A COMPLETE AXIOM SYSTEM FOR PROPOSITIONAL INTERVAL TEMPORAL LOGIC WITH INFINITE TIME

BEN MOSZKOWSKI

Software Technology Research Laboratory, De Montfort University, Leicester, UK

ABSTRACT. *Interval Temporal Logic (ITL)* is an established temporal formalism for reasoning about time periods. For over 25 years, it has been applied in a number of ways and several ITL variants, axiom systems and tools have been investigated. We solve the longstanding open problem of finding a complete axiom system for basic quantifier-free propositional ITL (PITL) with infinite time for analysing nonterminating computational systems. Our completeness proof uses a reduction to completeness for PITL with finite time and conventional propositional linear-time temporal logic. Unlike completeness proofs of equally expressive logics with nonelementary computational complexity, our semantic approach does not use tableaux, subformula closures or explicit deductions involving encodings of omega automata and nontrivial techniques for complementing them. We believe that our result also provides evidence of the naturalness of interval-based reasoning.

1. INTRODUCTION

Intervals and discrete linear state sequences offer a natural and flexible way to model both sequential and parallel aspects of computational processes involving hardware or software. *Interval Temporal Logic (ITL)* [Mos86] (see also [ITL12]) is an established formalism for rigorously reasoning about such intervals. ITL has a basic construct called *chop* for the sequential composition of two arbitrary formulas as well as an analogue of Kleene star for iteration called *chop-star*. Although originally developed for digital hardware specification [Mos83a, Mos83b, HMM83, Mos85], ITL is suitable for logic-based executable specifications [Mos86], compositional reasoning about concurrent processes [Mos94, Mos95, Mos98, Mos11], refinement [CZ97], as well as for runtime analysis [ZZC99].

Until now, in spite of research over many years involving ITL and its applications, there was no known complete axiom system for quantifier-free propositional ITL (PITL) with infinite time. We present one and prove completeness by a reduction to our earlier complete PITL axiom system for finite time [Mos04] (see also [BT03]) and conventional propositional linear-time temporal logic (PTL). We do not use subformula closures, tableaux, or explicit deductions involving encodings of omega automata and nontrivial techniques for complementing them. Such encodings are typically found in completeness proofs for comparable logics discussed later on (see §11.1), which like PITL have omega-regular expressiveness.

See Thomas [Tho90, Tho97] for more about omega-regular languages, omega automata and some associated logics. Our simple axiom system avoids complicated inference rules and proofs such as axiom systems for an equally expressive version of PITL with restricted sequential iteration [Pae89] and a less expressive version of PITL lacking sequential iteration [RP86]. In the future we plan to use our axiom system as a hierarchical basis for obtaining completeness for some PITL variants. We also believe it can be applied to some other logics and discuss this in Section 12.

Our earlier completeness proof for a larger, more complicated axiom system for quantified ITL with finite domains and infinite time [Mos00] does not work if variables are limited to being just propositional. So that result, while serving as a stepping stone for further research on ITL, even fails to establish axiomatic completeness for a quantified version of PITL (QPITL) with infinite time! For these reasons, we feel justified in regarding the problem of showing axiomatic completeness for full PITL with infinite time as a previously open problem.

We now mention some recent publications by others as evidence of ITL's continuing relevance. None specifically motivate our new completeness proof. Nevertheless, they arguably contribute to making a case for the study of ITL's mathematical foundations, which naturally include axiomatic completeness.

The KIV interactive theorem prover [RSSB98] has for a number of years included a slightly extended version of ITL for interactive theorem proving via symbolic execution both by itself (e.g., see [BBN+10, BSTR11]) and also as a backend notation which supports Statecharts [TSOR04] and UML [BBK+04]. KIV can employ ITL proof systems such as ours. The concluding remarks of [BSTR11] note the following advantages of ITL:

> Our ITL variant supports classic temporal logic operators as well as program operators.
>
> The interactive verifier KIV allows us to directly verify parallel programs in a rich programming language using the intuitive proof principle of symbolic execution. An additional translation to a special normal form (as e.g. in TLA [Temporal Logic of Actions [Lam02]]) using explicit program counters is not necessary.

Axiomatic completeness of PITL is not an absolute requirement for the KIV tool but does offer some benefits. This is because some axioms, inference rules and associated deductions employed to prove completeness can be exploited in KIV, thereby reducing the number of adhoc axioms and inference rules.[1]

Various imperative programming constructs are expressible in ITL and operators for projecting between time granularities are available (but not considered here). ITL influenced an assertion language called *temporal 'e'* [Mor99] which is part of the IEEE Standard 1647 [IEE08] for the system verification language *'e'*.

The *Duration Calculus (DC)* of Zhou, Hoare and Ravn [ZHR91] is an ITL extension for real-time and hybrid systems. The books by Zhou and Hansen [ZH04] and Olderog and Dierks [OD08] both employ DC with finite time and discuss relatively complete axiom systems for it. The second book utilises DC with timed automata to provide a basis for specifying, implementing and model checking suitable real-time systems. Indeed, Olderog and Dierks explain how they regard an interval-oriented temporal logic as being better suited for these tasks than more widely used point-based ones and timed process algebras.

---

[1]Our claim is supported by email correspondence in 2011 with Gerhard Schellhorn of the KIV group.

Concerning point-based logics, they make this comment (on page 23): "In our opinion this leads to complicated reasoning similar to that ... based on predicate logic." As for timed process algebras, they note the following (on page 25): "A difficulty with these formalisms is that their semantics are based on certain scheduling assumptions on the actions like urgency, which are difficult to calculate with."

Within the last ten years, other complete axiom systems for versions of propositional and first-order ITL with infinite time have been presented. These include two by Wang and Xu [WX04] for first-order variants with restricted quantifiers and no sequential iteration as well as a probabilistic extension of theirs by Guelev [Gue07] which all build on an earlier completeness result of Dutertre [Dut95] for first-order ITL restricted to finite time. Like Dutertre, Wang and Xu and also Guelev use a nonstandard abstract-time semantics (e.g., without induction over time) instead of ITL's standard discrete-time one. Their proofs employ Henkin-style infinite sets of maximal consistent formulas. Duan et al. [DZ08, DZK12] give a tableaux-like completeness proof for a related omega-regular logic called *Propositional Projection Temporal Logic (PPTL)*. The only primitive temporal operators in PPTL for sequential composition have varying numbers of operands and concern multiple time granularities. However, both chop and chop-star can be derived. The proof system has over 30 axioms and inference rules, some rather lengthy and intricate. The completeness proof itself involves the nontrivial task of complementing omega-regular languages which can be readily expressed in the logic but it is not discussed. Furthermore, the authors omit much of the prior work in the area developed in the course of over forty years (which we later survey in Section 11). More significantly, they do not explain how they bypass the associated hurdles faced by previous completeness proofs for logics with comparable expressiveness and nonelementary computational complexity. These points make checking the proof's handling of the complementation of omega-regular languages, liveness and other issues rather challenging. Mo, Wang and Duan [MWD11] describe promising applications of Projection Temporal Logic to specifying and verifying asynchronous communication. Zhang, Duan and Tian [ZDT12] investigate the modelling of multicore systems in Projection Temporal Logic. In view of this, the foundational issue of axiomatic completeness for PPTL should be addressed in the future more thoroughly and systematically and better related to other approaches. Incidentally, we already showed in [Mos95] that axiomatic completeness for a version of PITL with a standard version of temporal projection can be simply and hierarchically reduced to axiomatic completeness for PITL without temporal projection. Duan et al. [DZ08, DZK12] however make no mention of this by now long established and powerful technique in their review of prior work.

Here is the structure of the rest of this presentation: Section 2 overviews PITL and the new axiom system. Section 3 concerns a class of PITL theorems from which we can also deduce suitable substitution instances needed later on. Section 4 gives some infrastructure for systematically replacing formulas by other equivalent ones in deductions arising in the completeness proof. Section 5 introduces some useful PITL subsets for later use in the completeness proof. Section 6 reduces completeness for PITL with a kind of infinite sequential iteration to completeness for a subset without this. Section 7 shows how to represent deterministic finite-state semi-automata and automata in PITL. Section 8 employs semi-automata to test a given PITL formula in a finite interval's suffix subintervals. Section 9 shows completeness for the PITL subset without infinite sequential iteration. Section 10

includes some observations about the completeness proof. Section 11 reviews existing complete axiom systems for omega-regular logics. Section 12 discusses some topics for future research.

2. Propositional Interval Temporal Logic

We now describe the version of (quantifier-free) PITL used here. More on basic aspects of ITL can be found in [Mos83a, HMM83, Mos85, Mos86, Mos04] (see also Kröger and Merz [KM08], Fisher [Fis11] and the ITL web pages [ITL12]).

Below is the syntax of PITL formulas in BNF, where p is any propositional variable:

$$A ::= \ true \ | \ p \ | \ \neg A \ | \ A \vee A \ | \ skip \ | \ A^\frown A \ | \ A^\star.$$

The last two constructs are called **chop** and **chop-star**, respectively. The boolean operators $false$, $A \wedge B$, $A \supset B$ (*implies*) and $A \equiv B$ (*equivalence*) are defined as usual. We refer to $A^\frown B$ as **strong chop**, since a weak version $A; B$ also exists. In addition, $A^\star$ (**strong chop-star**) slightly differs from ITL's conventional **weak chop-star** A^*, although the two are interderivable. The strong variants of chop and chop-star taken as primitives here are chosen simply because, without loss of generality, they help streamline the completeness proof.

We use p, q, r and variants such as p' for propositional variables. Variables A, B, C and variants such as A' denote arbitrary PITL formulas. Let w and w' denote **state formulas** without the temporal operators $skip$, chop and chop-star. We have V denote a finite set of propositional variables. Also, V_A denotes the finite set of the formula A's variables.

Time within PITL is discrete and linear. It is represented by *intervals* each consisting of a sequence of one or more states. More precisely, an interval σ is any finite or ω-sequence of one or more states σ_0, σ_1, $\ldots$. Each state σ_i in σ maps each propositional variable p to either *true* and *false*. This mapping is denoted as $\sigma_i(p)$. An interval σ has an *interval length* $|\sigma| \geq 0$, which, if σ is finite, is the number of σ's states minus 1 and otherwise ω. So if σ is finite, it has states σ_0, $\ldots$, $\sigma_{|\sigma|}$. This (standard) version of PITL, with state-based propositional variables, is called **local PITL**.

A **subinterval** of σ is any interval which is a *contiguous* subsequence of σ's states. This includes σ itself.

The notation $\sigma \vDash A$, defined shortly by induction on A's syntax, denotes that interval σ **satisfies** formula A. Moreover, A is **valid**, denoted $\vDash A$, if all intervals satisfy it.

Below are the semantics of the first five constructs:

- True: $\sigma \vDash true$ trivially holds for any σ.
- Propositional variable: $\sigma \vDash p$ iff p is true in the initial state σ_0 (i.e., $\sigma_0(p) = true$).
- Negation: $\sigma \vDash \neg A$ iff $\sigma \nvDash A$.
- Disjunction: $\sigma \vDash A \vee B$ iff $\sigma \vDash A$ or $\sigma \vDash B$.
- Skip: $\sigma \vDash skip$ iff σ has exactly two states.

For natural numbers i, j with $0 \leq i \leq j \leq |\sigma|$, let $\sigma_{i:j}$ be the finite subinterval $\sigma_i \ldots \sigma_j$ (i.e., $j - i + 1$ states). Define $\sigma_{i\uparrow}$ to be σ's suffix subinterval from state σ_i.

Below are semantics for the versions of chop and chop-star found most suitable for the completeness proof. As already noted, other versions can be readily derived.

- Chop: $\sigma \vDash A^\frown B$ iff   for some natural number $i : 0 \leq i \leq |\sigma|$, both $\sigma_{0:i} \vDash A$ and $\sigma_{i\uparrow} \vDash B$. This is called *strong chop* because both A and B must be true.
- Chop-star: $\sigma \vDash A^\star$ iff   one of the following holds:

$\bigcirc A$	$\overset{\text{def}}{\equiv}$	$skip \frown A$	Next
$\Diamond A$	$\overset{\text{def}}{\equiv}$	$true \frown A$	Eventually
$\Box A$	$\overset{\text{def}}{\equiv}$	$\neg \Diamond \neg A$	Henceforth
$more$	$\overset{\text{def}}{\equiv}$	$\bigcirc true$	More than one state
$empty$	$\overset{\text{def}}{\equiv}$	$\neg more$	Only one state
$finite$	$\overset{\text{def}}{\equiv}$	$\Diamond empty$	Finite interval
inf	$\overset{\text{def}}{\equiv}$	$\neg finite$	Infinite interval
$fin\ A$	$\overset{\text{def}}{\equiv}$	$\Box(empty \supset A)$	Weak test of final state
$A \leftarrow B$	$\overset{\text{def}}{\equiv}$	$finite \supset ((fin\ A) \equiv B)$	Temporal assignment
$\Diamond\!\!\!\!\cdot\, A$	$\overset{\text{def}}{\equiv}$	$A \frown true$	Some initial finite subinterval
$\boxdot A$	$\overset{\text{def}}{\equiv}$	$\neg \Diamond\!\!\!\!\cdot\, \neg A$	All initial finite subintervals
$A; B$	$\overset{\text{def}}{\equiv}$	$(A \frown B)\ \vee\ (A \wedge inf)$	Weak chop
$\Diamond\!\!\!\!\cdot\, A$	$\overset{\text{def}}{\equiv}$	$A; true$	Some initial subinterval (even infinite)
$\boxdot A$	$\overset{\text{def}}{\equiv}$	$\neg \Diamond\!\!\!\!\cdot\, \neg A$	All initial subintervals (including infinite)
A^*	$\overset{\text{def}}{\equiv}$	$A^\star \vee \left(A^\star \frown (A \wedge inf)\right)$	Conventional (weak) chop-star
A^ω	$\overset{\text{def}}{\equiv}$	$A^\star \wedge inf$	Chop-omega

Table 1: Some useful derived PITL operators

  – Interval σ has only one state (i.e., it is *empty*).
  – σ is finite and either itself satisfies A or can be split into a finite number of (finite-length) subintervals which share end-states (like chop) and all satisfy A.
  – $|\sigma| = \omega$ and σ can be split into ω finite-length intervals sharing end-states (like chop) and each satisfying A.
In this version of chop-star, each iterative subinterval has finite length. The third case is called **chop-omega** and denoted as A^ω.

As an example, we depict the behaviour of variable p in some 5-state interval σ and denote *true* and *false* by t and f, respectively:

	σ_0	σ_1	σ_2	σ_3	σ_4
p	t	f	t	f	t

This interval satisfies the following formulas:

$$p \qquad skip \frown \neg p \qquad p \wedge (true \frown \neg p) \qquad (p \wedge (skip \frown skip))^\star.$$

For instance, the formula $skip \frown \neg p$ is true because $\sigma_0 \sigma_1$ satisfies *skip* and $\sigma_1 \ldots \sigma_4$ satisfies $\neg p$ since $\sigma_1(p) = false$. The fourth formula is true because both $\sigma_0 \ldots \sigma_2$ and $\sigma_2 \ldots \sigma_4$ satisfy $p \wedge (skip \frown skip)$. The interval does not satisfy the formulas below:

$$\neg p \qquad skip \frown p \qquad true \frown (\neg p \wedge \neg(true \frown p)).$$

Table 1 shows some useful derived PITL operators, including the weak versions of chop $A; B$ and chop-star A^*. The derived construct $A \leftarrow B$ for temporal assignment in Table 1 perhaps requires some more explanation. Its purpose is to specify that the value of A in a

Axioms:

VPTL	$\vdash$	Substitution instances of valid PTL formulas
P2	$\vdash$	$(A^\frown B)^\frown C \;\equiv\; A^\frown(B^\frown C)$
P3	$\vdash$	$(A \vee A')^\frown B \;\supset\; (A^\frown B) \vee (A'^\frown B)$
P4	$\vdash$	$A^\frown(B \vee B') \;\supset\; (A^\frown B) \vee (A^\frown B')$
P5	$\vdash$	$empty^\frown A \;\equiv\; A$
P6	$\vdash$	$finite \;\supset\; (A^\frown empty \;\equiv\; A)$
P7	$\vdash$	$w \;\supset\; \boxdot w$
P8	$\vdash$	$\boxdot(A \supset A') \wedge \Box(B \supset B') \;\supset\; (A^\frown B \supset A'^\frown B')$
P9	$\vdash$	$A^\star \;\equiv\; empty \vee (A \wedge more)^\frown A^\star$
P10	$\vdash$	$A \wedge \Box\big(A \supset (B \wedge more)^\frown A\big) \;\supset\; B^\omega$

Inference Rules:

MP	$\vdash A \supset B, \;\; \vdash A \;\;\Rightarrow\;\; \vdash B$
$\boxdot$**FGen**	$\vdash finite \supset A \;\;\Rightarrow\;\; \vdash \boxdot A$
$\Box$**Gen**	$\vdash A \;\;\Rightarrow\;\; \vdash \Box A$
$\boxdot$**Aux**	$\vdash \boxdot\big((fin\ p) \equiv B\big) \supset A \;\;\Rightarrow\;\; \vdash A$

In $\boxdot$**Aux**, propositional variable p must not occur in A or B.

Table 2: Axiom system for PITL with finite and infinite time

finite interval's last state equals the value of B for the interval. For example, the formula $p \leftarrow \Box\, q$ is true on an interval iff either (a) the interval is infinite or (b) it is both finite and has one of the following hold for the propositional variables p and q:

- The (finite) interval's last state has p true and all states have q true.
- The (finite) interval's last state has p false and at least one state has q false.

Below are some sample valid PITL formulas:

$$(finite \wedge \boxdot A) \supset A \qquad skip^\star \qquad A^{\star\star} \equiv A^\star \qquad (w \wedge A)^\frown B \;\equiv\; w \wedge (A^\frown B)$$

$$\boxdot(A \wedge B) \equiv (\boxdot A \wedge \boxdot B) \qquad (\Box \boxdot A) \equiv (\boxdot \Box A) \qquad (\boxdot \boxdot A) \equiv \boxdot A$$

$$\diamondsuit A \wedge \diamondsuit B \;\equiv\; \diamondsuit(\diamondsuit A \wedge \diamondsuit B) \qquad \boxdot\big((fin\ p) \equiv A\big) \;\supset\; (\boxdot A) \equiv (\Box p).$$

Let **PTL** be the subset of PITL with just *skip* and the (derived) temporal operators $\bigcirc$ and $\diamondsuit$ shown in Table 1. We use X and X' for PTL formulas.

