In *Proceedings of the Fifteenth Annual IEEE Symposium on Logic in Computer Science* (LICS 2000)
June 26-29, 2000, Santa Barbara, California, USA
IEEE Computer Society Press, pages 242–251

A Complete Axiomatization of Interval Temporal Logic with Infinite Time (Extended Abstract)

B. C. Moszkowski*
Software Technology Research Lab.
SERCentre
Hawthorn Building
De Montfort University
The Gateway
Leicester LE1 9BH
Great Britain
email: 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 give 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. The full paper (and another conference paper) presents the basic framework for finite time. Here and in the full paper the axiom system (and completeness) is extended to infinite time.*

1. Introduction

Interval Temporal Logic (ITL) [12, 18, 20] 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. On finite nonempty words, propositional ITL (even without quantifiers) has the same expression power as conventional finite-state automata, regular expressions, *Quantified Propositional Temporal Logic* (QPTL) for finite time and the *Weak Second-Order Theory of Successor* (WS1S) studied by Büchi [3] and Elgot [9]. Propositional ITL over ω-words has the expressiveness of Büchi automata, ω-regular expressions and the *Second-Order Theory of Successor* (S1S) investigated by Büchi [4] and QPTL. However, ITL's notation makes it particularly suitable for logic-based modular reasoning involving periods of time, refinement [6], sequential composition using assumptions and commitments which are fixpoints of various temporal operators [21, 23] as well as for executable specifications [20]. Various imperative programming constructs are expressible in ITL and operators for projecting between time granularities are available (but not considered here). Zhou Chaochen, Hoare and Ravn [13,34] have developed a real-time ITL extension called *Duration Calculus* for hybrid systems. Several researchers have looked at decision procedures and axioms systems for variations of ITL. However, to date no one has found a provably complete axiom system for a version of ITL over both finite and ω-words having no artificial restrictions on interval constructs and achieving the same expressiveness (over ω-words) as S1S and other comparable notations. We present here a natural and complete axiomatization for a subset of quantified ITL in which variables are limited to finite domains. The completeness proof describes an ITL decision procedure within ITL itself.

We build on the work of Siefkes [30] who first proved the completeness of an axiomatization of S1S and Kesten and Pnueli [15] who established the completeness of an axiomatization of QPTL with past-time operators. Our approach closely follows Kesten and Pnueli's technique. We re-

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

duce temporal formulas to finite-state automata and Büchi-automata as part of a decision procedure. The automata are themselves directly manipulated in the logic as advocated by Büchi himself [3, page 82]. Siefkes calls this *syntactization* of the decision procedure [30, page XI]. Following Kesten and Pnueli, we express Safra's intricate algorithm [29] within ITL to perform the hardest part of the proof involving the complementation of Büchi-automata. However we achieve completeness without the use of past-time. The actual axiom system and completeness proof vary substantially from Kesten and Pnueli's work. This reflects differences between conventional temporal logics and interval-based ones.

Our results show that a natural yet complete axiom system for a nontrivial subset of ITL is achievable and that the associated decision procedure can be represented in the logic. The use of automata leads to a proof that is more compositional than conventional tableaux methods for temporal logics which must repeatedly analyze several formulas in parallel. Besides that, there is no need to adapt Fischer-Ladner closures [10], originally developed for a propositional version of Pratt's Dynamic Logic [26], and to prove an associated small-model property. In addition, this work demonstrates that the ITL axiom system provides a logical framework for reasoning about classical operations on finite-state automata and regular expressions over both finite and ω-words. It also shows that a very complicated algorithm such Safra's and a version of König's Infinity Lemma are provable in ITL.

2. Related Work

We now discuss other work on ITL axiom systems. A completeness proof for such notations typically involves a decision procedure so we make mention of this as well. Halpern and Moszkowski [17, pages 23–24] prove the decidability of propositional ITL with quantifiers over finite time by translation to QPTL for finite time which is decidable. Rosner and Pnueli [28] investigate an axiom system for propositional, quantifier-free ITL with finite and ω-intervals. The ITL subset 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. This requires an adaptation of Fischer-Ladner closures mentioned earlier in §1. Unfortunately, one inference rule requires detailed meta-reasoning about tableaux transitions. This and the lack of *chop-star* limit the axiom system's practical use.

Paech [25] investigates a quantifier-free version of ITL with ω-intervals having *chop-star* limited, like Kleene-star, to finitely many iterations and including an additional temporal operator *unless*. Due to a theorem of Thomas [31] (later more simply proved by Y. Choueka and D. Peleg [7]), ITL with such a restricted *chop-star* is still as expressive as ω-regular expressions (and hence S1S and QPTL) as well as quantifier-free propositional ITL with unrestricted *chop-star* (which permits ω consecutive finite iterations) although with possibly less succinctness. Paech presents a complete Gentzen-style proof system including some nonconventional axioms restricting ITL formulas to be in a form analogous to regular expressions. This can potentially require complex meta-reasoning about arbitrary ITL formulas over finite intervals. Surprisingly, the axioms, unlike Rosner and Pnueli's, apparently limit intervals to be infinite. The completeness proof necessitates a generalized form of Fischer-Ladner closures.

Dutertre [8] gives two complete proof systems for first-order ITL without *chop-star* for finite time. The first uses a possible-worlds semantics of time and the second considers arbitrary linear orderings of states. Neither is complete for standard discrete-time intervals. Wang Hanpin and Xu Qiwen [33] generalize this to infinite time.

Kono [16] presents a tableaux-based decision procedure for propositional ITL with quantifiers and temporal projection over finite time which has been implemented in Prolog. No formal proof is given that all models are considered. A sketchy argument about termination is presented. Kono suggests using the transformations as the basis for a complete axiom system. Moszkowski [21] presents propositional and first-order ITL axiom systems for finite intervals. They support compositional proofs based on the *rely-guarantee* paradigm of Jones [14]. The propositional part is claimed to be complete but only an outline of a proof is given. This is extended in [22] to axioms for temporal projection.

Bowman and Thompson [1] present a tableaux-based decision procedure for quantifier-free propositional ITL over finite time with temporal projection. They omit a proof of termination. In [2] they look at termination and also obtain a completeness proof for an axiomatization of this version of ITL.

3. Overview of Interval Temporal Logic

We now briefly describe ITL for finite time. More details and examples are found in [12, 17–21, 23]. The treatment of ω-intervals is deferred until §5.

Basic ITL is a temporal logic with discrete, linear time. An interval σ in general 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.

A variable v's values in an interval range over the finite, nonempty set $domain(v)$ which for our purposes is either $\{false, true\}$ or some initial subsequence of the natural numbers. Finite data domains ensure the existence of

a decision procedure for our completeness result. We can readily extend *domain* to all constructs later introduced.

Basic ITL contains conventional operators such as $\wedge$, $\forall$ and $=$. Terms and formulas evaluate relative to an interval's beginning. Thus, the formula $I = 2$ is true on interval σ iff I's value in σ_0 equals 2.

