^P is generally used. We can also show for any automata $\mathcal{A}$ accepting no intervals, the formula $\chi^{\mathcal{A}}$ is provably false (i.e., $\vdash \neg\chi^{\mathcal{A}}$).

To prove that a valid formula P is a theorem, we construct from $\neg P$ an automaton $\mathcal{A}^{\neg P}$ with $\vdash \neg P \equiv \chi^{\neg P}$. Now P is valid, so $\mathcal{A}^{\neg P}$ accepts nothing and we deduce $\vdash \neg\chi^{\neg P}$. These together yield $\vdash P$.

We now describe *chop-automata* for recognizing finite intervals.

Definition 5.4 (Chop-automaton) *A (nondeterministic)* chop-automaton $\mathcal{A}$ *is a quintuple* $(V, K, q_0, \delta, \tau)$ *for which*

- V *is a possibly empty finite set of boolean and numerical state variables,*
- K *is a nonempty finite set of* automaton states,
- $q_0 \in K$ *is the* initial state,
- δ *is the* transition function *mapping* $K \times K$ *to quantifier-free state formulas over variables in* V,
- τ *is the* termination function *mapping* K *to quantifier-free state formulas over variables in* V.

It is necessary to introduce the notion of a *run* of a chop-automaton on an interval:

Definition 5.5 (Run and accepting run of chop-automaton) *A* run *of a chop-automata $\mathcal{A}$ over an interval σ is any finite sequence ρ of $|\sigma|+1$ elements $\rho_0, \ldots, \rho_{|\sigma|} \in K$ in which for each two adjacent automaton states ρ_i and ρ_{i+1} the interval state σ_i satisfies the transition formula $\delta(\rho_i, \rho_{i+1})$, (i.e., $\sigma_i \models \delta(\rho_i, \rho_{i+1})$).*

A run ρ is called an accepting run *of the chop-automaton $\mathcal{A}$ over the interval σ if the run's initial state ρ_0 is q_0 and in addition σ's final state $\sigma_{|\sigma|}$ satisfies the termination condition selected by the run's final automaton state $\rho_{|\sigma|}$, namely $\tau(\rho_{|\sigma|})$.*

We say that $\mathcal{A}$ accepts *an interval σ if there is at least one accepting run over σ.*

In contrast to conventional finite-state automata, a chop-automaton uses τ to test the very end of an interval without advancing to permit the operator *chop* to be represented.

An automaton $\mathcal{A}$'s accepting runs are expressible in ITL. Let Y be a numerical state variable *not* in V and with $K \subseteq \mathit{domain}(Y)$. Define the formula $\mathit{acc\_r}^{\mathcal{A}}(Y)$ as follows:

$$\mathit{acc\_r}^{\mathcal{A}}(Y) \quad \stackrel{\text{def}}{\equiv} \quad Y = q_0 \wedge \boxed{\text{m}}\, \delta(Y, \bigcirc Y) \wedge \mathit{fin}\, \tau(Y) \ .$$

The formula $\chi^{\mathcal{A}}$ now defined expresses the existence of some accepting run:

$$\chi^{\mathcal{A}} \quad \stackrel{\text{def}}{\equiv} \quad \exists Y.\, \mathit{acc\_r}^{\mathcal{A}}(Y) \ .$$

5.2 Automata Constructions

Given some normalized formula P (see Lemma 5.2 presented earlier), we construct an automaton $\mathcal{A}^P$. We sometimes denote the individual parts of $\mathcal{A}^P$ as V^P, K^P, etc. and abbreviate $\mathit{acc\_r}^{\mathcal{A}^P}(Y)$ and $\chi^{\mathcal{A}^P}$ as $\mathit{acc\_r}^P(Y)$ and χ^P, respectively.

Here is a list of formulas which we need to consider: w (quantifier-free state formula), $P \vee Q$, $\neg P$, $\exists v.\, P$, *skip*, and $P;Q$. These constructs are ones most readily translated to automata. We replace *chop-star* formulas using ITL Theorem **T7** in §4.2 to avoid the need to also directly reduce them to automata.

During the construction of automata, various operations can be performed such as renaming of an automaton's states or determinizing it. These are expressible as ITL theorems. The details are omitted here. We also have the following lemma:

Lemma 5.6 *If $\mathcal{A}$ has no accepting runs, then $\vdash \neg\chi^{\mathcal{A}}$.*

Proof Suppose that $\mathcal{A}$ has no accepting runs. The following formula is valid and hence a theorem by Lemma 5.3: $\vdash \neg \mathit{acc\_r}^{\mathcal{A}}(Y)$. We introduce an existential quantifier to deduce the theorem $\vdash \neg\exists Y.\, \mathit{acc\_r}^{\mathcal{A}}(Y)$ which reduces to $\vdash \neg\chi^{\mathcal{A}}$. □

Below are constructions for *w*, *skip* and *chop*. The full paper also looks at others.