Although we do not need **existential quantification** in our proof, it is convenient to define here since it helps the exposition concerning automata-based ways to represent PITL formulas in §7.2, §7.4 and §10.2 and also assists us when we compare our approach with related proofs for logics with quantification in Section 11. The syntax is $\exists p.\, A$ for any propositional variable p and formula A. We let $\sigma \models \exists p.\, A$ be true iff $\sigma' \models A$ is true for some interval σ' identical to σ except possibly for p's behaviour. Existential quantification together with PITL yields **QPITL** and together with PTL yields **QPTL**.

2.1. **PITL Axiom System.** Table 2 shows the PITL axiom system with finite and infinite time. Axiom **VPTL** permits PITL substitution instances of valid PTL formulas with *skip*, $\bigcirc$ and $\diamondsuit$. For instance, from the valid PTL formula $\bigcirc p \supset \diamondsuit p$ follows $\vdash \bigcirc A \supset \diamondsuit A$, for any PITL formula A. Axiom **P10** gives an inductive way to introduce chop-omega. Our

Axioms:

Taut $\vdash$ Substitution instances of conventional (nonmodal) tautologies

F2 $\vdash$ $(A^\frown B)^\frown C \equiv A^\frown(B^\frown C)$

F3 $\vdash$ $(A \vee A')^\frown B \supset (A^\frown B) \vee (A'^\frown B)$

F4 $\vdash$ $A^\frown(B \vee B') \supset (A^\frown B) \vee (A^\frown B')$

F5 $\vdash$ $empty^\frown A \equiv A$

F6 $\vdash$ $A^\frown empty \equiv A$

F7 $\vdash$ $w \supset \boxdot\, w$

F8 $\vdash$ $\boxdot(A \supset A') \wedge \square(B \supset B') \supset (A^\frown B) \supset (A'^\frown B')$

F9 $\vdash$ $A^\star \equiv empty \vee (A \wedge more)^\frown A^\star$

F10 $\vdash$ $\bigcirc A \supset \text{\textcircled{w}}\, A$

F11 $\vdash$ $A \wedge \square(A \supset \text{\textcircled{w}}\, A) \supset \square A$

Inference Rules:

MP $\vdash A \supset B, \ \vdash A \ \Rightarrow \ \vdash B$

$\boxdot$**Gen** $\vdash A \ \Rightarrow \ \vdash \boxdot A$

$\square$**Gen** $\vdash A \ \Rightarrow \ \vdash \square A$

Note: $\text{\textcircled{w}}\, A \overset{\text{def}}{\equiv} \neg\bigcirc\neg A$ (Weak next)

Table 3: Axiom system for PITL with just finite time

new Inference Rule $\boxdot$**Aux** permits auxiliary variables to capture behaviour in finite-length prefix intervals and is only needed for infinite time.

The axiom system in Table 2 for both finite and infinite time is adapted from our earlier one [Mos04] for just finite time (see Table 3), itself based on a previous one we originally presented in [Mos94]. That axiom system contains some axioms of Rosner and Pnueli [RP86] for PITL without chop-star and our own axioms and inference rule for the operators $\boxdot$ (defined using weak chop in Table 1) and chop-star. The new PITL axiom system in Table 2 adapts the axioms for $\boxdot$ to use $\boxdot$ instead to shorten the completeness proof since $\boxdot$ works better with the strong chop operator $^\frown$.

For consistency with our usage here, the version of the earlier axiom system for just finite time given in Table 3 uses strong chop $^\frown$ instead of weak chop ";" and likewise uses $\boxdot$ instead of $\boxdot$. It therefore very slightly differs from the original one in [Mos04] in an inessential way since for finite time the two pairs of operators are indistinguishable. In [Mos04] we prove completeness by reduction to PTL.

Appendix A contains a large variety of representative PITL theorems, derived rules and their proofs. Many are used directly or indirectly in our completeness proof.

Note that Inference Rule $\boxdot$**FGen** in Table 2 for $\boxdot$ mentions *finite* in it, whereas the analogous Inference Rule $\square$**Gen** for $\square$ does not. A version of $\boxdot$**FGen** without *finite* and called $\boxdot$**Gen** can be deduced (see the derived inference rule **DR4** in Appendix A). If just finite time is permitted, the two variants $\boxdot$**FGen** and $\boxdot$**Gen** for $\boxdot$ are in practice identical since *finite* is valid and hence deducible by Axiom **VPTL**. In fact, our earlier axiom system for PITL with just finite time in Table 3 uses the version without *finite*.

2.2. **Theoremhood, Soundness and Axiomatic Completeness.** A formula A deducible from the axiom system is a **theorem**, denoted $\vdash A$. Additionally, a formula A is **consistent** if $\neg A$ is *not* a theorem, i.e., $\nvdash \neg A$. We claim the axiom system is **sound**, that is, $\vdash A$ *implies* $\vDash A$. A logic is **complete** if each valid formula is deducible as a theorem in the logic's axiom system. In other words, if $\vDash A$, then $\vdash A$. Our goal is to show **completeness** for PITL. However, we actually prove a stronger result which requires some further definitions and we therefore defer the formal statement until Theorem 3.2 in Section 3. We also make use of the following variant way of expressing axiomatic completeness:

Lemma 2.1 (Alternative notion of completeness)**.** *A logic's axiom system is complete iff each consistent formula is satisfiable.*

We often use the next Theorem 2.2 about finite time:

Theorem 2.2 (Completeness of PITL Axiom System for Finite Time)**.** *Any valid PITL implication finite* $\supset A$ *is deducible as a PITL theorem* $\vdash$ *finite* $\supset A$ *using the axiom system for PITL with both finite and infinite time in Table 2.*

Proof.. This readily follows by deducing the axioms and inference rules of our earlier complete axiom system for PITL with just finite time [Mos04] in Table 3. The axiom system and proofs of theorems are easily *relativised* to make finite time explicit and deduced with the new axiom system for both finite and infinite time already presented in Table 2. The relativisation can use the fact that the two axiom systems are quite similar. □

One can alternatively disregard Theorem 2.2 and instead treat our presentation as a self-contained *proof reducing completeness for PITL with both finite and infinite time to that for PITL with just finite time.*

2.3. **Summary of the Completeness Proof.** Our proof of axiomatic completeness for PITL establishes that any consistent PITL formula is satisfiable (see the earlier Lemma 2.1). The completeness proof makes use of a PITL subset called PTL^U (defined later in §5.2) which is a version of PTL having an *until* operator. As we discuss in §5.2, axiomatic completeness for PTL^U readily follows from axiomatic completeness for basic PTL so any consistent PTL^U formula is satisfiable.

The PITL completeness proof can be roughly summarised as ensuring that for any consistent PITL formula A, there exists a consistent PTL^U formula Y_0, which possibly contains auxiliary propositional variables, such that the PITL implication $Y_0 \supset A$ is deducible. Completeness for PTL^U guarantees that Y_0 is satisfiable. The soundness of the PITL axiom system then ensures that any model of Y_0 also satisfies A thereby showing axiomatic completeness for PITL. Note that in the actual proof, we use make use of a PTL^U conjunction $Y \wedge X$ in place of Y_0.

In the course of the PITL completeness proof, we also employ another PITL subset called $PITL^K$ (defined later in §5.3). It is a version of PITL without omega-iteration and serves as a kind of bridge between full PITL and PTL^U. The PITL completeness proof first obtains from the PITL formula A a $PITL^K$ formula K such that we can deduce $A \equiv K$. We then show how to obtain the PTL^U formula Y_0 such that the implication $Y_0 \supset K$ is deducible. We further show that if A is consistent, so are K and Y_0. Axiomatic completeness for PTL^U ensures that the consistent PTL^U formula Y_0 is satisfiable. The implication $Y_0 \supset K$ together with the deduced equivalence $A \equiv K$ guarantees the deducibility of the previously

mentioned PITL implication $Y_0 \supset A$. Hence, any model of Y_0 also satisfies A, thereby establishing completeness for PITL since every consistent PITL formula is indeed satisfiable.

Here is a very brief summary of the main reductions:

$$\text{PITL} \quad \xrightarrow{\text{Section 6}} \quad \text{PITL}^{\text{K}} \quad \xrightarrow{\text{Section 9}} \quad \text{PTL}^{\text{U}}.$$

Only the reduction from PITL$^{\text{K}}$ to PTL$^{\text{U}}$ requires some explicit automata-theoretic constructions which involve finite words and are expressed in temporal logic.

Below is the structure of our reduction from PITL to PTL$^{\text{U}}$:

- In Section 3 we describe a class of PITL theorems with useful substitution instances.
- In Section 4 we present lemmas for systematically replacing some of a formula's subformulas by others in proofs.
- In Section 5 we formally introduce the very simple PTL subset NL^1 as well as the subsets PTL$^{\text{U}}$ and PITL$^{\text{K}}$. Although PITL$^{\text{K}}$ lacks chop-omega, it still has the same expressiveness as PITL. We also describe three other classes of formulas called *right-chops*, *chain-formulas* and *auxiliary temporal assignments*.
- In Section 6 we show that any PITL formula is deducibly equivalent to one in PITL$^{\text{K}}$.
- In Section 7 we show how to represent semi-automata and automata in PITL.
- Section 8 utilises the material in the previous section to test for a given PITL formula in suffixes of a finite interval. Sections 7 and 8 provide a basis for introducing suitable auxiliary variables via auxiliary temporal assignments.
- In Section 9 we use the constructed auxiliary variables to reduce an arbitrary consistent PITL$^{\text{K}}$ formula K to one in PTL$^{\text{U}}$. Axiomatic completeness for PITL with infinite time then readily follows from this.

A large portion of the reasoning is done at the semantic level (for example, all of Section 8). We then employ axiomatic completeness for restricted versions of PITL (such as PITL with finite time) to immediately deduce the theoremhood of key properties expressible as valid formulas in these versions. This significantly shortens the completeness proof by reducing the amount of explicit deductions.

3. RIGHT-INSTANCES, RIGHT-VARIABLES AND RIGHT-THEOREMS

Before proceeding further, we need to introduce a class of PITL theorems for which suitable substitution instances are themselves deducible as theorems. Now in the completeness proof for PITL later on, if a deducible PITL formula has propositional variables not occurring in the left of chops or in chop-stars (e.g., p in the formula $p \supset \Diamond p$), then in each step of the formula's deduction these particular variables likewise do not occur in the left of chops or chop-stars. We define more generally for any PITL formula A and subformula B in A, a **right-instance** of B in A to be an instance of B which does not occur within the left of a chop or within some chop-star. Consider for example the disjunction below:

$$(p \frown \neg q) \ \vee \ (p \frown p') \ \vee \ (p \frown p')^{\star}. \tag{3.1}$$

The subformulas $\neg q$, and $(p \frown \neg q)$ as well as the leftmost occurrence of $p \frown p'$ are right-instances in the overall formula (3.1). However, all three occurrences of p and the rightmost occurrences of p' and $p \frown p'$ are not right-instances in (3.1) because each is either in the left of a chop or in a chop-star.

Now let a PITL formula A's **right-variables** be the (finite) set $RV(A)$ of A's variables which have only right-instances in A, that is, do not occur in the left of chops or chop-stars.

We now look at why the concept of right-variable is needed. In the formula $p \supset \Diamond p$, the variable p is a right-variable. Therefore, from the validity of $p \supset \Diamond p$, we can infer the validity of the substitution instance $skip \supset \Diamond skip$. Lemma 3.1, which is shortly presented, formalises this idea. However, if a variable is not a right-variable in a valid formula, we might incorrectly infer that a substitution instance of the formula is also valid. For instance, the variable p is not a right-variable in the formula $p \supset \boxdot p$ which is an instance of Axiom **P7** in Table 2. This formula is valid but the substitution instance $skip \supset \boxdot skip$ is not.

Now all propositional variables in a propositional formula with no temporal operators are right-variables of that formula. More generally, all propositional variables in a PTL formula are right-variables. In contrast, a chop-star formula has no right-variables.

The next simple lemma concerns substitution into right-variables in valid formulas:

Lemma 3.1 (Substitution Instances into Right-Variables). *Suppose A is a PITL formula, p is one of A's right-variables (i.e., in $RV(A)$) and B is some PITL formula. Then if A is valid, so is the substitution instance A_p^B.*

Proof by contradiction.. Let q be a variable not occurring in A or B and let C be a variant of A with all instances of p replaced by q (i.e., A_p^q). The variable p is a right-variable of A so q is similarly a right-variable of C. It follows by induction on A's syntax that A_p^B and C_q^B denote exactly the same PITL formula. Consequently, in our reasoning about A_p^B, we can assume without loss of generality that p itself does not occur in B. This is because we can view A_p^B as being C_q^B.

Now suppose by contradiction that A_p^B is not valid. By our previous discussion, also assume that p does not occur in B. Then some interval σ satisfies $\neg(A_p^B)$. We construct a variant σ' in which the value of variable p in each state σ'_i equals true iff the suffix subinterval $\sigma_{i\uparrow}$ satisfies B. Hence $\sigma' \vDash \Box(p \equiv B)$ and $\sigma' \vDash \neg(A_p^B)$. It readily follows from this and p being a right-variable that σ' satisfies $\neg A$ since A_p^B only examines B in suffix subintervals. From $\sigma' \vDash \neg A$ we have that A is not valid. $\square$

Later in Section 6, our completeness proof will need a deductive analogue of the semantically oriented Lemma 3.1 to permit us to infer from a theorem A and right-variable p in $RV(A)$ another theorem A_p^B. One way to achieve this is by adding the next inference rule to the PITL axiom system in Table 2 for any formula A and variable p in $RV(A)$:

$$\vdash A \quad \Rightarrow \quad \vdash A_p^B. \tag{3.2}$$

Another possibility is an analogue of Inference Rule $\boxdot$**Aux** in Table 2:

$$\vdash \Box(p \equiv B) \supset A \quad \Rightarrow \quad \vdash A,$$

where the propositional variable p does not occur in A or B. However, it turns out that these are unnecessary since the axiom system in its current form is already sufficient to allow a suitable class of such substitutions. We now present a formal basis for this.

A PITL formula A which is theorem (i.e., $\vdash A$) is called a **right-theorem** (denoted $\vdash_{\mathrm{rt}} A$) if there exists a deduction of A in which A's right-variables never occur on the left of chop or in chop-star in any proof steps. However, any of A's variables not in $RV(A)$ as well as any subsequently introduced auxiliary variables in the deductions are permitted to appear in some deduction steps in the left of chops or chop-stars. For example, if p is a right-variable of A, then no proof step can use p with Axiom **P7** (e.g., $\vdash p \supset \boxdot p$) since p is not a right-variable here owing to $\boxdot p$.

The completeness proof for PITL will ensure that any valid PITL formula A is indeed deducible as a right-theorem. We will refer to this here as **right-completeness**. Below is our main theorem for axiomatic completeness of PITL using right-completeness:

Theorem 3.2 (Right-Completeness of PITL Axiom System). *Any valid PITL formula A is a right-theorem of the axiom system, that is, if $\vDash A$, then $\vdash_{\mathrm{rt}} A$.*

The proof of this, our main result, is described later and concludes in Section 9.

Right-theoremhood naturally yields the dual notion of **right-consistency** of a PITL formula A, that is, not $\vdash_{\mathrm{rt}} \neg A$. Our completeness proof for PITL can therefore be regarded as not only showing that valid PITL formulas are right-theorems but also that any right-consistent PITL formula is satisfiable (compare with Lemma 2.1).

As already pointed out, the main reason we are interested in right-theorems is that suitable substitution instances of them are PITL theorems. Our need for this occurs when in Section 6 we reduce right-completeness for PITL to right-completeness for its subset PITL$^{\mathrm{K}}$ without chop-omega. The lemma below formalises the substitution process:

Lemma 3.3 (Substitution Instances of Right-Theorems). *Let A and $B_1, \ldots, B_n$ be PITL formulas and $p_1, \ldots, p_n$ be some of A's right-variables. If A is a right-theorem, then so is the substitution instance $A_{p_1,\ldots,p_n}^{B_1,\ldots,B_n}$, that is, $\vdash_{\mathrm{rt}} A_{p_1,\ldots,p_n}^{B_1,\ldots,B_n}$.*

Proof.. We assume that auxiliary variables in A's proof (i.e., ones not in V_A) do not occur in $B_1, \ldots, B_n$. In each step of A's proof, we replace each p_i by B_i to obtain $\vdash_{\mathrm{rt}} A_{p_1,\ldots,p_n}^{B_1,\ldots,B_n}$. □

Many PITL theorems in Appendix A can be checked to be right-theorems by inspection of the proof steps. For example, those with no right-variables are immediate right-theorems. We have not indicated in the appendix which theorems are right-theorems and will normally only designate formulas as right-theorems in the completeness proof when this is needed.

The next lemma concerns the relationship between derived rules and right-theorems:

Lemma 3.4 (Right-Theorems from Some Derived Rules). *Suppose the assumptions of a derived rule which deduces some PITL formula A are right-theorems. Furthermore, suppose that in the derived rule's own proof of A, none of A's right-variables occur on the left of chop or in chop-star (including in any nested deduced PITL theorems and derived rules). If A's right-variables are a subset of the union of the assumptions' right-variables, then A itself is a right-theorem.*

We omit the proof. For example, Derived Rule **DR13** in Appendix A (see also the abbreviated Table 4 found later in §7.4) lets us infer from the theorem $\vdash \Box A \supset B$ the theorem $\Box A \supset \Box B$. It only requires the kind of reasoning mentioned in Lemma 3.4. Consequently, from $\vdash_{\mathrm{rt}} \Box A \supset B$ we can infer $\vdash_{\mathrm{rt}} \Box A \supset \Box B$.

Readers are strongly encouraged to initially try to understand our completeness proof without consideration of right-theoremhood by simply viewing it as ordinary theoremhood and ignoring the prefix "right-". This can even be rigorously done by assuming that the optional inference rule (3.2) is part of the PITL axiom system. A subsequent, more thorough study of the material can then better take right-theoremhood into account. Indeed, we can then regard our completeness proof as two parallel proofs, a simpler one with (3.2) and another more sophisticated one which is based on right-theoremhood and Lemma 3.3 and hence does not assume (3.2). Incidentally, our completeness proof ultimately ensures that (3.2) is obtainable as a derived inference rule even if it is not in the axiom system.

4. Some Lemmas for Replacement

We now consider some techniques used in the completeness proof to replace selected right-instances in a PITL formula by other formulas.