There are three primitive temporal operators:

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

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 $\textit{skip};I = J$ 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 $\textit{false}^*$) in true on a one-state interval (see §3.4). Figure 1 pictorially illustrates the semantics of *skip*, *chop*, and *chop-star*. Some simple ITL formulas together with intervals which satisfy them are shown in Fig. 2.

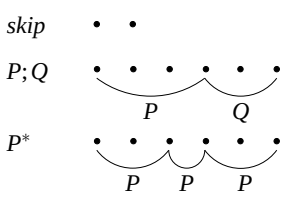

Figure 1. Informal illustration of ITL semantics

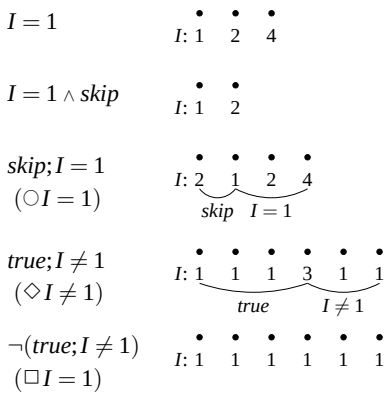

Figure 2. Some sample ITL formulas and satisfying intervals

Let us now look at the syntax and later the semantics of all the basic constructs used here.

3.1. Syntax of ITL

The syntax of ITL constructs considered here is now given in more detail. Here are permitted constructs using variable v, terms t and t' and formulas P and Q:

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

3.2. Semantics of Constructs

Let us take a more formal look at the semantics of basic ITL constructs. The notation $\sigma_i(v)$ denotes the value of variable v in the state of interval σ indexed by i. Furthermore, for natural numbers i, j with $i \leq j \leq |\sigma|$, the notation $\sigma_{i:j}$ denotes the subinterval of length $j - i$ (i.e., $j - i + 1$ states) with starting state σ_i and final state σ_j. We define the semantics of each term t and formula P on an interval σ using the respective notations $\mathcal{M}_\sigma[\![t]\!]$ and $\mathcal{M}_\sigma[\![P]\!]$.

3.3. Semantics of Terms

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

3.4. Semantics of Formulas

- Boolean static or state variable: $\mathcal{M}_\sigma[\![v]\!] = \sigma_0(v)$. The value of a boolean variable for an interval σ is the variable's value in the initial state σ_0.
- Equality: $\mathcal{M}_\sigma[\![t = t']\!] = \textit{true}$ iff $\mathcal{M}_\sigma[\![t]\!] = \mathcal{M}_\sigma[\![t']\!]$.
- Negation: $\mathcal{M}_\sigma[\![\neg P]\!] = \textit{true}$ iff $\mathcal{M}_\sigma[\![P]\!] = \textit{false}$.
- Conjunction: $\mathcal{M}_\sigma[\![P \wedge Q]\!] = \textit{true}$ iff $\mathcal{M}_\sigma[\![P]\!] = \mathcal{M}_\sigma[\![Q]\!] = \textit{true}$.
- Universal quantification: $\mathcal{M}_\sigma[\![\forall v.P]\!] = \textit{true}$ iff $\mathcal{M}_{\sigma'}[\![P]\!] = \textit{true}$, for every interval σ' identical to σ except possibly for variable v's behavior.
- Unit interval: $\mathcal{M}_\sigma[\![\textit{skip}]\!] = \textit{true}$ iff $|\sigma| = 1$.
- Chop: $\mathcal{M}_\sigma[\![P;Q]\!] = \textit{true}$ iff $\mathcal{M}_{\sigma'}[\![P]\!] = \textit{true}$ and $\mathcal{M}_{\sigma''}[\![Q]\!] = \textit{true}$, where $\sigma' = \sigma_{0:i}$ and $\sigma'' = \sigma_{i:|\sigma|}$ for some $i \leq |\sigma|$. The subintervals σ' and σ'' share state σ_i.
- Chop-star: $\mathcal{M}_\sigma[\![P^*]\!] = \textit{true}$ iff $\mathcal{M}_{\sigma_{l_i:l_{i+1}}}[\![P]\!] = \textit{true}$, for each $i : 0 \leq i < n$, for some $n \geq 0$ 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 take $n = 0$.

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 ITL theorems for quantified boolean and numerical variables. Therefore the equality $P = Q$ is identical to $P \equiv Q$.

3.5. 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 subset:

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

The conventional temporal operator *until*, though definable (with $\exists$), is not needed. Versions of $\bigcirc$ and *fin* for numerical terms are 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:

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

For example the following formula describes a boolean variable B which initially equals *false* and always flips:

$$B = \mathit{false} \wedge B \ \mathit{gets} \ \neg B \ .$$

Here are operators for examining initial (i.e., prefix) and arbitrary (i.e., infix) subintervals:

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

4. A Proof System

We now present a proof system for ITL. Experience with hundreds of proofs has helped us refine the axioms. The proof system has a quantifier-free part and another dealing with quantifiers.

4.1. Quantifier-Free Axioms and Inference Rules.

We use some of Rosner and Pnueli's axioms for *chop* [28] and our own for the operators $\boxed{\text{i}}$ and *chop-star* [21]. Here w denotes a *state formula* containing no temporal operators.

Basic $\vdash$ Substitution instances of all valid nonmodal quantifier-free 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 w only contains static variables.
P9 $\vdash \boxed{\text{i}}(P \supset P') \wedge \Box(Q \supset Q') \supset (P;Q) \supset (P';Q')$
P10 $\vdash \bigcirc P \supset \circledw P$
P11 $\vdash P \wedge \Box(P \supset \circledw P) \supset \Box P$
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 contain no explicit quantifiers but w, P, etc. can. In Axiom **Basic**, a boolean or numerical term t is substitutable into a variable v only if $\mathit{domain}(t) \subseteq \mathit{domain}(v)$.

A formula P deducible from the axiom system is called an *ITL theorem*, denoted $\vdash P$.

Here are some sample theorems:

T1 $\vdash \boxed{\text{i}}(P \supset P') \supset (P;Q) \supset (P';Q)$
T2 $\vdash \Box(P \supset Q) \supset \Box P \supset \Box Q$
T3 $\vdash \bigcirc(P \supset Q) \supset \bigcirc P \supset \bigcirc Q$
T4 $\vdash \Box P \supset P \wedge \circledw \Box P$
T5 $\vdash \mathit{skip}^*$
T6 $\vdash (w \wedge P);Q \equiv w \wedge (P;Q)$
T7 $\vdash \Diamond \mathit{empty}$
T8 $\vdash \Box(\circledw P \supset P) \supset P$
T9 $\vdash P^{**} \equiv P^*$

Theorems **T7** and **T8** only apply to finite intervals. Theorem **T8** provides a convenient way to do backward induction from an interval's end. In contrast, Axiom **P11** enables forward induction over time. The full paper has more theorems and derived inference rules and many proofs.

4.2. Axioms and Inference Rules for Quantifiers

Below are axioms and inference rules for quantifiers. Here v refers to an arbitrary static or state variable.

Q1 $\vdash \forall v.P \quad \supset \quad P^t_v$,
where the variable v is free for the term t in P. (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.