Automata for Quantifier-Free State Formulas For a quantifier-free state formula w, the automaton $\mathcal{A}^w$ has V equal the set of w's variables, $K = \{0,1\}$, $q_0 = 0$ with δ and τ as follows:

$$\begin{array}{llll} \delta(0,0)\colon \mathit{false} & \delta(0,1)\colon w & \tau(0)\colon w & \tau(1)\colon \mathit{true} \\ \delta(1,0)\colon \mathit{false} & \delta(1,1)\colon \mathit{true} & & \end{array}$$

Claim 5.7 *The automaton $\mathcal{A}^w$ accepts an interval σ iff σ satisfies w.*

Lemma 5.8 *The following equivalence is an ITL theorem:* $\vdash w \equiv \chi^w$.

Proof Let X be a numerical state variable with domain $\{0,1\}$ and not occurring in w. The following valid simple temporal formula is a theorem by Lemma 5.3:

$$\vdash \quad w \wedge X = 0 \wedge X \; gets \; 1 \; \supset \; acc\_r^w(X) \; . \tag{5.1}$$

The variable X can now be existentially created using ITL Theorem **T9**:

$$\vdash \quad \exists X . (X = 0 \wedge X \; gets \; 1) \; . \tag{5.2}$$

The two theorems (5.1) and (5.2) are combined to obtain the following:

$$\vdash \quad w \; \supset \; \exists X . acc\_r^w(X) \; . \tag{5.3}$$

For the converse, the valid formula $acc\_r^w(X) \supset w$ is a theorem by Lemma 5.3. We then deduce $\vdash \exists X . acc\_r^w(X) \supset w$ which with (5.3) yields $\vdash w \equiv \chi^w$. □

Automaton for *skip* Below is an automaton $\mathcal{A}^{skip}$ accepting two-state intervals:

$$V = \{\}, \; K = \{0,1\}, \; q_0 = 0,$$

$\delta(0,0)$: *false* $\quad \delta(0,1)$: *true* $\qquad \tau(0)$: *false* $\quad \tau(1)$: *true*
$\delta(1,0)$: *false* $\quad \delta(1,1)$: *false*

Claim 5.9 *The automaton $\mathcal{A}^{skip}$ accepts an interval σ iff σ satisfies skip.*

Lemma 5.10 *The following equivalence is provably true:* $\vdash \; skip \equiv \chi^{skip}$.

Proof We first look at deducing $skip \; \supset \; \chi^{skip}$. Let X have domain $\{0,1\}$. The next valid formula is a theorem by Lemma 5.3:

$$\vdash \quad skip \wedge \Box(X = if \; more \; then \; 0 \; else \; 1) \; \supset \; acc\_r^{skip}(X) \; . \tag{5.4}$$

A hidden instance of X is now created with ITL Theorem **T8** in §4.2:

$$\vdash \quad \exists X . \Box(X = if \; more \; then \; 0 \; else \; 1) \; . \tag{5.5}$$

We then combine the two theorems (5.4) and (5.5):

$$\vdash \quad skip \; \supset \; \exists X . acc\_r^{skip}(X) \; . \tag{5.6}$$

For the converse, the valid formula below is a theorem by Lemma 5.3:

$$\vdash \quad acc\_r^{skip}(X) \; \supset \; skip \; .$$

The variable X is existentially hidden to deduce the following:

$$\vdash \quad \exists X . acc\_r^{skip}(X) \; \supset \; skip \; . \tag{5.7}$$

We reach the goal by combining (5.6) and (5.7): $\vdash skip \equiv \chi^{skip}$. □

Automata for *chop* Let us now construct an automaton $\mathcal{A}^{P;Q}$ for the formula $P;Q$ and deduce the equivalence of $P;Q$ and $\chi^{P;Q}$. Assume by induction that $\mathcal{A}^P$ and $\mathcal{A}^Q$ are P's and Q's respective automata with disjoint K^P and K^Q. Here is a suitable $\mathcal{A}^{P;Q}$:

$$V = V^P \cup V^Q,\ K = K^P \cup K^Q,\ q_0 = q_0^P,$$

$\delta(q,q')$:		$\tau(q)$:	
$\delta^P(q,q')$,	for $q,q' \in K^P$	$\tau^P(q) \wedge \tau^Q(q_0^Q)$,	for $q \in K^P$
$\delta^Q(q,q')$,	for $q,q' \in K^Q$	$\tau^Q(q)$	for $q \in K^Q$
$\tau^P(q) \wedge \delta^Q(q_0^Q,q')$,	for $q \in K^P, q' \in K^Q$		
false,	otherwise		