Lemma 4.1. *Let A_1, A_2, B_1 and B_2 be PITL formulas. If A_2 can be obtained from A_1 by replacing zero or more right-instances of B_1 in A_1 by B_2, then the next implication is deducible as a right-theorem:*

$$\vdash_{\mathrm{rt}} \quad \Box(B_1 \equiv B_2) \quad \supset \quad A_1 \equiv A_2.$$

Proof.. The proof involves induction on the syntax of formula A_1, with each instance of B_1 regarded as atomic. We consider the cases when A_1 is B_1 itself, *true*, a propositional variable p, $\neg C$, $C_1 \vee C_2$, *skip*, $C_1 {^\frown} C_2$, and $C^{\star}$. The first three of these involve quite routine conventional propositional reasoning. The case for *skip* is trivial since A_1 and A_2 are identical. The case for chop-star is likewise trivial since this lemma does not permit replacement in its scope.

For the case for chop, assume A_1 and A_2 have the forms $C_1 {^\frown} C_2$ and $C_1 {^\frown} C_2'$, respectively. Note that no replacements are done in the left of chop. By induction on A_1's syntax, we deduce the next implication:

$$\vdash_{\mathrm{rt}} \Box(B_1 \equiv B_2) \quad \supset \quad C_2 \equiv C_2'.$$

This and PTL reasoning (see Derived Rule **DR13** in Appendix A and also in the abbreviated Table 4 found later in §7.4) yields the implication below:

$$\vdash_{\mathrm{rt}} \Box(B_1 \equiv B_2) \quad \supset \quad \Box(C_2 \equiv C_2').$$

Lemma 3.4 ensures that our use here of Derived Rule **DR13** indeed yields a right-theorem.

We can also deduce the next implication using Axiom **P8** and some further temporal reasoning (see PITL Theorem **T3** in Appendix A and also in Table 4 in §7.4):

$$\vdash_{\mathrm{rt}} \Box(C_2 \equiv C_2') \quad \supset \quad (C_1 {^\frown} C_2) \equiv (C_1 {^\frown} C_2').$$

These two implications together yield our goal below:

$$\vdash_{\mathrm{rt}} \Box(B_1 \equiv B_2) \quad \supset \quad (C_1 {^\frown} C_2) \equiv (C_1 {^\frown} C_2').$$

This concludes Lemma 4.1's proof. $\square$

Lemma 4.1 yields a derived inference rule for **Right Replacement** of formulas:

Lemma 4.2 (Right Replacement Rule). *Let A_1, A_2, B_1 and B_2 be PITL formulas. Suppose that A_2 can be obtained from A_1 by replacing zero or more right-instances of B_1 in A_1 by B_2. If B_1 and B_2 are deducibly equivalent as a right-theorem (i.e., $\vdash_{\mathrm{rt}} B_1 \equiv B_2$), then so are A_1 and A_2.*

Proof.. By Lemma 4.1, we deduce the next implication:

$$\vdash_{\mathrm{rt}} \quad \Box(B_1 \equiv B_2) \quad \supset \quad A_1 \equiv A_2.$$

Also, $\vdash_{\mathrm{rt}} B_1 \equiv B_2$ and Inference Rule $\Box$**Gen** yield $\vdash_{\mathrm{rt}} \Box(B_1 \equiv B_2)$. Then modus ponens yields $\vdash_{\mathrm{rt}} A_1 \equiv A_2$. $\square$

5. Useful Subsets of PITL

We now describe five subsets of PITL and some associated properties which will be extensively used later on in different parts of the PITL completeness proof. We have chosen to collect material about the subsets here instead of introducing each subset as the need arises. This should make it easier for readers to review the definitions and features when required and also make the main steps of the completeness proof shorter and more focused. In addition, when taken as a whole, the combined presentation of the PITL subsets enables us to give a technical overview of some of the proof steps encountered. Table 5 later lists variables used for the subsets and other subsequently defined categories.

5.1. **PTL with only Unnested Next Constructs.** Let $\mathbf{NL}^1$ denote the subset of PTL formulas in which the only temporal operators are unnested $\bigcirc$s (e.g., $p \vee \bigcirc \neg p$ but not $p \vee \bigcirc \bigcirc \neg p$). It is not hard to see that NL^1 formulas only examine an interval's first two states. They are therefore useful for describing automata transitions from one state to the next. The variables T and T' denote formulas in NL^1.

Below are some theorems which contain NL^1 formulas and are required in the completeness proof. None of these theorems are themselves in NL^1. The proofs are in Appendix A.

$$
\begin{array}{llll}
\mathbf{T62} & \vdash & \diamondsuit(\mathit{more} \wedge T) & \equiv & \mathit{more} \wedge T \\
\mathbf{T68} & \vdash & \diamondsuit(\mathit{skip} \wedge T) & \equiv & \mathit{more} \wedge T \\
\mathbf{T69} & \vdash & (\mathit{skip} \wedge T)^\frown A & \equiv & T \wedge \bigcirc A
\end{array}
$$

5.2. **PTL with Until.** Recall that for our purposes we define PTL to be the subset of PITL with just skip and the derived temporal operators $\bigcirc$ and $\diamondsuit$ shown in Table 1.

We also use a more expressive version of PTL denoted here as $\mathbf{PTL^u}$ with a *strong* version of the standard temporal operator until, derivable in PITL:

$$
T \ \mathit{until} \ A \quad \overset{\mathrm{def}}{\equiv} \quad (\mathit{skip} \wedge T)^{\star\frown} A.
$$

We limit until's lefthand operand to be a formula in NL^1 (defined previously in §5.1). Note that this definition of until using chop and chop-star results in any variable in the left operand of until not being a right-variable. Let Y and Y' denote PTL^U formulas.

We establish right-completeness for PITL by a reduction to PTL^U, instead of directly to PTL. It is not hard to show that our axiom system is complete for PTL^U formulas. This is because we can deduce the next two PTL^U axioms known to capture this kind of until's behaviour (the PITL proofs are in Appendix A):

$$
\mathbf{T70} \ \vdash \ T \ \mathit{until} \ A \ \equiv \ A \vee \big(T \wedge \bigcirc(T \ \mathit{until} \ A)\big) \qquad \mathbf{T71} \ \vdash \ T \ \mathit{until} \ A \ \supset \ \diamondsuit A.
$$

Consequently, we can reduce completeness for PTL^U to it for PTL. In fact every PTL^U theorem is a right-theorem. This is because the right-variables in $T \ \mathit{until} \ A$ remain so in **T70** and **T71**, Hence, the two PTL^U axioms ensure that these variables remain right-variables in the proof steps for deducing a PTL^U theorem in the PITL axiom system. See Kröger and Merz [KM08] for more about axioms for a variety of such binary temporal operators.

5.3. **PITL without Omega-Iteration.** Our completeness proof includes a step in which any chop-omega (defined in Table 1) is eliminated by re-expressing any chop-star not in the left of chop or another chop-star. This exploits a convenient alternative characterisation of omega-regular languages described by Thomas at the end of [Tho79] which does not involve omega-iteration. It instead employs *closure* under some other operations which include *complementation*:

Theorem 5.1 (Omega-Regularity using Closures). *The omega-regular languages of an alphabet Σ are exactly the closure of $\{\emptyset\}$ under the following: (1) union, (2) complementation (with respect to Σ^ω) and (3) left concatenation by Σ's regular languages.*

Here $\emptyset$ denotes the omega-language with no elements.

Let $\mathbf{PITL^K}$ denote the PITL subset in which chop-star only occurs on the left of chops (like (3) in Thomas' theorem above) and is therefore restricted to finite intervals. The K in PITL^K stands for "Kleene star". For example, the next two formulas are in PITL^K:

$$(skip \wedge p)^{\star\frown}q \qquad (skip^{\star\frown}skip) \vee \bigcirc p.$$

In contrast, the two formulas below are not in PITL^K:

$$(skip \wedge p)^{\star} \qquad p \supset \diamondsuit(skip \wedge q)^{\star}.$$

Observe that a PITL^K formula can contain chop-star subformulas, which by the definition of PITL^K are not themselves in it. An example is $(skip \wedge p)^{\star}$ in $(skip \wedge p)^{\star\frown}q$.

With just finite time, any PITL formula A is easily re-expressed in PITL^K as $A^\frown empty$ (compare with Axiom **P6** in Table 2). However this technique does not work for infinite time. We also need Thomas' theorem (Theorem 5.1) to ensure that any PITL formula A has a semantically equivalent PITL^K formula K for both finite and infinite time (i.e., $\vDash A \equiv K$). For example, one way to re-express the PITL formula $(skip \wedge p)^{\star}$ in PITL^K is $\square(more \supset p)$. It follows that any chop-omega formula is re-expressible in PITL^K. For instance, for any PITL formula B, the formula $(skip \wedge B)^\omega$ is semantically equivalent to $\square \diamondsuit(skip \wedge B)$.

Later on in Section 6 we employ Thomas' theorem to easily reduce axiomatic completeness for PITL to that for PITL^K. More precisely, we will formally establish there that for any PITL formula A, there exists a semantically equivalent PITL^K formula K such that the formula $A \equiv K$ is deducible as a PITL theorem. Hence, by simple propositional reasoning, if A is consistent, so is K and any model for K is also one for A. The remainder of the overall completeness proof then reduces completeness for PITL^K to it for PTL^U.

Choueka and Peleg [CP83] give a simpler proof of Thomas' theorem using standard deterministic omega automata. Readers favouring an automata-theoretic perspective can therefore regard the theorem in the context of PITL as a basis for *implicitly* determinising the original PITL formula, resulting in a semantically equivalent one in PITL^K.

5.4. **Right-Chops and Chain Formulas.** For any PITL formula A, we call a chop formula in A a **right-chop** if it is not in another chop's left operand or in a chop-star. Right-chops help reduce PITL^K to PTL^U. We illustrate them with the formula below:

$$((p^\frown p')^\frown \neg(q^\frown q')) \vee (p^\frown p'). \tag{5.1}$$

The following three formulas all occur as right-chops in this:

$$(p^\frown p')^\frown \neg(q^\frown q') \qquad q^\frown q' \qquad p^\frown p'.$$

Only the second instance of $p \frown p'$ in formula (5.1) is a right-chop. In contrast, the first instance of $p \frown p'$ is not a right-chop since it is within the left operand of another chop. Observe that the right-chops of a PITL formula A are exactly those subformulas in A, including possibly A itself, which have chop as their main operator and are right-instances (previously defined in Section 3).

In addition to right-chops, the reduction of a PITL^{K} formula to PTL^{U} employs a class of PTL^{U} formulas involving disjunctions and sequential **chains** of restricted constructs. Let a **chain formula** be any PTL^{U} formula with the syntax below, where w is a state formula, T is an NL^1 formula and G and G' are themselves chain formulas:

$$empty \qquad w \wedge G \qquad G \vee G' \qquad T \, until \, G.$$

The operator *until* in chain formulas involves a quite limited version of the PITL operator chop-star which is much easier to reason about than full chop-star. The next lemma exploits this and shows that a chop in which the left operand is a chain formula and the right one is in PTL^{U} can be re-expressed as a deducibly equivalent PTL^{U} formula.

Lemma 5.2. *For any chain formula G and PTL^{U} formula Y, there exists some PTL^{U} formula Y' such that the equivalence $(G \frown Y) \equiv Y'$ is deducible as a right-theorem.*

Proof.. We do induction on G's syntax using the deducible equivalences below in which w is a state formula, T is an NL^1 formula and G' and G'' are themselves chain formulas:

$$\vdash_{\text{rt}} \quad empty \frown Y \; \equiv \; Y \qquad\qquad \vdash_{\text{rt}} \quad (G' \vee G'') \frown Y \; \equiv \; (G' \frown Y) \vee (G'' \frown Y)$$
$$\vdash_{\text{rt}} \quad (w \wedge G') \frown Y \; \equiv \; w \wedge (G' \frown Y) \qquad \vdash_{\text{rt}} \quad (T \, until \, G') \frown Y \; \equiv \; T \, until \, (G' \frown Y).$$

The first of these is an instance of PITL Axiom **P5**. The second and third are respective instances of PITL Theorems **T42** and **T18** in Appendix A (see also the abbreviated Table 4 found later in §7.4). The fourth uses the earlier ITL-based definition of the temporal operator *until* in §5.2 and Axiom **P2** which itself concerns chop's associativity. $\square$

For example, the left chop operand in the PITL formula $\big(p \wedge (q \, until \, empty) \big) \frown skip$ is a chain formula. The chop itself is deducibly equivalent to the PTL^{U} formula $p \wedge (q \, until \, skip)$.

Our completeness proof will ultimately apply Lemma 5.2 when in Section 9 we later replace the left operands of a consistent PITL^{K} formula's right-chops with chain formulas. For this to work, we will also need auxiliary variables of the kind now described.

5.5. **Auxiliary Temporal Assignments.** When we later represent automata runs in PITL, it is convenient to generalise formulas of the form $p \leftarrow B$ (the temporal assignment construct defined in Table 1) to conjunctions of several of these. Please refer back to Section 2 for a brief explanation about the meaning of temporal assignment. We call such a conjunction an **Auxiliary Temporal Assignment (ATA)**. It has the form given below:

$$\bigwedge_{1 \leq i \leq n} (q_i \leftarrow A_i),$$

for some $n \geq 0$, where each A_i is a PITL formula, there are n distinct **auxiliary** propositional variables $q_1, \ldots q_n$ and the only ones of them permitted in each A_i are $q_1, \ldots q_{i-1}$. All other propositional variables are allowed in any A_i. Here is a sample ATA with one nonauxiliary variable r and two auxiliary variables p and q:

$$(p \leftarrow \bigcirc r) \wedge (q \leftarrow \square(r \supset \diamondsuit p)).$$

Variables such as D and D' denote ATAs. Two ATAs are **disjoint** if they have distinct auxiliary variables.

Let us now look at how to formally introduce ATAs containing auxiliary variable into deductions for later use within the completeness proof in §9.2.

Lemma 5.3 (Temporal Operators $\boxdot$, $\leftarrow$ and Right-Consistency). *Let A and B be PITL formulas with no instances of propositional variable p. If A is right-consistent, so is the conjunction $A \,\wedge\, \boxdot(p \leftarrow B)$.*

Proof by contradiction.. Suppose $A \,\wedge\, \boxdot(p \leftarrow B)$ is not right-consistent. Then $\boxdot(p \leftarrow B) \supset \neg A$ is a right-theorem. We re-express $\boxdot(p \leftarrow B)$ as $\boxdot\big((\textit{fin } p) \equiv B\big)$. By this and Inference Rule $\boxdot\mathbf{Aux}$, the formula $\neg A$ is a right-theorem. Therefore A is not right-consistent. $\qquad\square$

Lemma 5.3 readily generalises to reduce a formula's right-consistency to that for a conjunction of it and a suitable ATA:

Lemma 5.4 (The Temporal Operator $\boxdot$, ATAs and Right-Consistency). *Let A be a PITL formula and D an ATA with no auxiliary variables in A. If A is right-consistent, so is the formula $A \,\wedge\, \boxdot D$.*

Proof.. For some $n \geq 0$, the ATA D contains n auxiliary variables and has the form $\bigwedge_{1 \leq i \leq n}(q_i \leftarrow B_i)$. We first apply Lemma 5.3 n times to reduce the formula A's right-consistency to that for the next formula:

$$A \,\wedge\, \bigwedge_{1 \leq i \leq n} \boxdot(q_i \leftarrow B_i). \tag{5.2}$$

The conjunction of $\boxdot$-formulas is then re-expressed with a single $\boxdot$ (see PITL Theorem **T28** found in Appendix A and also included in the more abbreviated Table 4 later in §7.4) to obtain the formula below which is deducibly equivalent to (5.2):

$$A \,\wedge\, \boxdot\big(\bigwedge_{1 \leq i \leq n} q_i \leftarrow B_i\big).$$

This is the same as our goal $A \,\wedge\, \boxdot D$. $\qquad\square$

5.6. **Overview of Role of PITL Subsets in Rest of Completeness Proof.** The PITL completeness proof can now be summarised using the PITL subsets just presented. *Some readers may prefer to skip this material and proceed directly to the proof which starts in Section 6.* Our goal here is to show that any right-consistent PITL formula A is satisfiable. Here is an informal sequence of the transformations involved:

$$A \xrightarrow{\text{Section 6}} K \xrightarrow{\text{Section 9}} K' \wedge \boxdot D' \xrightarrow{\text{Section 9}} Y \wedge X,$$

where K is a PITL^K formula, K' is a PITL^K formula in which the left operands of all right chops are chain formulas, D' is an ATA and Y and X are respectively in PTL^U and PTL. If A is right-consistent, then so are the formulas in all steps. From the completeness of the PTL^U axiom system as discussed in §5.2 we have that the conjunction $Y \wedge X$ is satisfiable. Furthermore, our techniques ensure that the models of a formula obtained from one of the transformations also satisfy the immediately preceding formula and hence by transitivity the original PITL formula A as well.

Important automata-theoretic techniques presented in Sections 7 and 8 help with the reductions to $K' \wedge \boxdot D'$ and $Y \wedge X$ in Section 9. We show in Section 9 that the formulas $K \wedge \boxdot D'$, $K' \wedge \boxdot D'$ and $Y \wedge X$ are deducibly equivalent.

Note that in the actual completeness proof (in Lemma 9.4 in §9.2), which for technical reasons involves a sequence of transformations from K to K', we make use of a PITL^K formula denoted K'_{m+1} rather than simply K'.

6. Reduction of Chop-Omega

If we assume right-completeness for PITL^K (later proved as Lemma 9.4 in §9.2), then obtaining from a PITL formula a deducibly equivalent PITL^K one is relatively easy. We first look at re-expressing chop-omega formulas in PITL^K and then extend this to arbitrary PITL formulas.

Lemma 6.1 (Deducible Re-Expression of Chop-Omega in PITL^K). *Suppose we have right-completeness for PITL^K. Then for any PITL formula B, there exists a PITL^K formula K with the same variables and no right-variables and for which the equivalence $K \equiv B^\omega$ is a right-theorem (i.e., $\vdash_{rt} K \equiv B^\omega$).*

Proof.. Thomas' theorem (Theorem 5.1) ensures that there exists some PITL^K formula which is semantically equivalent to B^ω and contains the same variables. From that formula we obtain one denoted here as K which has no right-variables by conjoining a trivially true $\diamondsuit$-formula containing a disjunction of all of B's variables and their negations. We therefore have $\vDash K \equiv B^\omega$ and now deduce $\vdash_{rt} K \equiv B^\omega$:

Case for showing $\vdash_{rt}$ $K \supset B^\omega$:

The first step involves an instance of Axiom **P10**:

$$\vdash_{rt} \quad K \ \wedge \ \square(K \supset (B \wedge more)^\frown K) \quad \supset \quad B^\omega. \tag{6.1}$$

In addition, the next formula is valid:

$$\vDash \quad B^\omega \supset (B \wedge more)^\frown B^\omega.$$

From this and $\vDash K \equiv B^\omega$, we have $\vDash K \supset (B \wedge more)^\frown K$. We then use the assumed right-completeness of PITL^K to deduce the implication as a right-theorem. Now invoke $\square$-generalisation (Axiom $\square$**Gen**) on this to obtain $\vdash_{rt} \square(K \supset (B \wedge more)^\frown K)$. Simple propositional reasoning involving that and the earlier deduced implication (6.1) establishes our immediate goal $\vdash_{rt} K \supset B^\omega$.