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

The ITL Theorem **T10** expresses *chop-star* using an auxiliary boolean state variable:

T10 $\vdash P^* \equiv \exists B.\Big(B \wedge \boxdot\big(B \supset \diamondsuit(P \wedge \bigcirc \mathit{halt}\, B)\big)\Big)$,
where B does not occur freely in P.

One can easily show that the axiom system is *sound*, that is, $\vdash P$ implies $\models P$. Our full paper (and another conference paper [24]) conversely proves *completeness*, that is $\models P$ implies $\vdash P$, using finite-state automata. Here is a brief look:

Theorem 4.1 (Completeness for finite-time) *Any ITL formula which is valid for finite time is deducible as a theorem from the ITL axiom system.*

Proof (outline) Let us first consider for an *unsatisfiable* formula P not containing free static variables how to deduce the theorem $\vdash \neg P$. We translate P into an equivalent but quite low-level formula $\exists Y.P'$, where P' is expressed using the simple temporal logic subset described earlier in §3.5. The formula $\exists Y.P'$ explicitly describes a finite-state automaton's runs and is much easier to prove theorems about. The equivalence between P and $\exists Y.P'$ is deducible as the theorem $\vdash P \equiv \exists Y.P'$.

Now P itself is unsatisfiable and therefore the automaton has no accepting runs. As a result, $\exists Y.P'$ is also unsatisfiable and so is P'. Consequently, we readily deduce $\vdash \neg P'$ using a completeness theorem we prove for the simple temporal logic subset and then obtain $\vdash \neg\exists Y.P'$. From this and the earlier equivalence $\vdash P \equiv \exists Y.P'$ we easily deduce $\vdash \neg P$.

In order to establish completeness, suppose that Q is some arbitrary *valid* formula which we wish to deduce as a theorem. We first universally quantify any free static variables in it to obtain a new formula Q' which is valid iff Q is. In addition, we can readily prove the theorem $\vdash Q' \supset Q$. Now $\neg Q'$ is unsatisfiable and by using the automaton representation we obtain $\vdash \neg\neg Q'$. Combining this with the earlier theorem $\vdash Q' \supset Q$, we achieve our goal of Q's theoremhood, that is, $\vdash Q$. □

5. ITL with Infinite Time

The ITL formalism so far presented is limited to finite intervals. We now discuss modifications needed to also permit ω-intervals. First, we extend the semantics of *chop* and *chop-star* to ω-intervals. The formula $P;Q$ is true on an ω-interval if the interval can be divided into a finite subinterval satisfying P followed by another adjacent ω-interval satisfying Q or alternatively if the original ω-interval itself satisfies P. In the latter case, we ignore Q. Pratt first proposed this semantics in his definition of the *fusion product* operator [27, page 95]. The formula P^* is true on an ω-interval that is divisible into a finite number of subintervals where the last one itself has length ω and each satisfies P or alternatively into ω consecutive finite intervals each satisfying P. Our use of Kleene star is somewhat nonstandard since it is normally denotes a finite number of iterations whereas in ITL the formula P^* is satisfiable by ω iterations. However, this convention seems convenient and natural for ITL. In [21] we defined new constructs for testing whether an interval is infinite or finite and alter the definition of $\diamond$. We also include the operator *chop-omega* introduced here in a straightforward manner:

$$\mathit{inf} \stackrel{\text{def}}{\equiv} \mathit{true};\mathit{false} \qquad \diamond P \stackrel{\text{def}}{\equiv} \mathit{finite};P$$
$$\mathit{finite} \stackrel{\text{def}}{\equiv} \neg\mathit{inf} \qquad P^{\omega} \stackrel{\text{def}}{\equiv} \mathit{inf} \wedge (P \wedge \mathit{finite})^*$$

Note that the definitions of *fin* and *halt* given earlier in §3.5 are left unchanged. On ω-intervals the formula *fin P* is equivalent to *true* and *halt P* is equivalent to $\Box\neg P$.

We extend the proof system presented in §4 to include the following additional axioms:

E1 $\vdash (P \wedge \mathit{inf});Q \equiv P \wedge \mathit{inf}$

E2 $\vdash \mathit{inf} \wedge P \wedge \Box\big(P \supset (Q \wedge \mathit{more});P\big) \supset Q^*$

E3 $\vdash \mathit{inf} \wedge \big(\forall u.\exists v.(v = u \wedge P)\big)^* \supset \forall u.\exists v.(v = u \wedge P^*)$,
where u is a static variable not occurring freely in P and v is a state variable with exactly the same data domain as u.

Axiom **E1** and a slight variant of **E2** were already presented by us in [21]. Axiom **E2** helps to compensate for the lack of backward induction on ω-intervals since they have no final state from where to start. Axiom **E3** is new and intended for reasoning about an ω-interval spanned by a number of consecutive subintervals, each having its own instance of an existentially quantified variable. A single existential variable for the overall ω-interval is obtained by sequentially

fusing those of the subintervals together. In order to ensure that fusion is possible, Axiom **E3** works in a forward direction by requiring that within each iteration, the associated existential variable can initially equal any value in its data domain so that it is able to start with whatever value the corresponding variable in an immediately preceding iteration ends with. This axiom enables the construction of various auxiliary variables needed for the completeness proof.

Let us illustrate the use of Axiom **E3** with a small example in which a be a boolean static variable and B be a boolean state variable. Here the formula P is taken to be $\mathit{finite} \wedge \Diamond B \wedge \Diamond \neg B$:

$$\begin{array}{ll} \vdash & \mathit{inf} \wedge \big(\forall a.\, \exists B.\, (B = a \wedge \mathit{finite} \wedge \Diamond B \wedge \Diamond \neg B)\big)^* \\ & \supset \quad \forall a.\, \exists B.\, \big(B = a \wedge (\mathit{finite} \wedge \Diamond B \wedge \Diamond \neg B)^*\big) \ . \end{array}$$

Claim 5.1 *The extended axiom system is sound.*

We now let the notations $\models P$ and $\vdash P$ respectively denote the validity and theoremhood of P in the extended system over both finite and ω-intervals. The more explicit but equivalent notations $\models_{\leq\omega} P$ and $\vdash_{\leq\omega} P$ are sometimes used instead. The notations $\models_{<\omega} P$ and $\vdash_{<\omega} P$ refer to the original syntax, semantics and proof system presented earlier and limited to finite intervals.

Here are two representative ITL theorems:

$$\begin{array}{ll} \vdash & P^* \wedge \mathit{inf} \ \equiv \ \big((P^* \wedge \mathit{finite}); (P \wedge \mathit{inf})\big) \vee P^\omega \\ \vdash & \mathit{inf} \wedge (w \wedge P)^* \quad \supset \quad (P \wedge \mathit{fin}\, w)^* \ . \end{array}$$

The appendix contains a number of ITL theorem with proofs.