Claim 5.11 *The automaton $\mathcal{A}^{P;Q}$ accepts an interval σ iff σ satisfies $P;Q$.*

Lemma 5.12 *Formulas $P;Q$ and $\chi^{P;Q}$ are provably equivalent: $\vdash P;Q \equiv \chi^{P;Q}$.*

Proof We assume by induction $\vdash P \equiv \chi^P$ and $\vdash Q \equiv \chi^Q$ and then deduce the following:

$$\vdash P;Q \equiv \chi^P;\chi^Q \tag{5.8}$$

To show that the valid formula $\chi^P;\chi^Q \equiv \chi^{P;Q}$ is a theorem, we re-express it:

$$\big(\exists X.\, acc\_r^P(X)\big);\big(\exists Y.\, acc\_r^Q(Y)\big) \quad \equiv \quad \exists Z.\, acc\_r^{P;Q}(Z)\ .$$

Here X, Y and Z share a domain which is a superset of K^P, K^Q and K. The left subformula's quantifiers can be moved out of the *chop* operator:

$$\vdash\ \big(\exists X.\, acc\_r^P(X)\big);\big(\exists Y.\, acc\_r^Q(Y)\big) \equiv \exists X,Y.\, \big(acc\_r^P(X);acc\_r^Q(Y)\big)\ . \tag{5.9}$$

We now turn to the formula $acc\_r^P(X);acc\_r^Q(Y)$. This is provably equivalent to the formula $\exists B.\,\phi(B,X,Y)$, where B is a new boolean state variable and the subformula $\phi(B,X,Y)$ is as now defined:

$$\begin{array}{rcl}
\phi(B,X,Y) & \stackrel{\text{def}}{\equiv} & X = q_0^P \wedge B \\
& & \wedge\ \boxdot(B \supset \delta^P(X,\bigcirc X)) \\
& & \wedge\ \boxdot(\neg B \supset \delta^Q(Y,\bigcirc Y) \wedge \bigcirc \neg B) \\
& & \wedge\ \Box\big(B \wedge \circledcirc \neg B \supset \tau^P(X) \wedge Y = q_0^Q \wedge \big(\mathit{empty} \vee \delta^Q(Y,\bigcirc Y)\big)\big) \\
& & \wedge\ \mathit{fin}\,\tau^Q(Y)\ .
\end{array}$$

We represent the automata's behavior using ϕ because it is in simple temporal logic. The purpose of B is to indicate at each interval state which of the *chop* construct's two subintervals contains the state. When B is true, the state is enclosed in the left subinterval and automaton $\mathcal{A}^P$ is active. When B is false the state is within the right one and $\mathcal{A}^Q$ is active. In the case of the single state shared by both intervals, B remains true as in the left subinterval. If the right subinterval has only one state then B is always true.

The relationship between ϕ and $acc\_r^P(X);acc\_r^Q(Y)$ is now expressed:

$$\vdash \quad \exists B.\,\phi(B,X,Y) \quad \equiv \quad acc\_r^P(X);acc\_r^Q(Y)\ . \tag{5.10}$$

In order to prove this, we first deduce the next theorem:

$$\vdash \quad \phi(B,X,Y) \quad \equiv \quad (acc\_r^P(X) \wedge \Box B); (acc\_r^Q(Y) \wedge B \wedge \circledcirc \Box \neg B) \ .$$

Its proof uses temporal fixpoints and derived inference rules for compositionality. The details are omitted here. Lemma 5.3 yields the theoremhood of some valid formulas involving ϕ such as the next one for creating Z from B, X and Y:

$$\vdash \quad \phi(B,X,Y) \wedge \Box(Z = \mathit{if}\ B\ \mathit{then}\ X\ \mathit{else}\ Y) \supset acc\_r^{P;Q}(Z) \ .$$

We combine this with ITL Theorem **T8** in §4.2 and hide B and Z:

$$\vdash \quad \exists B.\, \phi(B,X,Y) \quad \supset \quad \exists Z.\, acc\_r^{P;Q}(Z) \ .$$

From this and ITL Theorems (5.9) and (5.10) we achieve half of the goal:

$$\vdash \quad \big(\exists X.\, acc\_r^P(X)\big); \big(\exists Y.\, acc\_r^Q(Y)\big) \supset \exists Z.\, acc\_r^{P;Q}(Z) \ . \tag{5.11}$$

For the converse of (5.11), we first deduce the next valid formula using Lemma 5.3:

$$\begin{array}{l} \vdash \quad acc\_r^{P;Q}(Z) \wedge \Box\big(B \equiv (Z \in K^P)\big) \wedge \Box(X = Z) \\ \qquad \wedge Y = q_0^Q \wedge Y\ \mathit{gets}\ \big(\mathit{if}\ (\bigcirc B)\ \mathit{then}\ Y\ \mathit{else}\ \bigcirc Z\big) \\ \qquad \supset \quad \phi(B,X,Y) \ . \end{array}$$