Case for showing $\vdash_{rt}$ $B^\omega \supset K$:

Let p be a propositional variable not in B^ω or K. The next formula is valid (and an instance of Axiom **P10**):

$$\vDash \quad p \ \wedge \ \square(p \supset (B \wedge more)^\frown p) \quad \supset \quad B^\omega.$$

We then replace B^ω by the semantically equivalent K:

$$\vDash \quad p \ \wedge \ \square(p \supset (B \wedge more)^\frown p) \quad \supset \quad K. \tag{6.2}$$

Now K is a PITL^K formula and furthermore $(B \wedge more)^\frown p$ is as well since even if B does contain some chop-stars, B is located within the left of a chop. The valid formula (6.2) is in PITL^K and hence a right-theorem by the assumed right-completeness for PITL^K:

$$\vdash_{rt} \quad p \ \wedge \ \square(p \supset (B \wedge more)^\frown p) \quad \supset \quad K.$$

Therefore, we can use Lemma 3.3 to obtain the theoremhood of the next PITL implication which has the formula B^ω substituted into the right-variable p:

$$\vdash_{rt} \quad B^\omega \wedge \square\big(B^\omega \supset (B \wedge more)^\frown B^\omega\big) \quad \supset \quad K. \tag{6.3}$$

We also deduce the following from the definition of chop-omega in terms of chop-star together with Axiom **P9** and some simple temporal reasoning:

$$\vdash_{\mathrm{rt}} \quad B^\omega \;\supset\; (B \wedge \mathit{more})^\frown B^\omega.$$

We now do $\Box$-generalisation (Axiom $\Box$**Gen**) on this and then use propositional reasoning on it with the previous formula (6.3) to obtain the right-theorem $\vdash_{\mathrm{rt}} B^\omega \supset K$, which is our immediate goal. $\qquad\Box$

Lemma 6.2 (Reduction of PITL to PITL$^{\mathrm{K}}$)**.** *If right-completeness holds for PITL$^{\mathrm{K}}$, then for any PITL formula A, there exists an equivalent PITL$^{\mathrm{K}}$ formula K with exactly the same propositional variables and right-variables such that $\vdash_{\mathrm{rt}} A \equiv K$.*

Proof.. We first re-express each of A's chop-stars $B_i^\star$ not in the left of chop or another chop-star using the next deducible equivalence (see PITL Theorem **T58** found in Appendix A and also included in the more abbreviated Table 4 in §7.4):

$$\vdash_{\mathrm{rt}} \quad B_i^\star \quad\equiv\quad (B_i^{\star\frown} \mathit{empty}) \vee B_i^\omega. \tag{6.4}$$

This splits $B_i^\star$ into cases for finite and infinite time. Note that there there are no right-variables in (6.4) since any variables occur in a chop-star. Hence the equivalence, once deduced, is trivially a right-theorem.

Lemma 6.1 ensures some PITL$^{\mathrm{K}}$ formula K_i' exists with the same variables as B_i, no right-variables and the right-theorem $\vdash_{\mathrm{rt}} K_i' \equiv B_i^\omega$. Hence like (6.4), the next equivalence is a right-theorem and both sides have the same variables and no right-variables:

$$\vdash_{\mathrm{rt}} \quad B_i^\star \quad\equiv\quad (B_i^{\star\frown} \mathit{empty}) \vee K_i'.$$

Then Right Replacement (Lemma 4.2) in A of each $B_i^\star$ by $(B_i^{\star\frown} \mathit{empty}) \vee K_i'$ yields a PITL$^{\mathrm{K}}$ formula K which the same variables as A and equivalent to it (i.e., $\vdash_{\mathrm{rt}} A \equiv K$). No right-variables in A are in any replaced $B_i^\star$. Hence A and K have the same right-variables. $\qquad\Box$

7. Deterministic Finite-State Semi-Automata And Automata

The remainder of our axiomatic completeness proof for PITL mostly concerns reducing PITL$^{\mathrm{K}}$ to PTL$^{\mathrm{U}}$. Now PITL with finite time expresses the regular languages and can readily encode regular expressions (see for example [Mos04] which reproduces our results with J. Halpern in [Mos83a]). We can therefore employ some kinds of *deterministic finite-state semi-automata and automata* which provide a convenient low-level framework for finite time to encode the behaviour of an arbitrary PITL formula. Our completeness proof utilises these semi-automata and automata to build a variant semi-automaton discussed in the next Section 8 to assist in reducing PITL formulas on the left of right-chops to chain formulas in PTL$^{\mathrm{U}}$. The reduction applying these techniques to go from PITL$^{\mathrm{K}}$ to PTL$^{\mathrm{U}}$ is presented in Section 9.

After introducing the semi-automata and automata, we will consider various semantically equivalent ways to represent them in temporal logic, each with its benefits. Some require PITL and others just PTL. The representations in PITL are at a higher level and fit well with our proof system, especially since we can assume completeness for PITL with finite time. In some later sections, we consider deducing some of the properties as theorems.

In order to define an **alphabet** for our semi-automata and automata, we introduce a special kind of state formula which serves as a **letter** and is called here an **atom**. An

atom is any finite conjunction in which each conjunct is some propositional variable or its negation and no two conjuncts share the same variable. The Greek letters α and β denote an individual atom. For any finite set of propositional variables V, let $\boldsymbol{\Sigma_V}$ be some set of $2^{|V|}$ logically distinct atoms containing exactly the variables in V. For example, if $V = \{p, q\}$, we can let Σ_V be the set of the four atoms shown below:

$$p \wedge q \qquad p \wedge \neg q \qquad \neg p \wedge q \qquad \neg p \wedge \neg q.$$

One simple convention is to assume that the propositional variables in an atom occur from left to right in lexical order. If V is the empty set, then Σ_V contains just the formula *true*.

A finite, nonempty sequence of atoms form a **word**. Each possible word corresponds to some collective state-by-state behaviour of the selected variables in a finite interval. For our interval-oriented application of words we never utilise the word containing no letters (commonly denoted ϵ in the literature).

7.1. Deterministic Finite-State Semi-Automata.

We define a **deterministic finite-state semi-automaton** S to be a quadruple $(V_S, Q_S, q_S^I, \delta_S)$ consisting of a finite set of propositional variables V_S, together with a finite, nonempty set of **control states** $Q_S = \{q_1, \ldots, q_m\}$, an **initial control state** $q_S^I \in Q_S$ and a **deterministic transition function** $\delta_S \colon Q_S \times \Sigma_{V_S} \to Q_S$. The sets V_S and Q_S must be disjoint, i.e., $V_S \cap Q_S = \emptyset$. We use propositional variables $q_1, \ldots, q_m$ to denote control states since this helps when expressing the semi-automaton's behaviour in PITL. A **run** on a finite word $\alpha_1 \ldots \alpha_k$ in $\Sigma_{V_S}^+$ with k atoms is a sequence of k control states $q_1' \ldots q_k'$ all in Q_S with $q_1' = q_S^I$ and $\delta_S(q_i', \alpha_i) = q_{i+1}'$ for each $i \colon 1 \le i < k$. Hence the semi-automaton makes just $k - 1$ transitions and consequently ignores the details of the last atom α_k. Therefore the semi-automaton differs from a conventional automaton which would have a run with $k + 1$ control states involving k transitions and the examination of all k atoms. Furthermore, the definition of a semi-automaton has no set of *final control states* and hence no *acceptance condition*. We abbreviate the set of atoms Σ_{V_S} as Σ_S since the elements of Σ_{V_S} serve as S's letters.

The semi-automaton S's behaviour is expressible in temporal logic by regarding each control state q_i to be a propositional variable which is true when q_i is S's current control state. Before showing how S's runs are expressed in PTL, we first define a state formula $init_S$ which ensures that the initial control state is q_S^I and also a transitional formula T_S in NL[1] which captures the behaviour of δ_S:

$$init_S \colon q_S^I \wedge \bigwedge_{q \in Q_S \colon q \neq q_S^I} \neg q$$

$$T_S \colon \bigwedge_{q \in Q_S} \left((\bigcirc q) \equiv \bigvee_{q' \in Q_S} \bigvee_{\alpha \in \Sigma_S \colon \delta_S(q', \alpha) = q} (q' \wedge \alpha) \right).$$

If we assume finite time, then a run starting at S's initial control state is expressed as the PTL formula $init_S \wedge \Box(more \supset T_S)$ or alternatively as the chain formula $init_S \wedge (T_S \ until \ empty)$ in PTL[U].

7.2. Deterministic Finite-State Automata.

Semi-automata do not have an acceptance test and hence do not have associated *accepting* runs. We therefore now define a **deterministic finite-state automaton** which includes an acceptance test. As we shortly illustrate, this can be constructed to recognise a given PITL formula in a finite interval. Let M be a quintuple $(V_M, Q_M, q_M^I, \delta_M, \tau_M)$. The first four entries are as for a semi-automaton. The

last entry $\tau_M \colon Q_M \to 2^{\Sigma_M}$ is a **conditional acceptance function** from control states to sets of letters. A run is the same as for a semi-automaton. Our notion of acceptance of a word does not use a conventional *set of final control states* but instead has the function τ_M make all control states **conditionally final**. An **accepting run** on a finite word $\alpha_1 \ldots \alpha_k$ in Σ_M^+ with k atoms is any run of k control states $q'_1 \ldots q'_k$ with $q'_k \in \tau_M(\alpha_k)$. Therefore, a control state $q \in Q_M$ is regarded as a final one only when the automaton sees an atom α with $\alpha \in \tau_M(q)$. A test for this is expressible as the state formula acc_M defined below:

$$acc_M \colon \quad \bigvee_{q \in Q_M} \bigvee_{\alpha \in \tau_M(q)} (q \wedge \alpha).$$

If we assume finite time, an accepting run of M starting at M's initial control state is expressed as the PTL formula $init_M \wedge \Box(more \supset T_M) \wedge fin\, acc_M$ or alternatively as the chain formula $init_M \wedge (T_M \, until \, (acc_M \wedge empty))$ in PTL$^{\mathrm{U}}$. As a result of our convention for runs and accepting runs, the automaton M's operation requires one state less than a conventional one to accept a word. For example, it can accept one-letter words without the need for any state transitions. In fact, such an automaton M only recognises words with at least one letter (i.e., in Σ_M^+). This is perfect when we utilise semi-automata and automata to mimic PITL formulas since ITL intervals have at least one state.

The regular expressiveness of PITL with finite time ensures that any PITL formula B can be recognised by some M. The set V_B of propositional variables in B and the set Q_M of M's control states are assumed to be distinct. Formally, we have the next valid formula expressed in QPITL (defined in Section 2):

$$\models \quad finite \quad \supset \quad B \equiv \exists q_1, \ldots, q_{|Q_M|}.\big(init_M \wedge \Box(more \supset T_M) \wedge fin\, acc_M\big).$$

For instance, below is a sample automaton M to recognise finite intervals satisfying the formula $(skip \wedge p)^\frown skip^\frown skip^{\star\frown}(empty \wedge \neg p)$, which is semantically equivalent to the PTL formula $p \wedge \bigcirc \bigcirc \Diamond(empty \wedge \neg p)$:

$$
\begin{aligned}
&V_M = \{p\} \text{ (so } \Sigma_M = \{p, \neg p\}) \quad Q_M = \{q_1, q_2, q_3, q_4\} \quad q_M^I = q_1 \\
&\delta_M(q_1, p) = q_2 \quad \delta_M(q_1, \neg p) = q_4 \quad \delta_M(q_2, p) = \delta_M(q_2, \neg p) = q_3 \\
&\delta_M(q_3, p) = \delta_M(q_3, \neg p) = q_3 \quad \delta_M(q_4, p) = \delta_M(q_4, \neg p) = q_4 \\
&\tau_M(q_1) = \tau_M(q_2) = \tau_M(q_4) = \{\} \quad \tau_M(q_3) = \{\neg p\}
\end{aligned}
\tag{7.1}
$$

Here is an accepting run for the 5-letter word $p \, \neg p \, p \, p \, \neg p$: $q_1 \, q_2 \, q_3 \, q_3 \, q_3$:

$$
\begin{aligned}
&init_M \colon \; q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4 \quad acc_M \colon \; q_3 \wedge \neg p \\
&T_M \colon \quad (\bigcirc q_1) \equiv false \; \wedge \; (\bigcirc q_2) \equiv (q_1 \wedge p) \\
&\qquad\quad \wedge \; (\bigcirc q_3) \equiv (q_2 \vee q_3) \; \wedge \; (\bigcirc q_4) \equiv ((q_1 \wedge \neg p) \vee q_4)
\end{aligned}
$$

Accepting run in PTL: $finite \wedge init_M \wedge \Box(more \supset T_M) \wedge fin\, acc_M$

Below are the values of $q_1, \ldots, q_4$ over an associated 5-state interval in which p has the behaviour $p \, \neg p \, p \, p \, \neg p$:

$$
\begin{aligned}
&(\boldsymbol{q_1}, \neg q_2, \neg q_3, \neg q_4) \quad (\neg q_1, \boldsymbol{q_2}, \neg q_3, \neg q_4) \quad (\neg q_1, \neg q_2, \boldsymbol{q_3}, \neg q_4) \\
&\quad (\neg q_1, \neg q_2, \boldsymbol{q_3}, \neg q_4) \quad (\neg q_1, \neg q_2, \boldsymbol{q_3}, \neg q_4).
\end{aligned}
\tag{7.2}
$$

In each tuple, we show the unique active control state in boldface. For instance, q_2 is true in the second interval state since $q_1 \wedge p$ is true in the first one.

7.3. **ATAs for Semi-Automata and Automata.** The runs of a deterministic semi-automaton or deterministic automaton from the initial control state can alternatively be expressed with an ATA (defined in §5.5). We will consider the case for a semi-automaton S, but the technique is identical for an automaton M. Now PITL with finite time can express all regular languages in Σ_S^+. For each control state q of S, the set of words in Σ_S^+ for which S starts in the initial control state q_S^I and ends in q is regular. The regular expressiveness of PITL with finite time ensures that there exists some corresponding PITL formula $C_{S,q}$ which only has variables in the set V_S and expresses this set of words. In principle, such a formula can be obtained by adapting standard techniques for constructing a regular expression from a conventional finite-state automaton. Now let the ATA D_S denote the conjunction $\bigwedge_{q \in Q_S}(q \leftarrow C_{S,q})$. We express finite runs in PITL using $finite \wedge \boxdot D_S$. Here is such an ATA for the earlier sample automaton in (7.1):

$$q_1 \leftarrow (empty \wedge p) \quad \wedge \quad q_2 \leftarrow (skip \wedge p) \quad \wedge \quad q_3 \leftarrow (skip \wedge p)^\frown skip^\frown skip^\star \quad \wedge \quad q_4 \leftarrow (more \wedge \neg p).$$

Note that the case for q_3 simplifies to $q_3 \leftarrow (p \wedge \bigcirc\bigcirc true)$. The 5-tuple sample run in (7.2) reflects behaviour in prefix subintervals for the previous illustrative word $p \neg p\, p\, p \neg p$. For example, q_2 is true in just the second interval state since the 2-state prefix subinterval is the only prefix subinterval satisfying the formula $skip \wedge p$.

For any deterministic automaton M, let D_M denote some ATA obtained from M in exactly the same way as for a semi-automaton.

7.4. **Formal Equivalence of the Two Representations of Runs.** For finite time, the PITL formula $\boxdot D_S$ expresses all runs of S starting from its initial control state. Hence for finite time this formula is semantically equivalent to the previous formulas for this behaviour (e.g., the PTL formula $init_S \wedge \square(more \supset T_S)$). Consequently, the next valid formula relates the two ways of expressing S's runs:

$$\vDash \quad finite \;\supset\; \Big(\boxdot D_S \;\equiv\; \big(init_S \wedge \square(more \supset T_S)\big)\Big). \tag{7.3}$$

The use of a single example (7.1) for both representations of S's runs can be justified from this. An automaton M's accepting runs can be expressed with $finite \wedge (\boxdot D_M) \wedge fin\, acc_M$. The QPITL formula below is valid for any PITL formula B and automaton M which recognises B:

$$\vDash \quad finite \quad\supset\quad B \;\equiv\; \exists q_1, \ldots, q_{|Q_M|}.\big(\boxdot D_M \;\wedge\; fin\, acc_M\big).$$

The valid PITL^K formula (7.3) just given relates two ways of representing in temporal logic the runs of a finite-state semi-automaton (that is, $\boxdot D_S$ and $init_S \wedge \square(more \supset T_S)$). It includes an explicit assumption about finite time. The next Lemma 7.1 eliminates this requirement and provides a way to re-express $\boxdot D_S$ as an equivalent PTL formula in deductions concerning infinite time. The proof of Lemma 7.1 only involves temporal logic and requires no explicit knowledge about omega automata.

For the convenience of readers studying our deductions here and later on in Section 9, Table 4 lists every PITL theorem and derived rule explicitly mentioned somewhere prior to Appendix A. The appendix itself contains all needed PITL theorems and derived rules and as well as their individual proofs.