6. Completeness and Automata for ω-Intervals

The extended axiom system's completeness proof is similar to Theorem 4.1's given in §4.2. However, automata suitable for both finite and ω-intervals are required. Before introducing them, we mention four lemmas:

Lemma 6.1 (Relative completeness for static variables) *If all valid formulas without static variables are theorems, so are those with them.*

Hence the basic completeness proof can assume formulas have no static variables. We also assume that any formula to be translated into an automaton is in a *normal form*:

Lemma 6.2 (Normal form) *For each formula we can deduce an equivalent one with no new variables and in which each equality is of the form $v = c$, where v is numerical and $c \in \mathit{domain}(v)$. If the original formula is the simple temporal logic defined in §3.5, so is the normalized one.*

Lemma 6.3 (Theorems of simple temporal logic) *Any valid quantifier-free formula in which the only temporal constructs used are $\Diamond$, $\bigcirc$ and others derivable from them is a theorem.*

Proof (outline) From the extended system a complete axiom system is deduced for a conventional quantifier-free temporal logic. We use Axioms **Basic**, **P8**, **P10** and **P11**, Inference Rules **MP** and □**Gen** and versions of Theorems **T2** and **T3** proved for the extended system. □

Lemma 6.4 *For any formula P (even one containing quantifiers and interval constructs) satisfied by all finite intervals (i.e., $\models_{<\omega} P$), the formula $\mathit{finite} \supset P$ is a theorem in the extended system (i.e., $\vdash_{\leq\omega} \mathit{finite} \supset P$).*

Proof (outline) The completeness of the original ITL axiom system ensures that P is a theorem in it, that is, $\vdash_{<\omega} P$. We then perform induction on the size of the proof to obtain $\vdash_{\leq\omega} \mathit{finite} \supset P$. □

We now examine *chop-automata* for recognizing both finite and ω-intervals. These are modified versions of conventional finite automata tailored for use with ITL. We adapt Kesten and Pnueli's techniques (based on Büchi [3,4]) for representing ω-automata in QPTL [15]. Unlike them, we also consider finite intervals and omit past-time.

Definition 6.5 (Chop-automaton) *A (nondeterministic)* chop-automaton $\mathcal{A}$ *for finite and ω-intervals is a sextuple $(V, K, q_0, \delta, \tau, F)$ 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,*
- *$F \subseteq K$ is the set of* final states.

We assume here that K is a subset of the natural numbers.

Let us now define the notions of a *run* and *an accepting run* of a chop-automaton on an interval:

Definition 6.6 (Run and accepting run) *A* run *of a chop-automata $\mathcal{A}$ over an interval σ is any finite or ω-sequence ρ of one or more elements $\rho_0, \ldots$ 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 finite interval σ if the run's initial state ρ_0 is q_0 and σ's final state $\sigma_{|\sigma|}$ satisfies the termination condition of the run's final automaton state $\rho_{|\sigma|}$, namely $\tau(\rho_{|\sigma|})$. In addition, the automaton accepts an ω-run iff at least one of the final states in F occurs infinitely often in the run. This is the well-known Büchi acceptance condition.*

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

Note that only τ is used for accepting finite intervals and only F is used for infinite ones. We say that $\mathcal{A}$ *accepts* an interval σ if there is at least one accepting run over σ.

Let Y be some numerical state variable not in V and with $K \subseteq \mathit{domain}(Y)$. The next formula $\mathit{acc\_r}^{\mathcal{A}}(Y)$ expresses both finite and infinite accepting runs of automaton $\mathcal{A}$:

$$\begin{array}{l} \mathit{acc\_r}^{\mathcal{A}}(Y) \stackrel{\text{def}}{\equiv} \\ \quad Y = q_0 \wedge \boxdot \delta(Y, \bigcirc Y) \\ \quad \wedge \left(\mathit{if\ finite\ then\ fin}\ \tau(Y)\ \mathit{else}\ \Box \Diamond Y \in F \right) \ . \end{array}$$

Unlike conventional finite automata, a chop-automaton uses τ to test a finite interval's very end without advancing. This permits operators such as *chop* to be represented.

6.1. Testing for Emptiness of Automata over Finite and ω-intervals

The completeness theorem requires that we can prove the falsity of any ITL formula that corresponds to an automaton having no accepting runs. The proof is analogous to the one used for the finite-interval proof system.

Lemma 6.7 *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 so an immediate theorem by Lemma 6.3:

$$\vdash \quad \neg \mathit{acc\_r}^{\mathcal{A}}(Y) \ .$$

Here Y is a numerical state variable not in V with $K \subseteq \mathit{domain}(Y)$. By introducing an existential quantifier, we have the next theorem:

$$\vdash \quad \neg \exists Y. \mathit{acc\_r}^{\mathcal{A}}(Y) \ .$$

This reduces to our immediate goal: $\vdash \neg \chi^{\mathcal{A}}$. □

6.2. Constructions for ω-Automata

Let us look at automata for a basic set of ITL constructs from which all others can be expressed. Here is a suitable list of formulas: 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 and are generally modified versions of ones for finite intervals (see conference paper [24]). The only exception is negation which is much harder for ω-intervals. A version of ITL Theorem **T10** in §4.2 is deduced to express *chop-star* using other operators.

Let us abbreviate $\mathit{acc\_r}^{\mathcal{A}^P}(Y)$ as $\mathit{acc\_r}^P(Y)$ and $\chi^{\mathcal{A}^P}$ as χ^P. In general we wish to deduce the following:

$$\vdash P \equiv \chi^P \ ,$$

where P is an arbitrary ITL formula. The proof is done inductively on the syntax of P.

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

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

Here is how to construct an automaton $\mathcal{A}^{P \vee Q}$ for the formula $P \vee Q$. 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. We denote the individual parts of the automaton $\mathcal{A}^P$ recognizing formula P as V^P, K^P, etc. and denote the parts of $\mathcal{A}^Q$ in a similar manner. The following is a suitable $\mathcal{A}^{P \vee Q}$:

$$V = V^P \cup V^Q,\ K = K^P \cup K^Q \cup \{q_0\},\ F = F^P \cup F^Q \ ,$$
for some new start state q_0 not in $K^P \cup K^Q$.

$$\delta(q,q')\colon \begin{cases} \delta^P(q_0^P, q'), & \text{for } q = q_0 \text{ and } q' \in K^P \\ \delta^Q(q_0^Q, q'), & \text{for } q = q_0 \text{ and } q' \in K^Q \\ \delta^P(q,q'), & \text{for } q, q' \in K^P \\ \delta^Q(q,q'), & \text{for } q, q' \in K^Q \\ \mathit{false}, & \text{otherwise} \end{cases}$$

$$\tau(q)\colon \begin{cases} \tau^P(q_0^P) \vee \tau^Q(q_0^Q), & \text{for } q = q_0 \\ \tau^P(q), & \text{for } q \in K^P \\ \tau^Q(q), & \text{for } q \in K^Q \end{cases}$$

The next automaton $\mathcal{A}^{\mathit{skip}}$ accepts two-state intervals:

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

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

Here is an automaton $\mathcal{A}^{\exists v.P}$ for recognizing $\exists v. P$:

$$V = V^P \backslash \{v\},\ K = K^P,\ q_0 = q_0^P,\ F = F^P$$

$$\delta(q,q')\colon \bigvee_{c \in \mathit{domain}(v)} \delta^P(q,q')_v^c \quad \tau(q)\colon \bigvee_{c \in \mathit{domain}(v)} \tau^P(q)_v^c$$

Note that the operator $\backslash$ denotes set difference and ensures here that V does not include v. In addition v does not occur in any of δ's or τ's elements. Instead, we use an explicit disjunction of all possible values v could assume.