We then existentially hide B, X and Y using ITL Theorems **T8** and **T9**:

$$\vdash \quad acc\_r^{P;Q}(Z) \supset \exists B,X,Y.\, \phi(B,X,Y) \ .$$

This, (5.9) and (5.10) lead to our desired equivalence's other direction:

$$\vdash \quad \exists Z.\, acc\_r^{P;Q}(Z) \supset \big(\exists X.\, acc\_r^P(X)\big); \big(\exists Y.\, acc\_r^Q(Y)\big) \ . \tag{5.12}$$

From ITL Theorems (5.11) and (5.12) and χ's definition, we get $\vdash \chi^P;\chi^Q \equiv \chi^{P;Q}$. This and theorem (5.8) yield our main goal: $\vdash P;Q \equiv \chi^{P;Q}$. □

6 Discussion

The version of ITL here uses numerical variables. Alternatively, we can restrict all variables to being boolean and encode numbers as Kesten and Pnueli do.

In [13] we look at a compositional axiom system for temporal projection over finite intervals which is claimed to be complete. We would also like support for ω-intervals.

Hale [5] first studied *framing* in ITL. At present, if a state variable does not change value, this must be explicitly specified. Framing makes this implicit and shortens specifications. A complete axiom system for framing would be helpful.

Acknowledgments

The author thanks Antonio Cau and Jordan Dimitrov for suggesting improvements to the presentation. Moshe Vardi and Wolfgang Thomas also provided helpful advice.

References

[1] H. Bowman and S. J. Thompson. A complete axiomatization of Interval Temporal Logic with projection. Technical Report 6-00, Computing Lab., Univ. of Kent, UK, Jan. 2000.

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

[3] B. Dutertre. Complete proof systems for first order interval temporal logic. In *Proc. 10th LICS*, pp. 36–43. IEEE Computer Soc. Press, June 1995.

[4] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, Apr. 1979.

[5] R. W. S. Hale. *Programming in Temporal Logic*. PhD thesis, Computer Laboratory, Cambridge University, Cambridge, England, Oct. 1988.

[6] J. Halpern, Z. Manna, and B. Moszkowski. A hardware semantics based on temporal intervals. In J. Diaz, ed., *Proc. ICALP '83*, LNCS 154, pp. 278–291. Springer-Verlag, 1983.

[7] Y. Kesten and A. Pnueli. A complete proof system for QPTL. In *Proc. 10th LICS*, pp. 2–12. IEEE Computer Soc. Press, 1995.

[8] B. Moszkowski. *Reasoning about Digital Circuits*. PhD thesis, Dept. Computer Science, Stanford Univ., 1983. Tech. rep. STAN–CS–83–970.

[9] B. Moszkowski. A temporal logic for multi-level reasoning about hardware. In *Proc. 6-th Int'l. Symp. on Computer Hardware Description Languages*, pp. 79–90, Pittsburgh, Penn., May 1983. North-Holland Pub. Co.

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

[11] B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge U. Press, 1986.

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

[13] B. Moszkowski. Compositional reasoning about projected and infinite time. In *Proc. 1st IEEE Int'l Conf. on Engineering of Complex Computer Systems (ICECCS'95)*, pp. 238–245. IEEE Computer Soc. Press, 1995.

[14] B. Moszkowski. Compositional reasoning using Interval Temporal Logic and Tempura. In W.-P. de Roever et al., eds., *Compositionality: The Significant Difference*, LNCS 1536, pp. 439–464. Springer-Verlag, 1998.

[15] B. Moszkowski. A complete axiomatization of interval temporal logic with infinite time. In *Proc. 15th Annual IEEE Symp. on Logic in Computer Science (LICS 2000)*. IEEE Computer Soc. Press, June 2000.

[16] B. Paech. Gentzen-systems for propositional temporal logics. In E. Börger et al., eds., *Proc. 2nd Workshop on Computer Science Logic, Duisburg (FRG)*, LNCS 385, pp. 240–253. Springer-Verlag, Oct. 1988.

[17] V. R. Pratt. Semantical considerations on Floyd-Hoare logic. In *17-th Annual IEEE Symposium on Foundations of Computer Science*, pp. 109–121, 1976.

[18] R. Rosner and A. Pnueli. A choppy logic. In *Proc. 1st LICS*, pp. 306–313. IEEE Computer Soc. Press, June 1986.

[19] D. Siefkes. *Decidable Theories I: Büchi's Monadic Second Order Successor Arithmetic*, Lecture Notes in Mathematics 120. Springer-Verlag, Berlin, 1970.

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

[21] Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Inf. Process. Lett.*, 40(5):269–276, 1991.