T1	$\vdash$	$\boxdot(A \supset A') \supset (A^\frown B) \supset (A'^\frown B)$	
T3	$\vdash$	$\Box(B \equiv B') \supset (A^\frown B) \equiv (A^\frown B')$	
DR4	$\vdash A \Rightarrow \vdash \boxdot A$		
DR12	$\vdash A \equiv B \Rightarrow \vdash \Box A \equiv \Box B$		
DR13	$\vdash \Box A \supset B \Rightarrow \vdash \Box A \supset \Box B$		
T18	$\vdash$	$(A \vee A')^\frown B \equiv (A^\frown B) \vee (A'^\frown B)$	
T25	$\vdash$	$\boxdot(A \supset B) \supset (\boxdot A) \supset (\boxdot B)$	
T28	$\vdash$	$\boxdot(A \wedge B) \equiv \boxdot A \wedge \boxdot B$	
T30	$\vdash$	$\boxdot(A \equiv A') \supset (A^\frown B) \equiv (A'^\frown B)$	
T37	$\vdash$	$\boxdot w \equiv w$	
T42	$\vdash$	$(w \wedge empty)^\frown A \equiv w \wedge A$	
T46	$\vdash$	$\boxdot \boxdot A \equiv \boxdot A$	
T55	$\vdash$	$\boxdot \Box A \equiv \Box \boxdot A$	
T58	$\vdash$	$A^\star \equiv (A^{\star \frown} empty) \vee A^\omega$	
T62	$\vdash$	$\Diamond(more \wedge T) \equiv more \wedge T$	
T63	$\vdash$	$\boxdot(more \supset T) \equiv more \supset T$	
T68	$\vdash$	$\Diamond(skip \wedge T) \equiv more \wedge T$	
T69	$\vdash$	$(skip \wedge T)^\frown A \equiv T \wedge \bigcirc A$	
T70	$\vdash$	$T \; until \; A \equiv A \vee \big(T \wedge \bigcirc(T \; until \; A)\big)$	
T71	$\vdash$	$T \; until \; A \supset \Diamond A$	

Table 4: PITL theorems and derived rules mentioned before Appendix A

Lemma 7.1. *For any deterministic finite-state semi-automaton S, the next $PITL^K$ equivalence involving S's ATA D_S and a PTL formula is a PITL theorem:*

$$\vdash \boxdot D_S \equiv \big(init_S \wedge \Box(more \supset T_S)\big). \tag{7.4}$$

Proof.. The validity of implication (7.3), together with completeness for PITL with finite time ensures that (7.3) is also a deducible theorem:

$$\vdash finite \supset \Big(\boxdot D_S \equiv \big(init_S \wedge \Box(more \supset T_S)\big)\Big).$$

We then deduce from that and Inference Rule $\boxdot$**FGen** the next theorem:

$$\vdash \boxdot\Big(\boxdot D_S \equiv \big(init_S \wedge \Box(more \supset T_S)\big)\Big).$$

From this and some interval-based temporal reasoning about $\boxdot$ (using properties of the underlying modal system K – see Appendix A.2) we can then deduce the equivalence below:

$$\vdash \boxdot D_S \equiv \boxdot init_S \wedge \boxdot \Box(more \supset T_S).$$

Let us now re-express $\boxdot init_S$ as the equivalent state formula $init_S$ (see PITL Theorem **T37**):

$$\vdash \boxdot D_S \equiv init_S \wedge \boxdot \Box(more \supset T_S).$$

We also want to re-express $\boxdot \Box(more \supset T_S)$ as the PTL formula $\Box(more \supset T_S)$. This can be done by first re-expressing $\boxdot \Box$ as $\Box \boxdot$ (see PITL Theorem **T55**) to yield the equivalence below:

$$\vdash \boxdot D_S \equiv init_S \wedge \Box \boxdot(more \supset T_S). \tag{7.5}$$

Let us now consider how to eliminate the operator $\boxdot$ in the subformula $\Box\,\boxdot(\textit{more} \supset T_S)$. The fact that any NL^1 formula T only sees an interval's first two states ensures that the next equivalence is valid and also deducible (see PITL Theorem **T62**):

$$\vdash \quad \Diamond(\textit{more} \wedge T) \quad \equiv \quad \textit{more} \wedge T.$$

A dual form (see PITL Theorem **T63**) is readily deduced for use with T_S:

$$\vdash \quad \boxdot(\textit{more} \supset T_S) \quad \equiv \quad \textit{more} \supset T_S.$$

We employ this with Derived Rule **DR12** to obtain an equivalence for eliminating the $\boxdot$ operator in $\Box\,\boxdot(\textit{more} \supset T_S)$:

$$\vdash \quad \Box\,\boxdot(\textit{more} \supset T_S) \quad \equiv \quad \Box(\textit{more} \supset T_S). \tag{7.6}$$

Equivalence (7.4)'s theoremhood, which is our immediate goal, then readily follows by simple propositional reasoning from the deduced equivalences (7.5) and (7.6). □

8. Compound Semi-Automata for Suffix Recognition

Let a **compound semi-automaton** R be a vector of semi-automata $S_1, \ldots, S_n$ for some $n \geq 1$ with disjoint sets of control states. We take V_R to be the set of propositional variables in the semi-automata $S_1, \ldots, S_n$ which are not also control states. The purpose of R is to perform what we call **suffix recognition**. This is a way to determine which of an finite interval's suffix subintervals satisfy some given PITL formula B. Suffix recognition is a stepping stone enabling us to subsequently perform the *infix recognition* already briefly mentioned in §5.6. Later on in Section 9 this feature of R ensures that for a given PITL^K formula K with m right-chops (previously defined in §5.4), we can utilise m such compound semi-automata to obtain an ATA for infix recognition to replace the left sides of K's right-chops with PTL^U chain formulas (also introduced in §5.4). The n individual semi-automata $S_1, \ldots, S_n$ in R are meant to operate lockstep in parallel and so simultaneously make state transitions. For each $i : 1 \leq i < n$, we require for the set $V_{S_{i+1}}$, which contains propositional variables examined by S_{i+1}, that $V_{S_{i+1}} \subseteq V_{S_i} \cup Q_{S_i}$. Hence the control states of S_i are allowed occur within the letters for S_{i+1} and any semi-automata of higher index but not vice versa. This enables each semi-automaton to optionally observe control states of all semi-automata with lower index when it makes transitions. In our particular construction of R, the set V_R simply equals the set V_B of propositional variables in the PITL formula B and also equals the lowest-indexed semi-automata S_1's set V_{S_1} of propositional variables used to form the atoms Σ_{S_1}. Let R's ATA D_R be a conjunction of the ATAs for the semi-automata $S_1, \ldots, S_n$. It is not hard to check that D_R obeys the ATA requirement limiting where auxiliary variables can occur (as specified in the definition of ATAs in §5.5) and is therefore well-formed.

We perform suffix recognition by exploiting standard techniques originally developed by McNaughton [McN66] to construct deterministic omega automata. Choueka [Cho74] later applied McNaughton's insights to some constructions for automata on finite words. Our discussion here likewise concerns finite-time behaviour and avoids omega automata. Furthermore, this section deals with semantic issues but not deductions.

wil

8.1. **Overview of Construction of Compound Semi-Automaton.** The compound semi-automaton R to suffix recognise B is built from several modified copies of a deterministic automaton running lockstep in parallel. We also define an associated chain formula G_R. Here is a summary:

- We initially construct R and G_R to just check whether B is true in any given finite suffix subinterval of the overall *finite* interval in which R is run. Consequently, G_R can be used to mimic B.
- We first construct a deterministic finite-state automaton M (discussed in §7.2) to recognise the regular language associated with B in finite time. Let n be the number of control states, that is, $n = |Q_M|$.
- We do not use M directly but instead construct $n+1$ semi-automata S_1, ..., S_{n+1} based on M. The compound semi-automaton R is a vector of them.
- Our construction ensures that always at least one semi-automaton is in (its copy of) M's initial control state and so available to start testing for B in the suffix subinterval commencing at the current state.
- A suffix subinterval satisfies B iff there is exists a simulation of an accepting run of M which starts in the subinterval's first state, ends in its last one (the same as the overall interval's final state) and is formed by combining up to $n+1$ pieces of runs of the semi-automata S_1, ..., S_{n+1}. The successive partial runs are performed on semi-automata of decreasing index.

8.2. **Construction of the Individual Semi-Automata.** Let us now consider the details of the $n+1$ semi-automata variants S_1, ..., S_{n+1} of M. A semi-automaton S_k has its own disjoint set $Q_{S_k} = \{q_1^{S_k}, \dots, q_n^{S_k}\}$ of copies of the n control states in M and is initialised exactly as M would be and hence starts in (its copy of) M's initial control state. We let S_k examine the control states of semi-automata with lower index (i.e., $S_1, \dots, S_{k-1}$) when it makes its transitions in lockstep with them. Hence, the set of propositional variables V_{S_k} is the union of V_M and $\bigcup_{1 \le j < k} Q_{S_j}$ and all propositional variables in an atom α in Σ_{S_k} are therefore either in V_M or are control states in the semi-automata $S_1, \dots, S_{k-1}$.

We now define the **transition function δ_{S_k}** of each semi-automaton S_k in R for use when all of the semi-automata operate in lockstep. The transition function $\delta_{S_k} \colon Q_{S_k} \times \Sigma_{S_k} \to Q_{S_k}$ is deterministic like M's, but more complicated. For each pair $\langle q_i^{S_k}, \alpha \rangle$ in $Q_{S_k} \times \Sigma_{S_k}$, there are two distinct possible cases based on the values of $q_i^{S_k}$ and α. We now define these cases and the associated transitions:

- **The pair $\langle q_i^{S_k}, \alpha \rangle$ is active:** This occurs when for every $j < k$, the pair's atom α assigns the control variable $q_i^{S_j}$ to be false. It corresponds to a situation where S_k is the semi-automaton of lowest index in R currently in (its own copy $q_i^{S_k}$ of) M's control state q_i^M and itself also called **active**.

  Let $\beta \in \Sigma_M$ be the atom in Σ_M obtained from α by only using the propositional variables in V_M and thereby ignoring the control variables in α. Now we have that $\delta_M(q_i^M, \beta) = q_j^M$ for some $q_j^M \in Q_M$. Define the transition $\delta_{S_k}(q_i^{S_k}, \alpha)$ to be the corresponding $q_j^{S_k} \in Q_{S_k}$.

- **The pair $\langle q_i^{S_k}, \alpha \rangle$ is inactive:** If the first case does not apply, then S_k shares (its copy of) M's control state q_i^M with some semi-automaton of lesser index as seen by S_k via

the atom α. We define the transition $\delta_{S_k}(q_i^{S_k}, \alpha)$ to equal the initial control state of S_k. Hence S_k makes a transition from its current control state to (its copy of) M's initial control state so in effect reinitialises itself. Our construction of R ensures that some other semi-automaton with lower index which is both active and presently in (its own copy of) the same control state q_i^M of M now indeed takes over from S_k. We also say that S_k is **inactive** and that the two semi-automata **merge**.

Figure 1 gives an example of an deterministic automaton M with four states and a run of an associated compound semi-automaton with five semi-automata $S_1, \ldots, S_5$.

Recall that our representation of M's n control states using n propositional variables $q_1^M, \ldots, q_n^M$ has exactly one of the variables being true at any time. Hence we represent the n control states for a semi-automata S_k using n propositional variables $q_1^k, \ldots, q_n^k$. Therefore the subset of atoms in Σ_{S_k} extracted from R's composite runs always have exactly one variable q_i^j true for each semi-automaton S_j with $j < k$. This property of the runs follows by induction on k. In contrast, the full set of atoms for Σ_{S_k} includes for each index j with $j < k$ some **pathological atoms** in which none or more than one of the q_i^j are true. Nevertheless, actual runs of S_k in R never encounter such atoms so we need not concern ourselves with the precise way δ_{S_k} is defined to handle them in transitions.

8.3. **Formalisation of Suffix Recognition in PITL.** The following lemma formalises the finite-time behaviour of the compound semi-automaton R in PITL and uses an associated chain formula G_R in PTL$^{\text{U}}$ which we construct in the proof:

Lemma 8.1. *For any PITL formula B, there exists a compound semi-automaton R with $V_R = V_B$ and associated ATA D_R and chain formula G_R such that R's control variables are not in B and the next implication is valid:*

$$\vDash \quad \text{finite} \wedge \boxdot D_R \quad \supset \quad \Box(B \equiv G_R). \tag{8.1}$$

This lemma provides a way to replace right-instances of a PITL formula B by a chain formula G_R in formulas restricted to finite time. However, it serves as basis for later replacing *lefthand* sides of chops with chain formulas. The lemma is entirely semantic and so does not depend on any particular axiom system or deductions. We will later readily deduce the lemma's implication (8.1) by invoking the completeness for PITL with finite time to obtain immediate theoremhood of the implication and some valid variants of it. Hence, from the standpoint of axiom systems and deductions, there is no need to know Lemma 8.1's proof or even any further details of R, D_R and G_R.

Proof of Lemma 8.1.. The construction for R ensures that the set union $Q_{S_1} \cup \cdots \cup Q_{S_{n+1}}$ of control variables of the semi-automata S_1, $\ldots$, S_{n+1} contains no elements of the set V_B of propositional variables occurring in B.

We will obtain the chain formula G_R by mimicking an accepting run of M. This involves combining together pieces of runs from the some of the semi-automata S_1, $\ldots$, S_{n+1}. It needs at most n merges since when two semi-automata merge, only the one of lesser index continues testing. The chain formula G_R, when suitably combined with the compound semi-automaton R's ATA, will capture the needed behaviour which we previously formalised in the implication (8.1).

We first define state formulas to test for active and merging semi-automata and also introduce a modified acceptance test:

Sample formula B: $(skip \wedge p) \frown skip \frown skip^\star \frown (empty \wedge \neg p)$

Sample automaton M for B (already presented in (7.1)):

$V_M = \{p\}$ (so $\Sigma_M = \{p, \neg p\}$) $\quad Q_M = \{q_1, q_2, q_3, q_4\} \quad q_M^I = q_1$

$\delta_M(q_1, p) = q_2 \quad \delta_M(q_1, \neg p) = q_4 \quad \delta_M(q_2, p) = \delta_M(q_2, \neg p) = q_3$

$\delta_M(q_3, p) = \delta_M(q_3, \neg p) = q_3 \quad \delta_M(q_4, p) = \delta_M(q_4, \neg p) = q_4$

$\tau_M(q_1) = \tau_M(q_2) = \tau_M(q_4) = \{\} \quad \tau_M(q_3) = \{\neg p\}$

$init_M$: $\quad q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4 \quad acc_M$: $\quad q_3 \wedge \neg p$

T_M: $\quad (\bigcirc q_1) \equiv false \ \wedge \ (\bigcirc q_2) \equiv (q_1 \wedge p)$
$\quad\quad\quad \wedge \ (\bigcirc q_3) \equiv (q_2 \vee q_3) \ \wedge \ (\bigcirc q_4) \equiv ((q_1 \wedge \neg p) \vee q_4)$

Control state behaviour of each S_k in sample 8-state interval σ:

State in σ	p's value	S_1	S_2	S_3	S_4	S_5
σ_0	$\neg p$	**1**	1	1	1	1
σ_1	p	**4**	**<u>1</u>**	1	1	1
σ_2	$\neg p$	**4**	**<u>2</u>**	1	1	1
σ_3	p	**4**	**<u>3</u>**	4	**<u>1</u>**	1
σ_4	p	**4**	**<u>3</u>**	**<u>1</u>**	**<u>2</u>**	1
σ_5	$\neg p$	**4**	**<u>3</u>**	**<u>2</u>**	$S_2{\leftarrow}3$	**1**
σ_6	p	**4**	**<u>3</u>**	$S_2{\leftarrow}3$	**<u>1</u>**	**4**
σ_7	$\neg p$	**4**	**<u>3</u>**	**<u>1</u>**	**<u>2</u>**	**1**

Value of acc_k' for each S_k at end in state σ_7:

$$false \quad true \quad false \quad false \quad false$$

Some explanations about the sample 8-state interval $\sigma_0 \dots \sigma_7$:

Only control states' indices are shown (e.g., 1 for q_1).

Active semi-automata are shown in **boldface**.

All control states used in any accepting runs of M are <u>underlined</u>.

"$S_2{\leftarrow}$" shows merge into semi-automaton S_2 in accepting run for M.

Compound accepting runs of M to recognise B:

Suffix subinterval $\sigma_1 \dots \sigma_7$ (S_2: $\sigma_1 \sigma_2 \sigma_3 \sigma_4 \sigma_5 \sigma_6 \sigma_7$): $\underbrace{q_1, q_2, q_3, q_3, q_3, q_3, q_3}_{S_2}$

Suffix subinterval $\sigma_3 \dots \sigma_7$ (S_4: $\sigma_3 \sigma_4$, S_2: $\sigma_5 \sigma_6 \sigma_7$): $\underbrace{q_1, q_2,}_{S_4} \underbrace{q_3, q_3, q_3}_{S_2}$

Suffix subinterval $\sigma_4 \dots \sigma_7$ (S_3: $\sigma_4 \sigma_5$, S_2: $\sigma_6 \sigma_7$): $\underbrace{q_1, q_2,}_{S_3} \underbrace{q_3, q_3}_{S_2}$
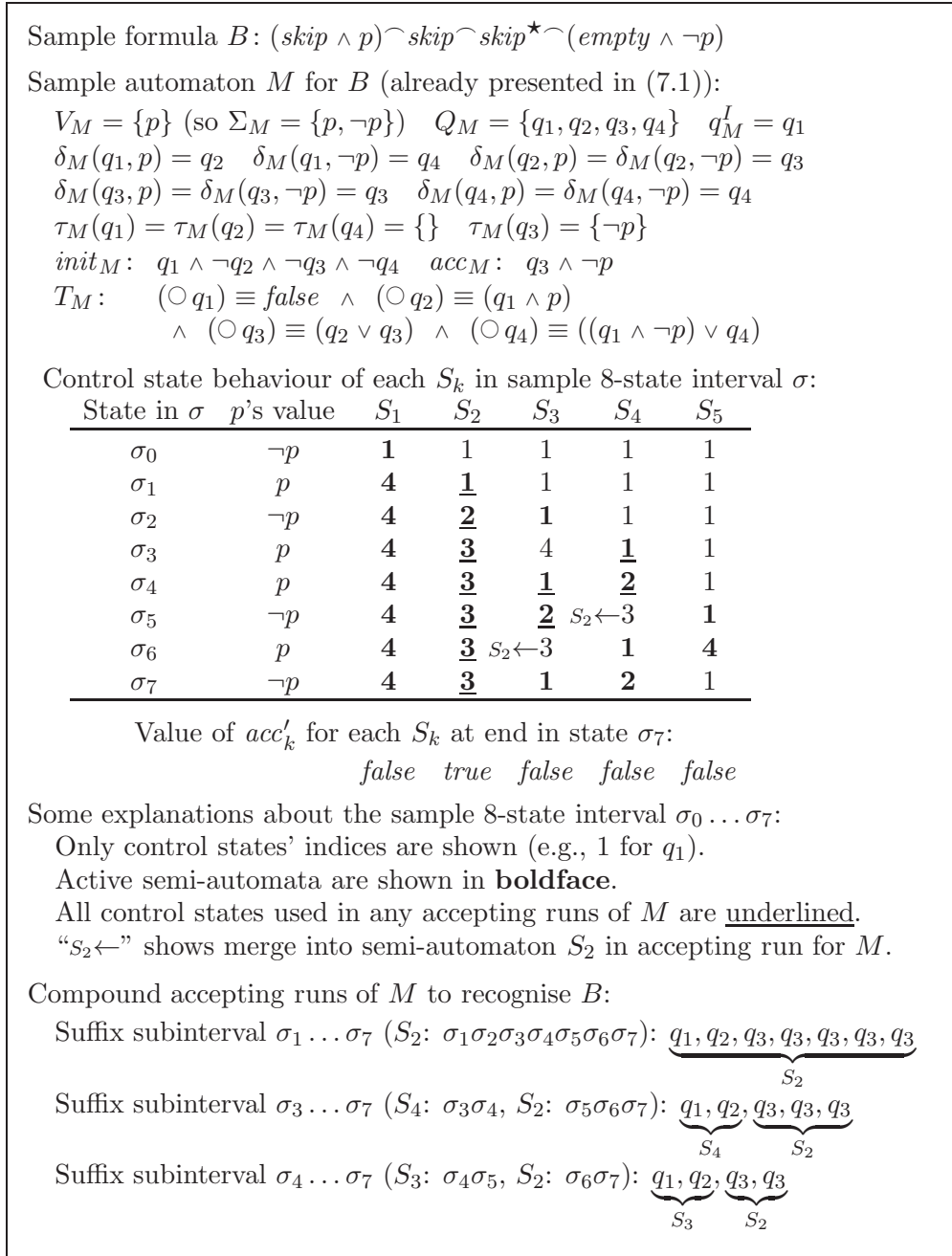
Figure 1: Sample behaviour of compound semi-automaton in 8-state interval

- **$active_k$:** True iff semi-automaton S_k is *active*.

$$active_k \quad \stackrel{\text{def}}{\equiv} \quad \bigvee_{1 \leq i \leq n+1} \left(q_i^{S_k} \wedge \bigwedge_{1 \leq j < k} \neg q_i^{S_j} \right).$$

- **$merge_{j,k}$:** True iff the active semi-automaton S_j and inactive semi-automaton S_k merge.

$$merge_{j,k} \quad \stackrel{\text{def}}{\equiv} \quad \bigvee_{1 \le i \le n+1} (q_i^{S_j} \wedge q_i^{S_k} \wedge active_j \wedge \neg active_k).$$

  It follows from the definition of an active semi-automaton that $j < k$.