To construct an automaton for $P;Q$, assume without loss of generality that K^P and K^Q are disjoint sets. Here are the values used for $\mathcal{A}^{P;Q}$:

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

$$\delta(q,q')\colon \begin{cases} \delta^P(q,q'), & \text{for } q, q' \in K^P \\ \delta^Q(q,q'), & \text{for } q, q' \in K^Q \\ \tau^P(q) \wedge \delta^Q(q_0^Q, q'), & \text{for } q \in K^P, q' \in K^Q \\ \mathit{false}, & \text{otherwise} \end{cases}$$

$$\tau(q)\colon \begin{cases} \tau^P(q) \wedge \tau^Q(q_0^Q), & \text{for } q \in K^P \\ \tau^Q(q) & \text{for } q \in K^Q \end{cases}$$

The treatment of negation requires a major modification of the techniques for finite intervals. The reason is that complementing an ω-automata is much harder than complementing a conventional one. We now look at this.

6.3. ω-Automata for Negation

Following Kesten and Pnueli, we use Safra's algorithm [29] (see also Thomas' discussion in [32]). This is a nontrivial but elegant variation of the standard method of complementing an automaton by first constructing a deterministic version of it. Our purpose is not to give an exposition and justification of Safra's methods but rather to embed them in ITL. Consequently, where appropriate we can rely on certain intermediate results proved by Safra. The full paper describes the embedding in detail.

Safra's algorithm makes use of two intermediate ω-automata which are themselves not Büchi-automata. It first produces from the original automaton, say $\mathcal{A}^P$, a *deterministic Rabin-automaton*, denoted here $\mathcal{A}^{DR}$, which accepts an interval iff it is an ω-interval accepted by $\mathcal{A}^P$. This is the most subtle part of the construction. Deterministic Büchi automata are not adequate since they have strictly less expressiveness than nondeterministic ones. The Rabin-automaton $\mathcal{A}^{DR}$ is then readily complemented to produce a *deterministic Streett-automaton*, denoted here $\mathcal{A}^{DS}$. Finally an equivalent conventional nondeterministic Büchi-automaton, denoted $\mathcal{A}^{NB}$, is obtained with only a little effort from $\mathcal{A}^{DS}$. The automaton $\mathcal{A}^{NB}$ achieves the goal of recognizing the complement of the original automaton's language over ω-intervals. Therefore, the correctness of the algorithm leads to us having $\models \chi^{NB} \equiv (\mathit{inf} \wedge \neg\chi^P)$.

The only difference between Büchi-, Rabin- and Streett-automata involves the acceptance conditions used. Instead of having a single set of final states F, Rabin- and Streett-automata both include a finite number of pairs of sets of states (L_i, U_i). Table 1 shows all three variations of acceptance conditions together with a convenient notation used to denote each as well the semantics given as an ITL formula for use in *acc_r*, χ and any related constructs. We let $L_i(Y)$ and $U_i(Y)$ denote state formulas containing Y as the sole free variable (i denotes a constant in this context).

In order to embed Safra's algorithm in our completeness result, we deduce within the axiom system the following logical equivalences:

$$\vdash \mathit{inf} \wedge \chi^P \equiv \chi^{DR}, \quad \vdash \mathit{inf} \supset \chi^{DR} \equiv \neg\chi^{DS}, \quad \vdash \chi^{DS} \equiv \chi^{NB}.$$

After all this, we still need to obtain an automaton which accepts the complement of $\mathcal{A}^P$ on both finite and ω-intervals. A new automaton $\mathcal{A}^{\neg P}$ is therefore defined in the manner now described. Let $\mathcal{A}'$ be the negated automaton for finite intervals. Let the set of final states F' in $\mathcal{A}'$ be empty. Construct $\mathcal{A}^{NB}$ using Safra's algorithm and for each $q \in K^{NB}$, let $\tau(q) = \mathit{false}$. Finally, use the construction for *logical disjunction* to obtain a new automaton $\mathcal{A}^{\neg P}$ which recognizes the union of the languages of the two automata $\mathcal{A}'$ and $\mathcal{A}^{NB}$. Our goal is achieved: $\vdash \mathcal{A}^{\neg P} \equiv \neg\mathcal{A}^P$.

Table 1. Büchi-, Rabin- and Streett-acceptance conditions for ω-automata

Type	Notation	Semantics in temporal logic
Büchi	$F \subseteq K$	$\mathit{inf} \wedge \Box\Diamond Y \in F$
Rabin	$\bigvee_i (L_i \wedge \neg U_i)$	$\mathit{inf} \wedge \bigvee_i \big(\Box\Diamond L_i(Y) \wedge \Diamond\Box \neg U_i(Y)\big)$
Streett	$\bigwedge_i (L_i \supset U_i)$	$\mathit{inf} \wedge \bigwedge_i \big(\Box\Diamond L_i(Y) \supset \Box\Diamond U_i(Y)\big)$

6.3.1 A Version of König's Lemma

Safra uses König's Lemma in order to relate $\mathcal{A}^P$ and $\mathcal{A}^{DR}$. We now consider a restricted version of it suitable for our purposes:

Lemma 6.8 *Assume a forest of n trees for some $n \geq 1$ containing altogether a countably infinite number of nodes. Some trees can however be finitary. If at each level of depth there are exactly n nodes among all the trees, then at least one tree's root has an infinite path of descendant nodes.*

To express this in ITL, let D and L be numerical state variables with D ranging over the n values $0, \ldots, n-1$ and L ranging over $0, \ldots, n^2-1$. We have L encode a vector of n values each ranging over $0, \ldots, n-1$. Let $L[i]$ denote its i-th element. We can define this using nested conditional terms. The behavior of L over an ω-interval corresponds to a forest described in Lemma 6.8. Specifically, if in some state (level) we have $L[i] = j$, then node j in the immediately previous state (level) acts as the father of node i. Nodes in the initial state are roots. The next ITL theorem expresses Lemma 6.8's guarantee of an infinite path's existence:

$$\vdash \quad \mathit{inf} \supset \exists D. \Box(D = \bigcirc L[D]).$$

Its proof inductively constructs a path from some root node in the initial state (level) and needs the following ITL theorem ensuring that one of them has descendants on every later level:

$$\vdash \quad \mathit{inf} \quad \supset \quad \bigvee_{i<n} \phi(i) \ ,$$

where $\phi(i)$ is defined below:

$$\phi(i) \quad \stackrel{\text{def}}{\equiv} \quad \boxdot_1 \Big(\mathit{finite} \supset \exists D. \big(D = i \wedge \boxdot(D = \bigcirc L[D])\big)\Big) \ .$$

We also require that if a node has this property, so does some immediate descendant:

$$\vdash \quad \mathit{inf} \wedge \phi(j) \quad \supset \quad \bigcirc \bigvee_{i<n} \big(L[i] = j \wedge \phi(i)\big) \ .$$

The following ITL theorem then incrementally constructs a suitable D:

$$\vdash \quad \exists D. \Big(D = \min i. \phi(i) \\ \wedge D \ \mathit{gets} \min i. \big((\bigcirc L[i]) = D \wedge \bigcirc \phi(i)\big)\Big) \ ,$$

where i's domain is the same as D's. For a static numerical variable u, the term $\min u.P$ is defined to denote u's minimum value satisfying the formula P or 0 if none exists. It is easily expressed using conditional terms. The proofs of the theorems given here are in the full paper. The next important observation is exploited:

Lemma 6.9 *For any formula $\boxdot(\mathit{finite} \supset P)$, there exists a deterministic automaton $\mathcal{A}$ for finite intervals (i.e., $F^{\mathcal{A}} = \{\}$) with the following deducible equivalence:*

$$\vdash \boxdot(\mathit{finite} \supset P) \equiv \exists Y. \big(Y = q_0 \wedge \boxplus \delta(Y, \bigcirc Y) \wedge \Box \tau(Y)\big) \ .$$

Here Y is a numerical state variable not in $V^{\mathcal{A}}$ and with $K^{\mathcal{A}} \subseteq \mathit{domain}(Y)$.

The right formula is much simpler to reason about since there is only one quantifier and its body is in simple temporal logic. Furthermore Y's behavior is deterministic.

Proof (Outline) Let $\mathcal{A}$ be an automaton corresponding to P for finite time. We prove its existence in the full paper as part of Theorem 4.1. Without loss of generality, assume that $\mathcal{A}$ is deterministic. This is also provable in ITL. Now we express the subformula P in $\boxdot(\mathit{finite} \supset P)$ using $\mathcal{A}$:

$$\vdash \boxdot(\mathit{finite} \supset P) \quad \equiv \\ \boxdot\Big(\mathit{finite} \supset \exists Y. \big(Y = q_0 \wedge \boxplus \delta(Y, \bigcirc Y) \wedge \mathit{fin}\, \tau(Y)\big)\Big) \ .$$

The deterministic behavior of $\mathcal{A}$ ensures that all runs starting from a given automaton state are identical in any of their common interval states. Therefore a single instance of Y is sufficient and we can export most of the behavior out of $\boxdot$:

$$\vdash \boxdot\Big(\mathit{finite} \supset \exists Y. \big(Y = q_0 \wedge \boxplus \delta(Y, \bigcirc Y) \wedge \mathit{fin}\, \tau(Y)\big)\Big) \equiv \\ \exists Y. \Big(Y = q_0 \wedge \boxplus \delta(Y, \bigcirc Y) \wedge \boxdot\big(\mathit{finite} \supset \mathit{fin}\, \tau(Y)\big)\Big).$$

We then re-express the subformula $\boxdot\big(\mathit{finite} \supset \mathit{fin}\, \tau(Y)\big)$ in simple temporal logic:

$$\vdash \quad \boxdot\big(\mathit{finite} \supset \mathit{fin}\, \tau(Y)\big) \quad \equiv \quad \Box \tau(Y) \ .$$

Combining these steps leads to the desired goal. □

6.4. Conversion of a Büchi-Automaton to a Deterministic Rabin-Automaton

The full paper considers the various types of ω-automata needed in Safra's construction and their relation to one another. Here we only overview the first and hardest step in the process. Let us start with the formula P and its automaton $\mathcal{A}^P$ which we seek to complement. We ignore the termination condition τ^P since it is irrelevant on ω-intervals. A deterministic Rabin-automaton $\mathcal{A}^{DR}$ is now constructed using Safra's algorithm. Our presentation only explicitly deals with those features of the construction which are relevant here. The reader should refer to Safra's paper for further details. Safra establishes that every accepting run of $\mathcal{A}^{DR}$ has a corresponding accepting run of $\mathcal{A}^P$ (*soundness*[1]) and vice versa (*completeness*[2]). These properties can both be represented in ITL by formulas which are guaranteed to be valid by Safra's original proof of correctness:

$$\begin{array}{ll} \text{soundness of } \mathcal{A}^{DR}\text{:} & \models \chi^{DR} \supset \mathit{inf} \wedge \chi^P \ , \\ \text{completeness of } \mathcal{A}^{DR}\text{:} & \models \mathit{inf} \wedge \chi^P \supset \chi^{DR} \ . \end{array}$$

We need to ensure that both of these formulas are deducible. In the full paper, suitable constructions are given and theorems are proved for the deterministic Streett-automaton $\mathcal{A}^{DS}$ and the nondeterministic Büchi-automaton $\mathcal{A}^{NB}$.

As with other automata here, we represent the states of the deterministic Rabin automaton $\mathcal{A}^{DR}$ as natural numbers. In addition, in line with Safra's construction, the Rabin automaton has an associated nonempty, finite set denoted here as *labels*. The labels are taken to be some initial segment of the natural numbers. Each state $q \in K^{DR}$ of the Rabin automation has a nonempty set $\mathit{nodes}(q) \subseteq \mathit{labels}$ equaling a set of *nodes* associated with q. Each of q's nodes is colored either *white* or *green* with the possibly empty set of q's green nodes denoted $\mathit{green}(q)$. The full proof requires some further information about the internal structure of the nodes but we do not describe this here.

We can omit the details of Safra's transition function $\delta^{DR}(q, q')$ and just assume it exists. Safra also defines the particular acceptance condition of the Rabin automaton, denoted here in ITL as $\mathit{acc}^{DR}(Z)$, to ensure that some node n is eventually always associated with each automaton state and that this node is infinitely often green:

$$\mathit{acc}^{DR}(Z) \quad \stackrel{\text{def}}{\equiv} \quad \mathit{inf} \wedge \bigvee_{n \in \mathit{labels}} \big(\Box \Diamond L_n(Z) \wedge \Diamond \Box \neg U_n(Z)\big) \ ,$$

where $L_n(Z)$ and $U_n(Z)$ are defined as follows:

$$L_n(Z) \stackrel{\text{def}}{\equiv} n \in \mathit{green}(Z) \ , \qquad U_n(Z) \stackrel{\text{def}}{\equiv} n \notin \mathit{nodes}(Z) \ .$$

[1] Not to be confused with *logical soundness* of a proof system.
[2] Not to be confused with *logical completeness* of a proof system.

We can re-express $acc^{DR}(Z)$ in the manner given below:

$$\vdash\ acc^{DR}(Z) \quad \equiv \quad inf \wedge \bigvee_{n \in labels} \big(\Diamond\Box\, n \in nodes(Z) \wedge \Box\Diamond\, n \in green(Z)\big)\ .$$

This formula is valid and so a theorem by Lemma 6.3.

Here is the definition of an accepting run of $\mathcal{A}^{DR}$:

$$acc\_r^{DR}(Z) \quad \stackrel{\text{def}}{\equiv} \quad Z = q_0^{DR} \wedge run^{DR}(Z) \wedge acc^{DR}(Z)\ ,$$

where $run^{DR}(Z) \stackrel{\text{def}}{=} \boxed{\text{m}}\, \delta^{DR}(Z, \bigcirc Z)$.