- **acc_k':** Let us also define a propositional test acc_k' based on the state formula acc_M for checking M's conditional acceptance test τ_M. We use a substitution instance of acc_M to adapt it to S_k and its own copies of M's control states.

$$acc_k' \quad \stackrel{\text{def}}{\equiv} \quad (acc_M)_{q_1^M,...,q_n^M}^{q_1^{S_k},...,q_n^{S_k}}.$$

  Note that a semi-automaton S has no conditional acceptance test τ_S and indeed the role of acc_k' here somewhat differs from that of acc_M.

As usual, for an individual semi-automaton S_k in the compound semi-automaton R, the state formula $init_{S_k}$ tests for the initial control state of S_k and the NL^1 formula T_{S_k} expresses the transition function δ_{S_k} of S_k in temporal logic.

Let us now inductively define for each pair $j, k : 1 \le j \le k \le n+1$ a chain formula $G_{k,j}'$ to be true iff a run segment starts with currently active semi-automaton S_k in some unspecified control state, involves exactly j active automata (i.e., $j-1$ mergers) and ends with acceptance of the word seen.

$$\begin{aligned} G_{k,1}': & \quad (active_k \wedge T_{S_k}) \, until \, (acc_k' \wedge empty) \\ G_{k,j+1}': & \quad (active_k \wedge T_{S_k}) \, until \, \bigvee_{1 \le i < k} (merge_{i,k} \wedge G_{i,j}'). \end{aligned}$$

For example, the chain formula $init_{S_1} \wedge active_1 \wedge G_{1,1}'$ corresponds to an accepting run of M in which the semi-automaton S_1 recognises B on its own. The conjunction $init_{S_2} \wedge active_2 \wedge G_{2,2}'$ corresponds to an accepting run of M involving first semi-automaton S_2 and then semi-automaton S_1. The semi-automaton S_2 starts recognising B and eventually merges with semi-automaton S_1 which completes the accepting run.

Now let us construct from the chain formulas $G_{k,j}'$ the chain formula G_R specifying an accepting run involving some of the $n+1$ semi-automata to recognise the PITL formula B. Like in the examples, we start in some active copy of M's initial control state:

$$G_R: \quad \bigvee_{1 \le k \le n+1} (init_{S_k} \wedge active_k \wedge \bigvee_{1 \le j \le k} G_{k,j}').$$

The construction of the compound semi-automaton R together with D_R and G_R ensures the desired validity of implication (8.1). $\qquad\square$

To assist readers, we list in Table 5 a variety of variables and where they are introduced.

9. Reduction of PITL to PTL with Until

Most of the remaining part of the PITL completeness proof concerns using compound semi-automata to show right-completeness for PITL^K by reduction to PTL^U. Recall from §5.4 that any chop construct in a formula A is a right-chop iff it does not occur in another chop's left operand or in a chop-star.

The PITL theorems mentioned here in proofs are found in Table 4 in §7.4 and also Appendix A.

Variable names	Category	Where defined
A, A', B, C	Arbitrary PITL formulas	§2
α, β	Atoms (letters)	§7
acc_M	State formula for automaton M's acceptance	§7.2
D, D'	Auxiliary temporal assignments (ATA)	§5.5
D_S, D_M, D_R	ATA for use in expressing runs of S, M and R	§7.3, §8
δ_S, δ_M	Deterministic transition function for semi-automaton S and automaton M	§7.1, §7.2
G, G'	Chain formulas	§5.4
$init_S, init_M$	State formula to force the initial control state of semi-automaton S and automaton M	§7.1, §7.2
K, K'	PITL$^{\mathrm{K}}$ formulas	§5.3
M	Deterministic finite-state automaton	§7.2
p, p', q, r	Propositional variables	§2
Q_S, Q_M	Sets of control states of semi-automaton S and automaton M	§7.1, §7.2
R	Compound finite-state semi-automaton	§8
S	Deterministic finite-state semi-automaton	§7.1
Σ_V	Atoms (letters) formed from variables in set V	§7
Σ_S, Σ_M	Atoms tested by semi-automaton S and automaton M	§7.1, §7.2
T, T'	NL^1 formulas	§5.1
T_S, T_M	NL^1 formula for transitions of semi-automaton S and automaton M	§7.1, §7.2
τ_M	Conditional acceptance test for automaton M	§7.2
V	Finite set of propositional variables	§2
V_A, V_S, V_M, V_R	Finite set of propositional variables in PITL formula A and in atoms of semi-automaton S, automaton M and compound semi-automaton R	§2, §7.1, §7.2, §8
w, w'	State formulas	§2
X, X'	PTL formulas	§2
Y, Y'	PTL$^{\mathrm{U}}$ formulas	§5.2

Table 5: Naming conventions for different variables

9.1. **Application of Suffix Recognition, Right-Chops and Chain Formulas.** The next Lemma 9.1, which employs the compound semi-automaton R, generalises suffix recognition to **infix recognition** for checking which of a (possibly infinite-time) interval's finite-time infix subintervals satisfy some given PITL formula by instead using a chain formula.

Lemma 9.1. *For any PITL formula B, there exists a compound semi-automaton R with $V_R = V_B$, associated ATA D_R and chain formula G_R such that R's control variables are not in B and the next formula is a PITL theorem:*

$$\vdash \quad \boxdot D_R \quad \supset \quad \Box\,\boxdot(B \equiv G_R). \tag{9.1}$$

Proof. Lemma 8.1 ensures the validity of the implication below for some compound semi-automaton R, associated ATA D_R and chain formula G_R:

$$\vDash \quad \mathit{finite} \wedge \boxdot D_R \quad \supset \quad \Box(B \equiv G_R).$$

This and completeness for PITL with finite time (Theorem 2.2) ensures the next implication's theoremhood:

$$\vdash \quad \textit{finite} \quad \supset \quad \big(\boxdot D_R \; \supset \; \Box(B \equiv G_R)\big).$$

This and Inference Rule $\boxdot$**FGen** yield the next formula:

$$\vdash \quad \boxdot\big(\boxdot D_R \; \supset \; \Box(B \equiv G_R)\big).$$

Simple reasoning about $\boxdot$ (see PITL Theorem **T25**) results in the following:

$$\vdash \quad \boxdot \boxdot D_R \quad \supset \quad \boxdot \Box(B \equiv G_R).$$

We re-express $\boxdot \boxdot D_R$ as $\boxdot D_R$ and commute $\boxdot \Box$ (see PITL Theorems **T46** and **T55**) to obtain our goal (9.1). $\qquad\square$

The lemma below later plays a key role in reducing right-chops in a PITL$^{\mathrm{K}}$ formula to PTL$^{\mathrm{U}}$ formulas by first replacing their left sides with chain formulas in PTL$^{\mathrm{U}}$:

Lemma 9.2. *For any PITL formulas B and C, there exists a compound semi-automaton R with $V_R = V_B$, associated ATA D_R and chain formula G_R such that R's control variables are not in B or C and the next formula is deducible as a right-theorem:*

$$\vdash_{\mathrm{rt}} \quad \boxdot D_R \quad \supset \quad \Box\big((B^\frown C) \equiv (G_R^\frown C)\big). \tag{9.2}$$

Proof.. Lemma 9.1 yields R, D_R, G_R and the next implication for infix recognition of B:

$$\vdash \quad \boxdot D_R \quad \supset \quad \Box\boxdot(B \equiv G_R). \tag{9.3}$$

Note that this has no right variables. We also employ the next implication which is an instance of PITL Theorem **T30** and concerns interval-based reasoning about the left of chop:

$$\vdash_{\mathrm{rt}} \quad \boxdot(B \equiv G_R) \quad \supset \quad (B^\frown C) \equiv (G_R^\frown C). \tag{9.4}$$

Inference Rule $\Box$**Gen** then obtains from implication (9.4) the formula below:

$$\vdash_{\mathrm{rt}} \quad \Box\big(\boxdot(B \equiv G_R) \; \supset \; (B^\frown C) \equiv (G_R^\frown C)\big).$$

This with PTL-based reasoning involving the valid PTL formula $\Box(p \supset q) \supset \big((\Box p) \supset (\Box q)\big)$ with Axiom **VPTL**, where p is replaced by $\boxdot(B \equiv G_R)$ and q by $(B^\frown C) \equiv (G_R^\frown C)$, together with modus ponens results in the following:

$$\vdash_{\mathrm{rt}} \quad \Box\boxdot(B \equiv G_R) \quad \supset \quad \Box\big((B^\frown C) \equiv (G_R^\frown C)\big). \tag{9.5}$$

Implications (9.3) and (9.5) and simple propositional reasoning yield our goal (9.2). $\qquad\square$

Lemma 9.3. *Any PITL$^{\mathrm{K}}$ formula K in which the left sides of all right chops are chain formulas is deducibly equivalent to some PTL$^{\mathrm{U}}$ formula Y, that is, $\vdash_{\mathrm{rt}} K \equiv Y$.*

Proof.. Starting with K's right-chops not nested in other right-chops, we inductively replace them by equivalent PTL$^{\mathrm{U}}$ formulas. More precisely, if n is the number of K's right chops, then we use n applications of Lemma 5.2 and the Right Replacement Rule (Lemma 4.2) to show that K is deducibly equivalent to some PTL$^{\mathrm{U}}$ formula Y (i.e., $\vdash_{\mathrm{rt}} K \equiv Y$). $\qquad\square$

For example, suppose K is $(G_1 \frown skip) \vee \big(G_2 \frown (G_3 \frown w)\big)$ and hence has 3 right-chops. We could start by first re-expressing either $G_1 \frown skip$ or $G_3 \frown w$ by an equivalent PTL^U formula. For instance, if G_2 is the chain formula $p \, until \, empty$ and G_3 is the chain formula $q \, until \, empty$, then $G_3 \frown w$ will be replaced by the equivalent PTL^U formula $q \, until \, w$. After this, $G_2 \frown (G_3 \frown w)$ will first reduce to $G_2 \frown (q \, until \, w)$ and finally to the PTL^U formula $p \, until \, (q \, until \, w)$.

9.2. Proof of the Main Completeness Theorem.
We now establish right-completeness for PITL^K and then use this to obtain right-completeness for PITL.

Lemma 9.4. *Any valid PITL^K formula can be deduced as a right-theorem.*

Proof.. We show that a right-consistent PITL^K formula K is satisfiable. Our proof transforms K to a PTL^U formula. Let m equal the number of K's right-chops. We employ m compound semi-automata to obtain ATAs for systematically replacing the left operands of K's right-chops by PTL^U chain formulas. Note that if $m = 0$, then K has no chops but perhaps $skip$ so K itself is in PTL. We will construct a sequence of $m + 1$ PITL^K formulas $K_1', \ldots, K_{m+1}'$. In the final one K_{m+1}', left operands of all right-chops are chain formulas so K_{m+1}' is deducibly equivalent to some PTL^U formula by Lemma 9.3. For example, suppose K has the form $(B_1 \frown w) \supset \big(B_2 \frown (B_3 \frown skip)\big)$. Then K has 3 right-chops so m equals 3 and K_4' has the form $(G_1 \frown w) \supset \big(G_2 \frown (G_3 \frown skip)\big)$, where G_1, G_2 and G_3 in K_4''s 3 right-chops' left sides are all chain formulas.

Let K_1' be K. For each $i : 1 \leq i \leq m$, we choose a right-chop in K_i'. This has the form $B_i \frown K_i''$ for some PITL formula B_i and PITL^K formula K_i''. Lemma 9.2 yields a compound semi-automaton R_i', ATA $D_{R_i'}$ and a chain formula $G_{R_i'}$ for which the next right-theorem is deducible:

$$\vdash_{\mathrm{rt}} \quad \boxdot D_{R_i'} \quad \supset \quad \Box\big((B_i \frown K_i'') \equiv (G_{R_i'} \frown K_i'')\big). \tag{9.6}$$

We employ Lemma 4.1 concerning replacement of right-instances to relate K_i' and K_{i+1}' by replacing the selected $B_i \frown K_i''$ by $G_{R_i'} \frown K_i''$:

$$\vdash_{\mathrm{rt}} \quad \Box\big((B_i \frown K_i'') \equiv (G_{R_i'} \frown K_i'')\big) \quad \supset \quad K_i' \equiv K_{i+1}'.$$

This and implication (9.6) together ensure the right-theorem $\vdash_{\mathrm{rt}} \boxdot D_{R_i'} \supset (K_i' \equiv K_{i+1}')$. Without loss of generality, assume the control variables in the compound semi-automata $R_1', \ldots, R_m'$ are distinct. We deduce from the m implications $\vdash_{\mathrm{rt}} \boxdot D_{R_i'} \supset (K_i' \equiv K_{i+1}')$ just mentioned the next right-theorem:

$$\vdash_{\mathrm{rt}} \quad \bigwedge_{1 \leq i \leq m}(\boxdot D_{R_i'}) \quad \supset \quad K \equiv K_{m+1}'. \tag{9.7}$$

The left operand of each right-chop in K_{m+1}' is a chain formula. Hence by Lemma 9.3, we can deduce the equivalence of K_{m+1}' and some PTL^U formula Y to obtain the PITL right-theorem $\vdash_{\mathrm{rt}} K_{m+1}' \equiv Y$. By this and implication (9.7), the next implication is a right-theorem:

$$\vdash_{\mathrm{rt}} \quad \bigwedge_{1 \leq i \leq m}(\boxdot D_{R_i'}) \quad \supset \quad K \equiv Y. \tag{9.8}$$

Right-variables in the original formula K do not occur in any $D_{R_i'}$ since the construction of each $D_{R_i'}$ only involves the left sides of K's right-chops. The right-variables in K are still right-variables in Y and implication (9.8). Now K's right-consistency and m applications of Lemma 5.4 ensure the right-consistency of $K \wedge \bigwedge_{1 \leq i \leq m}(\boxdot D_{R_i'})$. This is re-expressible as $K \wedge \boxdot D'$, where the ATA D' is the conjunction of the ATAs $D_{R_1'}, \ldots, D_{R_m'}$ (we use

PITL Theorem **T28**). Hence the formula $K \wedge \boxdot D'$ is right-consistent. We deduce the equivalence of $\boxdot D'$ and some PTL formula X as $\vdash X \equiv \boxdot D'$ by invoking Lemma 7.1 on the individual basic semi-automata in each R'_i to re-express each one's runs in PTL and then forming the conjunction of results. Now D' and X have the same variables. Hence the equivalence $X \equiv \boxdot D'$ has no right-variables because of $\boxdot D'$ and is a right-theorem (i.e., $\vdash_{\mathrm{rt}} X \equiv \boxdot D'$). This with the equivalence $\vdash_{\mathrm{rt}} \boxdot D' \equiv \bigwedge_{1 \leq i \leq m}(\boxdot D_{R'_i})$ and implication (9.8) then yield the equivalence of formulas $K \wedge \boxdot D'$ and $Y \wedge X$ as a right-theorem. Therefore the PTL$^{\mathrm{U}}$ formula $Y \wedge X$, like $K \wedge \boxdot D'$, is right-consistent and by right-completeness for PTL$^{\mathrm{U}}$ (discussed in §5.2) is satisfiable as is K. $\qquad\square$

We now prove our main result Theorem 3.2 about right-completeness for PITL:

Proof of Theorem 3.2.. Let A be a right-consistent PITL formula. Lemma 9.4 ensures right-completeness for PITL$^{\mathrm{K}}$. Hence by this and Lemma 6.2, there exists some PITL$^{\mathrm{K}}$ formula K having the same variables and right-variables as A and with the deducible equivalence $\vdash_{\mathrm{rt}} A \equiv K$. Now K like A is right-consistent and so satisfiable by right-completeness for PITL$^{\mathrm{K}}$ (Lemma 9.4). Hence A is satisfiable. $\qquad\square$

As we already remarked in Section 3, the completeness proof can be regarded as two parallel proofs. The simpler one uses the extra inference rule (3.2) mentioned there to avoid right-theorems and right-completeness. The more sophisticated proof uses right-theoremhood instead of the inference rule and ensures that any valid PITL formula is not just a theorem but a right-theorem.

This concludes the PITL completeness proof.

10. Some Observations about the Completeness Proof

We now consider various issues concerning the new PITL axiom system and techniques employed in the completeness proof. Most of the points address questions previously raised by others.