6.4.1 Soundness of the Deterministic Rabin-Automaton $\mathcal{A}^{DR}$

The proof of $\mathcal{A}^{DR}$'s soundness has the following five steps:

1. Expression of $\mathcal{A}^{DR}$'s accepting ω-runs as a disjunction in which each disjunction tests one pair of L_i and U_i.
2. Splitting of an accepting ω-run of $\mathcal{A}^{DR}$ into a finite prefix and a ω-suffix with
   $\Box\, n \in nodes(Z) \wedge \Box\Diamond\, n \in green(Z)$.
3. Synthesis of finite run of $\mathcal{A}^{P}$ from finite prefix of $\mathcal{A}^{DR}$'s run.
4. Synthesis of ω-run of $\mathcal{A}^{P}$ from rest of $\mathcal{A}^{DR}$'s run.
5. Fusion of these two runs into an accepting run of $\mathcal{A}^{P}$.

6.4.2 Completeness of the Deterministic Rabin-Automaton $\mathcal{A}^{DR}$

Deduction of the completeness of the Rabin-automaton $\mathcal{A}^{DR}$ requires ensuring that for any accepting ω-run of $\mathcal{A}^{P}$, some accepting ω-run of $\mathcal{A}^{DR}$ also exists. Due to its deterministic behavior, $\mathcal{A}^{DR}$ has a run on every finite and ω-interval starting with any desired automaton state in K^{DR}. We use this to first show that for any interval there exists a run of $\mathcal{A}^{DR}$ with the initial state being q_0^{DR}. However, this run is not necessarily accepting. Therefore we establish that if $\mathcal{A}^{P}$ has an accepting ω-run, than the associated ω-run of $\mathcal{A}^{DR}$ starting in automaton state q_0^{DR} is also accepting.

Synthesis of run of $\mathcal{A}^{DR}$ Every automaton state of $\mathcal{A}^{DR}$ is deterministic. Therefore, the behavior of a run can be expressed as the formula $Z\ gets\ t$ where t is a conditional term readily build using the transitions specified by δ^{DR}. For each value in Z's data domain not in K^{DR}, we have t act as the identity function. The following valid formula formalizes t's behavior and is a theorem by Lemma 6.3:

$$\vdash \quad Z \in K^{DR} \quad \supset \quad Z\ gets\ t \ \equiv\ run^{DR}(Z)\ . \qquad (6.1)$$

We then deduce an ITL theorem which in essence computes a suitable Z for $\mathcal{A}^{DR}$ in any interval:

$$\vdash \quad \exists Z.\, (Z = a \wedge Z\ gets\ t)\ .$$

Here the numerical variable a's data domain is the same as Z's. The automaton state q_0^{DR} can be selected as the initial state. This together with (6.1) then permits the subformula $Z\ gets\ t$ to be replaced by the formula $run^{DR}(Z)$:

$$\vdash \quad \exists a.\, (Z = q_0^{DR} \wedge run^{DR}(Z))\ . \qquad (6.2)$$

Proof of existence of accepting run of $\mathcal{A}^{DR}$ Safra proves that any accepting ω-run of the original automaton $\mathcal{A}^{P}$ guarantees that the corresponding ω-run of $\mathcal{A}^{DR}$ is also accepting. This can be expressed by the following valid formula which by Lemma 6.3 is also an immediate theorem:

$$\vdash\ inf \wedge acc\_r^{P}(X) \wedge Z = q_0^{DR} \wedge run^{DR}(Z)\ \supset\ acc^{DR}(Z)\ .$$

This together with (6.2) and some reasoning about quantifiers leads to the existence of an accepting ω-run for $\mathcal{A}^{DR}$ from any accepting ω-run of $\mathcal{A}^{P}$:

$$\vdash \quad inf \wedge \exists X.\, acc\_r^{P}(X) \quad \supset \quad \exists Z.\, \big(Z = q_0^{DR} \wedge run^{DR}(Z) \wedge acc^{DR}(Z)\big)\ .$$

This is re-expressed as our goal of automaton completeness:

$$\vdash \quad inf \wedge \chi^{P} \quad \supset \quad \chi^{DR}\ .$$

In the full paper we use the results mentioned here as part of a detailed analysis establishing completeness:

Theorem 6.10 (Completeness of the extended system) *Any ITL formula which is valid for all finite and ω-intervals is a theorem of the extended axiom system.*

7. Discussion

The completeness proof currently uses numerical state variables in the representation of automata. For ITL with only finite intervals, one can restrict all variables to being boolean and encode numerical ones. In fact, this is exactly what Kesten and Pnueli do in their proof for QPTL over ω-intervals. However, our Axiom **E3** for ω-intervals loses its simplicity if an encoding is done since it can require a single numerical variable be replaced by several boolean ones. We are looking at a pair of alternative axioms without this limitation to use instead of **E3**:

$$\vdash \quad inf \quad \supset \quad \exists v.\, \boxed{\text{i}}\big(finite\ \supset\ (fin\ v) = t\big)$$
$$\vdash \quad inf \wedge (\bigcirc \exists v.\, P)^{*} \quad \supset \quad \exists v.\, (\bigcirc P)^{*}\ ,$$

where the state variable v does not occur in term t and has $domain(t) \subseteq domain(v)$.

In [22] 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 [11] first studied *framing* in ITL. In ITL, if a state variable does not change value, this must be made explicit. 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 tableaux method for interval temporal logic with projection. In *TABLEAUX'98, International Conference on Analytic Tableaux and Related Methods*, Lecture Notes in AI 1397, pp. 108–123. Springer-Verlag, May 1998.

[2] 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.

[3] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[4] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. Int. Congress on Logic, Methodology, and Philosophy of Science 1960*, pp. 1–12, Stanford, Calif., 1962. Stanford Univ. Press. Reprinted in [5, pp. 425–435].

[5] J. R. Büchi. *The Collected Works of J. Richard Büchi*, S. Mac Lane and D. J. Siefkes, eds. Springer-Verlag, New York, 1990.

[6] 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.

[7] Y. Choueka and D. Peleg. A note on ω-regular languages. *EATCS Bulletin*, (21):21–23, Oct. 1983.

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

[9] C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98:21–51, 1961.

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

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

[12] 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.

[13] M. R. Hansen and Zhou Chaochen. Duration calculus: Logical foundations. *Formal Aspects of Computing*, 9(3):283–330, 1997.

[14] C. B. Jones. Specification and design of (parallel) programs. In R. E. A. Mason, ed., *Proc. IFIP Congress '83*, pp. 321–332, Amsterdam, 1983. North Holland Pub. Co.

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

[16] S. Kono. A combination of clausal and non-clausal temporal logic programs. In M. Fisher and R. Owens, eds., *Executable Modal and Temporal Logics*, Lecture Notes in AI 897, pp. 40–57. Springer-Verlag, Feb. 1995.

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

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

[24] B. Moszkowski. An automata-theoretic completeness proof for interval temporal logic. In *Proc. 27th Int'l. Colloquium on Automata, Languages and Programming (ICALP 2000)*, LNCS, July 2000.

[25] 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.

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

[27] V. R. Pratt. Process logic. In *Sixth ACM Symp. on Principles of Programming Languages*, pp. 93–100, 1979.

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

[29] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on the Foundations of Computer Science*, pp. 319–327. IEEE Computer Soc. Press, 1988.

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

[31] W. Thomas. Star-free regular sets of ω-sequences. *Inf. and Control*, 42(2):148–156, Aug. 1979.

[32] W. Thomas. Languages, automata, and logic. In G. Rozenburg and A. Salomaa, eds., *Handbook of Formal Languages* vol. 3: Beyond words, chapter 7, pp. 389–455. Springer-Verlag, Berlin, 1997.

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

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

A. Some ITL Theorems for the Extended Axiom System

In this appendix we use the extended axiom system to deduce a number of ITL theorems about ω-intervals. Many of the theorems are also applicable to finite intervals as well. However theorems **T7** and **T8** mentioned in §4.1 depend on time being finite and are therefore not theorems in the extended system. Versions of them which explicitly require finite time can be expressed and proved.

Here are a number a theorems and derived rules which have corresponding finite-time versions.

ET4 $\vdash \quad \boxdot(P \supset P') \quad \supset \quad (P;Q) \supset (P';Q)$
EDR1 $\vdash P \supset P' \quad \Rightarrow \quad \vdash P;Q \supset P';Q$
EDR2 $\vdash P \equiv P' \quad \Rightarrow \quad \vdash P;Q \equiv P';Q$
ET5 $\vdash \quad (P \vee P');Q \quad \equiv \quad P;Q \vee P';Q$
ET6 $\vdash \quad (\bigcirc P);Q \quad \equiv \quad \bigcirc(P;Q)$
EDR3 $\vdash P \supset Q \quad \Rightarrow \quad \vdash \bigcirc P \supset \bigcirc Q$
ET7 $\vdash \quad \bigcirc P \supset more$
ET8 $\vdash \quad skip \quad \supset \quad more$
ET9 $\vdash \quad \bigcirc more \quad \supset \quad \neg skip$

Below are some theorems and proofs specifically obtained from the extended axiom system. We use **Prop** to denote propositional reasoning.

ET10 $\vdash \quad inf \quad \equiv \quad \bigcirc inf$
Proof:

1 $\vdash$	$inf \equiv true;false$	def. of *inf*
2 $\vdash$	$true \equiv more \vee \neg more$	**Prop**
3 $\vdash$	$true \equiv more \vee empty$	2,def. of *empty*
4 $\vdash$	$true;false \equiv (more \vee empty);false$	3,**EDR2**
5 $\vdash$	$(more \vee empty);false \equiv more;false \vee empty;false$	**ET5**
6 $\vdash$	$empty;false \equiv false$	**P5**
7 $\vdash$	$more;false \equiv (\bigcirc true);false$	def. of *more*
8 $\vdash$	$(\bigcirc true);false \equiv \bigcirc(true;false)$	**ET6**
9 $\vdash$	$\bigcirc(true;false) \equiv \bigcirc inf$	def. of *inf*
10 $\vdash$	$inf \equiv \bigcirc inf$	1,4-9,**Prop**

ET11 $\vdash \quad inf \quad \supset \quad skip^*$
Proof:

1 $\vdash$	$inf \equiv \bigcirc inf$	**ET10**
2 $\vdash$	$\bigcirc inf \equiv skip;inf$	def. of $\bigcirc$
3 $\vdash$	$skip \supset more$	**ET8**
4 $\vdash$	$skip \supset skip \wedge more$	3,**Prop**
5 $\vdash$	$skip;inf \supset (skip \wedge more);inf$	4,**EDR1**
6 $\vdash$	$inf \supset (skip \wedge more);inf$	1,2,5,**Prop**
7 $\vdash$	$\Box\big(inf \supset (skip \wedge more);inf\big)$	6,$\Box$**Gen**
8 $\vdash$	$inf \wedge inf \wedge \Box\big(inf \supset (skip \wedge more);inf\big) \supset skip^*$	**E2**
9 $\vdash$	$inf \supset skip^*$	7,8,**Prop**

ET12 $\vdash \quad inf \quad \supset \quad more$
Proof:

1 $\vdash$	$inf \equiv \bigcirc inf$	**ET10**
2 $\vdash$	$\bigcirc inf \supset more$	**ET7**
3 $\vdash$	$inf \supset more$	1,2,**Prop**

ET13 $\vdash \quad inf \quad \supset \quad \Box inf$
Proof:

1 $\vdash$	$inf \equiv \bigcirc inf$	**ET10**
2 $\vdash$	$inf \supset \bigcirc inf$	1,**Prop**
3 $\vdash$	$\bigcirc inf \supset \textcircled{w} inf$	**P10**
4 $\vdash$	$inf \supset \textcircled{w} inf$	2,3,**Prop**
5 $\vdash$	$\Box(inf \supset \textcircled{w} inf)$	4,$\Box$**Gen**
6 $\vdash$	$inf \wedge \Box(inf \supset \textcircled{w} inf) \supset \Box inf$	**P11**
10 $\vdash$	$inf \supset \Box inf$	5,6,**Prop**

ET14 $\vdash \quad inf \quad \supset \quad \neg skip$
Proof:

1 $\vdash$	$inf \equiv \bigcirc inf$	**ET10**
2 $\vdash$	$inf \supset more$	**ET12**
3 $\vdash$	$\bigcirc inf \supset \bigcirc more$	2,**EDR3**
4 $\vdash$	$\bigcirc more \supset \neg skip$	**ET9**
5 $\vdash$	$inf \supset \neg skip$	1,3,4,**Prop**

ET15 $\vdash \quad skip \quad \supset \quad finite$
Proof:

1 $\vdash$	$inf \supset \neg skip$	**ET14**
2 $\vdash$	$skip \supset \neg inf$	1,**Prop**
3 $\vdash$	$skip \supset finite$	2,def. of *finite*

ET16 $\vdash \quad skip \quad \supset \quad \Diamond empty$
Proof:

1 $\vdash$	$skip \supset finite$	**ET15**
2 $\vdash$	$finite;empty \equiv finite$	**P6**
3 $\vdash$	$skip \supset finite;empty$	1,2,**Prop**
4 $\vdash$	$skip \supset \Diamond empty$	3,def. of $\Diamond$

ET17 $\vdash \quad empty \quad \supset \quad finite$
Proof:

1 $\vdash$	$inf \supset more$	**ET12**
2 $\vdash$	$\neg more \supset \neg inf$	1,**Prop**
3 $\vdash$	$empty \supset finite$	2,def. of *empty*, def. of *finite*

ET18 $\vdash \quad \Diamond finite \quad \supset \quad finite$
Proof:

1 $\vdash$	$inf \supset \Box inf$	**ET13**
2 $\vdash$	$inf \supset \neg\Diamond\neg inf$	1,def. of $\Box$
3 $\vdash$	$\Diamond\neg inf \supset \neg inf$	2,**Prop**
4 $\vdash$	$\Diamond finite \supset finite$	3,def. of *finite*