10.1. **Alternative Axioms for PTL.** Axiom **VPTL** in Table 2 can optionally be replaced by four lower level axioms. Readers may wish to skip over the details now given. One of the lower level axioms is **Taut** in Table 3 permitting PITL formulas which are substitution instances of conventional (nonmodal) tautologies. For example, from the valid propositional formula $p \supset (p \vee q)$ follows $\vdash A \supset (A \vee B)$, for any PITL formulas A and B. The other three axioms involve PTL. These are Axioms **F10** and **F11** found in Table 3 and also $\vdash skip \supset finite$. The three Axioms **Taut**, **F10** and **F11** together with the remaining PITL axioms and inference rules in Table 2 then suffice to derive a slight variant proposed by us in [Mos04] of the complete PTL axiom system D^0X for $\bigcirc$ and $\diamond$ (and $\square$) of Gabbay et al. [GPSS80], itself based on an earlier one DX of Pnueli [Pnu77]. We denote our D^0X variant here as D^0X'. It permits both finite and infinite time, whereas D^0X assumes infinite time. We previously did an explicit deduction of D^0X' in our completeness proof for PITL with just finite time as described in [Mos04]. However, for infinite time we need the additional axiom $\vdash skip \supset finite$ because Axiom **P6** (unlike Axiom **F6** in Table 3) does not suffice on its own to deduce $\vdash skip \equiv \bigcirc empty$ to re-express $skip$ using $\bigcirc$. Without $\vdash skip \supset finite$, we can only deduce the PITL theorem $\vdash finite \supset (skip \equiv \bigcirc empty)$ from Axiom **P6** together with the definition of $\bigcirc$ in terms of $skip$ and chop. In addition, from

$D^0 X'$ (once deduced), we can obtain $\vdash (\bigcirc \, empty) \supset finite$. These two implications combined with $\vdash skip \supset finite$ and simple propositional reasoning (involving Axiom **Taut** and modus ponens) yield our goal $\vdash skip \equiv \bigcirc \, empty$.

10.2. **Feasibility of Reduction from PITL to PTL.**

Some people have expressed serious doubts about our proof's technical feasibility owing to the significant gap in expressiveness between PITL and PTL. We therefore believe it is worthwhile to emphasis that in spite of this gap, any PITL formula can be represented by some PTL formula containing auxiliary variables. This is because conventional semantic reasoning about omega-regular languages and omega automata ensures that for any PITL formula A, there exist conventional nondeterministic omega automata (such as Büchi automata) which recognise A. For example, we present in [Mos00] a decidable version of quantified ITL which includes QPITL (defined earlier in Section 2) as a subset and then show how to encode formulas in Büchi automata. Various deterministic omega automata (e.g., with Muller, Rabin and Streett acceptance conditions) are also suitable for this. Such an automaton's accepting runs can be trivially encoded by some PTL formula X with auxiliary variables p_1, …, p_n representing the automaton's control state. Hence the PITL formula A and the QPTL formula $\exists p_1 \ldots p_n . X$ are semantically equivalent, where $\exists$ is defined earlier in Section 2. Furthermore, the (quantifier-free) PITL implication $X \supset A$ is valid and consequently any model of X can also serve as one for A. Indeed the technique of re-expressing formulas in omega-regular logics by means of nondeterministic and deterministic omega automata expressed in versions of PTL (subsequently enclosed in a simple sequence of existential quantifiers) is central to the completeness proofs for QPTL variants by Kesten and Pnueli [KP02] and French and Reynolds [FR03]. A related approach can be used to reduce decidability of PTL with the (full) *until* operator to PTL without *until*. This works in spite of the fact that PTL with *until* is strictly more expressive as proved by Kamp [Kam68] (see also Kröger and Merz [KM08]). We replace each *until* in a formula with an auxiliary variable which mimics its behaviour along the lines of the two axioms for *until* previously mentioned in §5.2. For example, when testing the satisfiability of the formula $p \wedge \bigcirc(p \, until \, q) \wedge \neg(p \, until \, q)$, we transform it into the formula below with an extra auxiliary variable r:

$$p \; \wedge \; \bigcirc r \; \wedge \; \neg r \; \wedge \; \Box\big(r \; \equiv \; q \vee (p \wedge \bigcirc r)\big) \; \wedge \; \Box(r \supset \Diamond q).$$

10.3. **Benefits of Restricted Chop-Stars in Chain Formulas.**

Lemma 9.4 states that any valid PITL$^{\mathrm{K}}$ formula can be deduced as a right-theorem. Within the proof of this lemma, all chop-star formulas found in the PITL$^{\mathrm{K}}$ formula K'_{m+1} only occur in chain formulas. Such chop-star formulas therefore have the very restricted form $(skip \wedge T)^{\star}$ for expressing the PITL-based version of *until* defined earlier in §5.2 for PTL$^{\mathrm{U}}$. The simplicity of these chop-star constructs greatly helps us to reduce K'_{m+1} to the semantically equivalent PTL$^{\mathrm{U}}$ formula Y and show that their equivalence is a deducible theorem. Incidentally, in [Mos07] we prove that any PITL formula $(skip \wedge T)^{\star}$ can be expressed in PTL as $\Box(more \supset T)$ and make extensive use of this equivalence. In contrast, arbitrary chop-star formulas cannot necessarily be re-expressed as semantically equivalent PTL formulas.

10.4. **Thomas' Theorem and the Size of Deductions.** Section 6 uses Thomas' theorem to re-express a PITL formula A as a semantically equivalent PITL^K formula K. The two known proofs of Thomas' theorem by Thomas himself [Tho79] and Choueka and Peleg [CP83] unfortunately do not ensure that K is in some sense natural and succinct or even obtainable in a computationally feasible way. Therefore our completeness proof does not guarantee simple deductions. The main problem concerns the difficulties in nontrivial transformations on the underlying omega automata representing PITL formulas. Other established completeness proofs for comparable omega-regular logics with nonelementary complexity such as QPTL [KP95, KP02, FR03] currently share a similar fate. However, our proof bypasses an explicit embedding of the intricate process of complementing nondeterministic omega automata.

10.5. **Justification for Using ATAs in the Completeness Proof.** Some readers will wonder why we need ATAs introduced in §7.3 and do not just use the PTL-based representation of semi-automata and automata presented in §7.1 and §7.2. The main reason is that, as far as we currently know, this requires a more intricate inference rule than our PITL-based one $\boxdot$**Aux**. In particular, a PTL-based rule suitable for our purposes must permit the simultaneous introduction of *multiple* auxiliary propositional variables analogous to the one French and Reynolds [FR03] were compelled to employ for QPTL without past time (see also [KM08]).

11. Existing Completeness Proofs for Omega-Regular Logics

We now compare our axiomatic completeness proof with related ones for other omega-regular logics. Here is a list of a number of such formalisms:

- Logics with nonelementary complexity:
  - The *Second-Order Theory of Successor (S1S)* [Büc62]
  - *Regular Logic* [Pae89] (This includes a PITL subset.)
  - Various temporal logics with quantification:
    * QPTL (with and without past time) (e.g., see [KM08])
    * Quantified ITL with finite domains [Mos00]
- Logics with elementary complexity:
  - *Extended Propositional Linear-Time Temporal Logic (ETL)* [Wol83]
  - *Linear-Time μ-Calculus (νTL)* [BKP86, BB89]
  - *Dynamic Linear Time Temporal Logic* [HT99]

Kröger and Merz [KM08] summarise QPTL and νTL and some axiomatisations. See also the earlier surveys about the expressiveness of various formalisms such as PTL and QPTL given by Lichtenstein et al. [LPZ85] and Emerson [Eme90]. Like S1S and QPTL, PITL has nonelementary complexity (e.g., see our results in collaboration with J. Halpern in [Mos83a] (reproduced in [Mos04])). In contrast, ETL and νTL have only elementary complexity.

11.1. **Omega-Regular Logics with Nonelementary Complexity.** Let us consider axiomatic completeness for omega-regular logics which, like PITL, have nonelementary complexity. We later discuss some with elementary complexity in §11.2.

We are not the first to consider a version of quantifier-free PITL with infinite time. Paech [Pae89] in a workshop paper presents completeness proofs for Gentzen-style axiom systems for versions of a *Regular Logic* with branching-time and linear-time and both finite and infinite time (see also [Pae88]). The linear-time variant LRL can be regarded as PITL with the addition of a binary temporal operator *unless*. Paech's framework is presented in a rather different way from ours to accommodate both branching-time and linear-time models of time, with the overwhelming emphasis on the branching-time one. Perhaps more significantly, the chop-star operator A^* in LRL is limited, like Kleene star, to finitely many iterations (we look at a closely related PITL subset, called by us PITL^K, in §5.3). Due to a theorem of Thomas [Tho79] (which we discuss and use in §5.3 and Section 6), LRL has omega-regular expressiveness, although it is less succinct than full PITL. Paech's restricted chop-star does not support chop-omega's infinite iteration. Indeed, Thomas' theorem is not at all mentioned in the completeness proof and does not serve as a bridge in the way we apply it in Section 6. Paech's stimulating and valuable presentation is quite detailed, especially in the extended version [Pae88]. Nevertheless, in our opinion (based on many years of experience with doing proofs in ITL), its treatment of LRL needs some clarification, as the following points demonstrate:

- The unwinding of chop-star does not take into account that for induction over time to work in PITL, individual iterations need to take at least two states. This contrasts with our Axioms **P9** and **P10** in Table 2 and an analogous one which Bowman and Thompson use in [BT03]. Kono's tableaux-based decision procedure for PITL [Kon95] likewise ensures that iterations have more than one state.
- The proof system includes nonconventional rules requiring some temporal formulas to be in a form analogous to regular expressions.
- The main proof concerns a branching-time semantics. In contrast, only a couple of sentences are devoted to extending the proof to a linear-time interval framework appropriate for LRL.
- The completeness proof uses constructions involving deterministic automata for finite words. It also mentions Thomas' theorem which ensures omega-regular expressiveness of LRL. Now the proof by Choueka and Peleg [CP83] of Thomas' theorem using standard deterministic omega automata quite clearly shows the link between LRL and these automata. However Paech does not discuss how the LRL completeness proof relates to techniques previously developed by McNaughton [McN66] and others for building deterministic omega automata from deterministic automata for finite words in order to recognise omega-regular languages. Some kind of explicitly described adaptation of such methods seems to us practically unavoidable. In contrast, our proof quite clearly benefits from this work as we discuss in detail in §8.
- Except for the LRL construct L_0 (the same as *empty* in PITL), no derived interval-oriented operators are defined (e.g, to examine prefix subintervals or to perform a test in a finite interval's final state). Moreover, it does not appear that the LRL proof system was ever used for anything.
- One minor puzzling feature of the LRL axiom system is that in its stated form, the linear-time proof rules for Paech's unary construct $\bigcirc A$ (which is actually the weak-next operator Ⓦ mentioned by us in Table 3) ensure that every state has a successor state.

This clearly forces the linear-time variant to be limited to infinite state sequences. In practice, such a requirement is counterproductive for LRL, which permits finite time and in particular has a primitive finite-time construct L_1 that is identical to our own construct *skip* for two-state intervals. The LRL formula L_1^* is used in rules to force finite intervals. The LRL proof rules for $\bigcirc$ which impose infinite time clash with rules containing the formula L_1^* and likewise with rules having L_0 to specify one-state intervals. However, the difficulty with the LRL operator $\bigcirc$ and infinite intervals seems to be an easily correctable oversight.

Unfortunately, no subsequent versions of Paech's completeness proof for LRL with more explanations and clarifications have been published. Indeed, the difficulties faced at the time by Paech and others such as Rosner and Pnueli [RP86] (discussed below) when attempting to develop complete axiomatisations of versions of ITL with infinite time were such that subsequent published work in this area did not appear until over ten years later. Incidentally, the manner of Paech's proof based on *Propositional Dynamic Logic* (PDL) [FL79, HKT00] and the associated *Fischer-Ladner closures* suggests that it could have connections with much later research by Henriksen and Thiagarajan [HT99] on axiomatising *Dynamic Linear Time Temporal Logic*, a formalism combining PTL and PDL which we shortly mention in §11.2. On the other hand, our own PITL completeness proof here and our earlier one for PITL with just finite time [Mos04] do not involve Fischer-Ladner closures.

Completeness proofs for logics such as S1S [Sie70], QPTL with past time [KP95, KP02] and without past time [FR03] and one by us for quantified ITL with finite domains [Mos00] use quantified formulas encoding omega automata and explicit deductions involving nontrivial techniques to complement them. As we already noted in Section 1, our earlier axiomatic completeness proof [Mos00] for quantified ITL with finite domains requires the use of quantifiers and does not work when formulas were limited to have just propositional variables. French and Reynold's [FR03] axiom system for QPTL without past time contains a nontrivial inference rule for introducing a variable number of auxiliary variables. This inference rule is required by the automata-based completeness proof.

The axiomatic completeness proofs for the logics with quantification just mentioned with nonelementary complexity involve using quantified auxiliary variables to re-express a formula A as another semantically equivalent formula $\exists p_1 \ldots p_n. X$, where $\exists$ for QPITL and QPTL is defined earlier in Section 2. Here $p_1, \ldots, p_n$ are the auxiliary variables and X is a formula in a much simpler logical subset, such as some version of (quantifier-free) PTL. Axiomatic completeness for the subset is much easier to show than for the original logic. Completeness is then proved by the standard technique of demonstrating that any consistent formula A (i.e., not deducibly false) in the full logic is also satisfiable. In particular, we deduce as a theorem the equivalence $A \equiv \exists p_1 \ldots p_n. X$. Now from this, the assumed logical consistency of A and simple propositional reasoning, we readily obtain consistency for $\exists p_1 \ldots p_n. X$. Standard reasoning about quantifiers then ensures X is consistent. Completeness for the logical subset yields a model for X which can also serve as one for A. Normally in such completeness proofs, the formula X encodes some kind of omega automaton such as a nondeterministic Büchi automata. The details are not relevant for our purposes here. The deduction of the equivalence $A \equiv \exists p_1 \ldots p_n. X$ in these proofs has always involved explicitly embedding nontrivial techniques for manipulating such omega automata.

In contrast to our approach, most of the established axiomatic completeness proofs for logics with nonelementary complexity need quantifiers. The one exception is Paech's Regular Logic, which does not have quantifiers and in linear time is like our PITL^K, the

subset of PITL without chop-omega defined earlier in §5.3. Our quantifier-free proof also benefits from the hierarchical application of some previously obtained semantic theorems and related techniques expressible as valid formulas in restricted versions of PITL (such as PITL with just finite time). This largely spares us from explicit, tricky reasoning about complementing omega automata. Once we have ensured axiomatic completeness for these versions of PITL, valid formulas in them can be immediately deduced as theorems. For example, we invoke (without proof) the theorem of Thomas at the end of [Tho79] to show that PITL^K has the same expressiveness as full PITL. Our completeness proof then combines this result with completeness for PITL^K to demonstrate that any PITL formula is deducibly equivalent to one in PITL^K.

Our completeness proof for PITL with both finite and infinite time does not actually require a proof of the axiomatic completeness of a version of PTL with this time model because Axiom **VPTL** in Table 2 includes all substitution instances of valid PTL formulas. For our purposes, even axiomatic completeness for PTL^U can be based on a reduction to PTL which invokes Axiom **VPTL**. However, as we noted in §10.1, some alternative, lower level axioms for the PITL axiom system can be used which would actually involve the reliance on a complete PTL axiom system. Our older axiom system for PITL with just finite time in Table 3 includes explicit axioms of this sort but of course can be readily modified to similarly use just a version of Axiom **VPTL** for finite time.

Even if we choose to use the alternative axioms and therefore explicitly rely on some provably complete PTL axiom system, the proofs are fairly easy to obtain via tableaux and other means (e.g., see Gabbay et al. [GPSS80], Lichtenstein and Pnueli [LP00], Kröger and Merz [KM08] and Moszkowski [Mos07]). Such methods often have associated practical decision procedures which in many cases are not so hard to implement. This contrasts with the explicit encoding in deductions of much more difficult automata-theoretic and combinatorical techniques to complement omega-regular languages in completeness proofs for other omega-regular logics with nonelementary complexity such as S1S [Sie70] and two versions of QPTL [KP02, FR03]. Furthermore, the completeness proofs for QPTL in any case also rely on reductions to some form of axiomatic completeness for PTL (which, like in our presentation, can be used without reproving it). Those QPTL axiom systems could alternatively be modified to include a suitable version of our Axiom **VPTL**. So even if we add a few extra axioms for PTL, we still feel justified in regarding our approach, which is partly based on invoking Thomas' theorem without having to encode a proof of it in deductions, as indeed being much more implicit than previous completeness proofs for omega-regular logics with nonelementary complexity such as S1S and QPTL.

Remark 11.1. As noted above, unlike previous automata-based approaches, ours avoids explicitly defining omega automata and embedding various associated explicit deductions concerning complicated proofs of some known results about them. Nevertheless, omega automata can be used in a simple semantic argument ensuring that for any satisfiable PITL formula, there exists some satisfiable PTL formula which implies it. This is because any omega-regular language can be recognised by such an automaton which itself is encodable in a QPTL formula of the form $\exists p_1 \ldots p_n. X'$, for some PTL formula X'. So for any PITL formula, there is some semantically equivalent QPTL formula of this kind and its quantifier-free part therefore implies the PITL formula. Clearly, the PITL formula is satisfiable iff the PTL subformula is.

Rosner and Pnueli's version of PITL [RP86] with infinite time and without chop-star is not an omega-regular logic since it has the (more limited) expressiveness of conventional PTL. Nevertheless, it in common with S1S, QPTL and PITL has nonelementary computational complexity. Rosner and Pnueli's complete axiom system includes a complicated inference rule which requires the construction of a table.

11.2. **Omega-Regular Logics with Elementary Complexity.** As we previously noted, ETL, νTL and Dynamic Linear Time Temporal Logic have only elementary complexity. Wolper [Wol82, Wol83] proves axiomatic completeness for ETL but Banieqbal and Barringer [BB86] later present a correction to Wolper's axiom system and proof requiring a table-based inference rule. Walukiewicz [Wal95] is the first to show axiomatic completeness for the *modal mu-calculus* [Koz83, Sti01, BS06] which subsumes νTL. Kaivola's [Kai95] subsequent less complicated completeness proof for just νTL uses a partially semantic approach which has some similar aims to ours for PITL, but is nevertheless technically quite different. It involves a clever normal form and tableaux. Every formula is shown to be deducibly equivalent to one in the normal form. We believe that our proof, although longer, is in certain respects more natural and straightforward than even Kaivola's at the deductive level.

Dynamic Linear Time Temporal Logic combines PTL and *Propositional Dynamic Logic (PDL)* [FL79, HKT00] in a linear-time framework with infinite time. The axiom system for this formalism has axioms concerning a variety of transitions [HT99]. The completeness proof is an adaptation of an earlier one for PDL by Kozen and Parikh [KP81]. It uses consistent sets of formulas.

12. FUTURE WORK

Our plans include using the axiom system as a hierarchical basis for completeness of PITL variants with weak chop and chop-star taken as primitives as well as quantification. Further possibilities include multiple time granularities (see our work [Mos95] for finite time), a temporal Hoare logic and also logics such as QPTL (by encoding within QPTL a complete axiom system for quantified PITL instead using of omega automata). The last would show interval logics can be applied to point-based ones.

In [Mos04], we used semantic techniques to prove axiomatic completeness for PITL with finite time by a simple reduction to an equally expressiveness subset called by us *Fusion Logic* and closely related to *Propositional Dynamic Logic (PDL)* [FL79, HKT00]. Fusion Logic, like some variants of PDL, uses discrete linear sequences of states instead of binary relations as its semantic basis. Some of the semantic techniques we presented in Section 6 for reducing PITL to its expressively equivalent subset $PITL^K$ by eliminating instances of chop-omega could shorten the completeness proof for Fusion Logic in [Mos04], since that proof contains a similar elimination of chop-star by reduction down to PTL. Furthermore, our completeness proof for PITL with just finite time in [Mos04] uses a separate complete axiom system for Fusion Logic. This now seems unnecessary for the overall completeness proof for PITL with finite time. Instead, the PITL axiom system should also suffice for Fusion Logic in view of our positive experiences with the current much more streamlined approach for PITL with infinite time.

The PITL operators $\diamondsuit$ and $\boxdot$ for finite prefix subintervals play a major role in our new completeness proof and appear worthy of more consideration. For example, we have

recently studied techniques for reasoning about them with *time reversal* [Mos11]. This is a natural mathematical way to exploit the symmetry of time in finite intervals. We can show the validity of suitable finite-time formulas concerning ⌶ and prefix subintervals from the validity of analogous ones for □ and suffix subintervals which themselves might even be in conventional PTL with the operator *until*. The time symmetry considered here only applies to finite intervals. However, a valid finite-time formula obtained in this way can sometimes then be generalised to infinite intervals. One potential use of time reversal is to provide an algorithmic reduction of suitable higher-level PITL formulas to lower-level PTL ones for model checking. It also helps extend compositional techniques we described in [Mos94, Mos96, Mos98].

Conclusions

We have presented a simple axiom system for PITL with infinite time and proved completeness using a semantic framework and reductions to finite time and PTL. Our axiom system is demonstrably simpler than the one which Paech presents for LRL, even though we support omega-iteration and LRL does not. Moreover, the explicitly stated deductions in our proof can be regarded as being technically less complex then others for quantified omega-regular logics with nonelementary complexity such as S1S and QPTL. This is because known completeness proofs for those logics involve an *explicit* deductive embedding of proofs of theorems about complementing omega-regular languages and require reasoning about nontrivial algorithms (typically utilising quantifier-based encodings of omega automata). Such completeness proofs therefore do not merely use one such theorem but incorporate significant aspects of its complicated proof, in effect *reproving it.* In contrast, we simply invoke Thomas' theorem without referring to how it is proved. In our opinion, this conforms much more to the conventional mathematical practice of using previously established theorems, even hard-to-prove ones, as modular "black boxes". However, we appreciate that some readers will argue about the significance of this technical point.

The overall results we have described in our new completeness proof seem to complement our recent analysis of PTL using PITL [Mos07]. One surprise during the development of our completeness proof concerned how much explicit deductions could be minimised by application of valid properties proved with semi-automata and automata on finite words. Another unexpected benefit arose from the insights into time reversal.

Acknowledgements

References

[BB86] Behnam Banieqbal and Howard Barringer. A study of an extended temporal logic and a temporal fixed point calculus. Technical Report UMCS-86-10-2, Dept. of Computer Science, University of Manchester, England, October 1986. revised June 1987.

[BB89]   Behnam Banieqbal and Howard Barringer. Temporal logic with fixed points. In Behnam Banieqbal, Howard Barringer, and Amir Pnueli, editors, *Temporal Logic in Specification, Proceedings (Altrincham, UK, April, 1987)*, volume 398 of *LNCS*, pages 62–74, Berlin, 1989. Springer-Verlag.

[BBK+04]   Michael Balser, Simon Bäumler, Alexander Knapp, Wolfgang Reif, and Andreas Thums. Interactive verification of UML state machines. In Jim Davies, Wolfram Schulte, and Michael Barnett, editors, *Proc. 6th International Conference on Formal Engineering Methods (ICFEM 2004)*, volume 3308 of *LNCS*, pages 434–448. Springer-Verlag, 2004.

[BBN+10]   Simon Bäumler, Michael Balser, Florian Nafz, Wolfgang Reif, and Gerhard Schellhorn. Interactive verification of concurrent systems using symbolic execution. *AI Communications*, 23(2–3):285–307, 2010.

[BKP86]   Howard Barringer, Ruurd Kuiper, and Amir Pnueli. A really abstract concurrent model and its temporal logic. In *Proc. 13th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'86)*, pages 173–183. ACM, 1986.

[BS06]   Julian Bradfield and Colin Stirling. Modal mu-calculi. In *The Handbook of Modal Logic*, pages 721–756. Elsevier, 2006.

[BSTR11]   Simon Bäumler, Gerhard Schellhorn, Bogdan Tofan, and Wolfgang Reif. Proving linearizability with temporal logic. *Formal Aspects of Computing*, 23:91–112, 2011.

[BT03]   Howard Bowman and Simon J. Thompson. A decision procedure and complete axiomatization of finite Interval Temporal Logic with projection. *Journal of Logic and Computation*, 13(2):195–239, April 2003.

[Büc62]   J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. Int. Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–12. Stanford University Press, 1962.

[Che80]   Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, England, 1980.

[Cho74]   Yaacov Choueka. Theories of automata on omega-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8(2):117–141, 1974.

[CP83]   Yaacov Choueka and David Peleg. A note on ω-regular languages. *Bulletin of the European Association for Theoretical Computer Science*, 21:21–23, October 1983.

[CZ97]   A. Cau and H. Zedan. Refining Interval Temporal Logic specifications. In M. Bertran and T. Rus, editors, *Transformation-Based Reactive Systems Development*, volume 1231 of *LNCS*, pages 79–94. AMAST, Springer-Verlag, 1997.

[Dut95]   Bruno Dutertre. Complete proof systems for first order Interval Temporal Logic. In *Proc. 10th Ann. IEEE Symp. on Logic in Computer Science (LICS '95)*, pages 36–43, Los Alamitos, Calif., USA, June 1995. IEEE Computer Society Press.

[DZ08]   Zhenhua Duan and Nan Zhang. A complete axiomatization of propositional projection temporal logic. In *2nd IEEE/IFIP Int'l Symp. on Theoretical Aspects of Software Eng. (TASE 2008)*, pages 271–278. IEEE Computer Society Press, 2008.

[DZK12]   Zhenhua Duan, Nan Zhang, and Maciej Koutny. A complete axiomatization of propositional projection temporal logic. *Theor. Comp. Sci.*, 2012. doi: 10.1016/j.tcs.2012.01.026.

[Eme90] E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 16, pages 995–1072. Elsevier/MIT Press, Amsterdam, 1990.

[Fis11] Michael Fisher. *An Introduction to Practical Formal Methods Using Temporal Logic*. John Wiley & Sons, 2011.

[FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, April 1979.

[FR03] Tim French and Mark Reynolds. A sound and complete proof system for QPTL. In P. Balbiani, N-Y. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 4, pages 127–148. King's College Publications, London, 2003.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th Ann. ACM Symp. on Principles of Programming Languages (POPL '80)*, pages 163–173. ACM, 1980.

[Gue07] Dimitar P. Guelev. Probabilistic interval temporal logic and duration calculus with infinite intervals: Complete proof systems. *Logical Methods in Computer Science*, 3(3), 2007.

[HC96] George E. Hughes and Max J. Cresswell. *A New Introduction to Modal Logic*. Routledge, London, 1996.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, Massachusetts, 2000.

[HMM83] J. Halpern, Z. Manna, and B. Moszkowski. A hardware semantics based on temporal intervals. In J. Diaz, editor, *Proc. 10th Int'l. Colloquium on Automata, Languages and Programming (ICALP '83)*, volume 154 of *LNCS*, pages 278–291, Berlin, 1983. Springer-Verlag.

[HT99] Jesper G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1-3):187–207, 1999.

[IEE08] IEEE. *Standard for the Functional Verification Language e, Standard 1647-2008*. ANSI/IEEE, New York, 2008. Produced by the **e** Functional Verification Language Working Group.

[ITL12] Interval Temporal Logic webpages. `http://www.tech.dmu.ac.uk/STRL/ITL/`, 2012.

[Kai95] Roope Kaivola. Axiomatising linear time mu-calculus. In Insup Lee and Scott A. Smolka, editors, *CONCUR '95*, volume 962 of *LNCS*, pages 423–437. Springer-Verlag, 1995.

[Kam68] Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[KM08] Fred Kröger and Stephan Merz. *Temporal Logic and State Systems*. Texts in Theoretical Computer Science (An EATCS Series). Springer-Verlag, 2008.

[Kon95] Shinji Kono. A combination of clausal and non-clausal temporal logic programs. In Michael Fisher and Richard Owens, editors, *Executable Modal and Temporal Logics*, volume 897 of *LNCS*, pages 40–57, Berlin, February 1995. Springer-Verlag.

[Koz83] Dexter Kozen. Results on the propositional μ-calculus. *Theor. Comp. Sci.*, 27(3):333–354, 1983.

[KP81] Dexter Kozen and Rohit Parikh. An elementary proof of the completeness of PDL. *Theor. Comp. Sci.*, 14:113–118, 1981.

[KP95]   Y. Kesten and A. Pnueli. A complete proof system for QPTL. In *Proc. 10th IEEE Symp. on Logic in Computer Science (LICS'95)*, pages 2–12. IEEE Computer Society Press, 1995.

[KP02]   Y. Kesten and A. Pnueli. Complete proof system for QPTL. *Journal of Logic and Computation*, 12(5):701–745, December 2002.

[Lam02] Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Professional, 2002.

[LP00]   Orna Lichtenstein and Amir Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In R. Parikh et al., editors, *Logics of Programs*, volume 193 of *LNCS*, pages 196–218, Berlin, 1985. Springer-Verlag.

[McN66] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Inf. and Control*, 9(5):521–530, October 1966.

[Mor99] Matthew J. Morley. Semantics of temporal *e*. In T. F. Melham and F. G. Moller, editors, Banff'99 *Higher Order Workshop: Formal Methods in Computation, Ullapool, Scotland, 9–11 Sept. 1999*, pages 138–142. University of Glasgow, Department of Computing Science Technical Report, 1999.

[Mos83a] B. Moszkowski. *Reasoning about Digital Circuits*. PhD thesis, Department of Computer Science, Stanford University, June 1983. Technical report STAN–CS–83–970.

[Mos83b] B. Moszkowski. A temporal logic for multi-level reasoning about hardware. In *Proc. 6th Int'l. Symp. on Computer Hardware Description Languages*, pages 79–90, Pittsburgh, Pennsylvania, 1983. North-Holland Pub. Co.

[Mos85] B. Moszkowski. A temporal logic for multilevel reasoning about hardware. *Computer*, 18:10–19, 1985.

[Mos86] B. Moszkowski. *Executing Temporal Logic Programs*. Cambridge University Press, Cambridge, England, 1986.

[Mos94] Ben Moszkowski. Some very compositional temporal properties. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi (PROCOMET'94)*, volume A-56 of *IFIP Transactions*, pages 307–326. IFIP, Elsevier Science B.V. (North–Holland), 1994.

[Mos95] Ben Moszkowski. Compositional reasoning about projected and infinite time. In *Proc. 1st IEEE Int'l Conf. on Engineering of Complex Computer Systems (ICECCS'95)*, pages 238–245. IEEE Computer Society Press, 1995.

[Mos96] Ben Moszkowski. Using temporal fixpoints to compositionally reason about liveness. In He Jifeng, John Cooke, and Peter Wallis, editors, *BCS-FACS 7th Refinement Workshop*, electronic Workshops in Computing, London, 1996. BCS-FACS, Springer-Verlag and British Computer Society.

[Mos98] Ben Moszkowski. Compositional reasoning using Interval Temporal Logic and Tempura. In Willem-Paul de Roever, Hans Langmaack, and Amir Pnueli, editors, *Compositionality: The Significant Difference*, volume 1536 of *LNCS*, pages 439–464, Berlin, 1998. Springer-Verlag.

[Mos00] Ben Moszkowski. A complete axiomatization of Interval Temporal Logic with infinite time (extended abstract). In *Proc. 15th Ann. IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 242–251. IEEE Computer Society Press, June 2000.

[Mos04] Ben Moszkowski. A hierarchical completeness proof for Propositional Interval Temporal Logic with finite time. *Journal of Applied Non-Classical Logics*, 14(1–2):55–104, 2004. Special issue on Interval Temporal Logics and Duration Calculi. V. Goranko and A. Montanari, guest editors.

[Mos07] Ben Moszkowski. Using temporal logic to analyse temporal logic: A hierarchical approach based on intervals. *Journal of Logic and Comp.*, 17(2):333–409, April 2007.

[Mos11] Ben Moszkowski. Compositional reasoning using intervals and time reversal. In *18th Int'l Symp. on Temporal Representation and Reasoning (TIME 2011)*, pages 107–114. IEEE Computer Society, 2011.

[MWD11] Dapeng Mo, Xiaobing Wang, and Zhenhua Duan. Asynchronous communication in MSVL. In Shengchao Qin and Zongyan Qiu, editors, *13th Int'l Conf. on Formal Engineering Methods (ICFEM 2011)*, volume 6991 of *LNCS*, pages 82–97. Springer-Verlag, 2011.

[OD08] Ernst-Rüdiger Olderog and Henning Dierks. *Real-Time Systems: Formal Specification and Automatic Verification*. Cambridge University Press, Cambridge, England, 2008.

[Pae88] Barbara Paech. Gentzen-systems for propositional temporal logics. Technical Report 88/01, Institut für Informatik, Ludwig-Maximilians-Universität, Munich, Germany, February 1988.

[Pae89] Barbara Paech. Gentzen-systems for propositional temporal logics. In E. Börger, H. Kleine Büning, and M. M. Richter, editors, *Proceedings of the 2nd Workshop on Computer Science Logic (CSL'88)*, volume 385 of *LNCS*, pages 240–253. Springer-Verlag, 1989.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proc. 18th Ann. IEEE Symp. on the Foundation of Computer Science (FOCS)*, pages 46–57. IEEE Computer Society Press, 1977.

[RP86] R. Rosner and A. Pnueli. A choppy logic. In *Proc. 1st Ann. IEEE Symp. on Logic in Computer Science (LICS'86)*, pages 306–313. IEEE Computer Society Press, June 1986.

[RSSB98] Wolfgang Reif, Gerhard Schellhorn, Kurt Stenzel, and Michael Balser. Structured specifications and interactive proofs with KIV. In Wolfgang Bibel and Peter H. Schmitt, editors, *Automated Deduction – A Basis for Applications, Volume II: Systems and Implementation Techniques*, pages 13–39. Kluwer Academic Publishers, Dordrecht, 1998.

[Sie70] Dirk Siefkes. *Decidable Theories I: Büchi's Monadic Second Order Successor Arithmetic*, volume 120 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1970.

[Sti01] Colin Stirling. *Modal and Temporal Properties of Processes*. Springer-Verlag, New York, 2001.

[Tho79] Wolfgang Thomas. Star-free regular sets of ω-sequences. *Inf. and Control*, 42(2):148–156, August 1979.

[Tho90] W. Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 4, pages 133–191. Elsevier/MIT Press, Amsterdam, 1990.

[Tho97] W. Thomas. Languages, automata, and logic. In G. Rozenburg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3: Beyond words, chapter 7,

pages 389–455. Springer-Verlag, Berlin, 1997.

[TSOR04] Andreas Thums, Gerhard Schellhorn, Frank Ortmeier, and Wolfgang Reif. Interactive verification of Statecharts. In Hartmut Ehrig, Werner Damm, Jörg Desel, Martin Große-Rhode, Wolfgang Reif, Eckehard Schnieder, and Engelbert Westkämper, editors, *SoftSpez Final Report*, volume 3147 of *LNCS*, pages 355–373. Springer-Verlag, 2004.

[Wal95] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional mu-calculus. In *Proc. 10th Ann. Symp. on Logic in Computer Science (LICS'95)*, pages 14–24. IEEE Computer Society Press, 1995.

[Wol82] P. L. Wolper. *Specification and Synthesis of Communicating Processes Using an Extended Temporal Logic*. PhD thesis, Department of Computer Science, Stanford University, 1982.

[Wol83] P. [L.] Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1-2):72–99, 1983.

[WX04] Hanpin Wang and Qiwen Xu. Completeness of temporal logics over infinite intervals. *Discrete Applied Mathematics*, 136(1):87–103, 2004.

[ZDT12] Nan Zhang, Zhenhua Duan, and Cong Tian. A cylinder computation model for many-core parallel computing. *Theor. Comp. Sci.*, 2012. doi: `10.1016/j.tcs.2012.02.011`.

[ZH04] Zhou Chaochen and Michael R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. Monographs in Theoretical Computer Science (An EATCS series). Springer-Verlag, 2004.

[ZHR91] Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.

[ZZC99] Shikun Zhou, Hussein Zedan, and Antonio Cau. A framework for analysing the effect of "change" in legacy code. In *15th IEEE International Conference on Software Maintenance (ICSM'99)*, pages 411–420, 1999.

Appendix A. Some PITL theorems and Their Proofs

This appendix gives a representative set of PITL theorems and derived inference rules together with their proofs. Many are used either directly or indirectly in the completeness proof for PITL with both finite and infinite time. We have partially organised the material, particularly in §A.2, along the lines of some standard modal logic systems [Che80, HC96].

The PITL theorems and derived rules have a shared index sequence (e.g., **T1**–**T3** are followed by **DR4** rather than **DR1**). We believe that this convention simplifies locating material in this appendix and also in Table 4 found earlier in §7.4.

Proof steps can refer to axioms, inference rules, previously deduced theorems, derived inference rules and also the following:

- assump.: Assumptions which are regarded as being previously deduced.
- **Prop**: Conventional nonmodal propositional reasoning (by restricted application of Axiom **VPTL**) and modus ponens.
- ⊃-**chain**: A chain of implications.
- ≡-**chain**: A chain of equivalences.

In principle, $\supset$**-chain** and $\equiv$**-chain** are subsumed by **Prop** but are used here to make the reasoning more explicit.

- **PITLF**: Our assumption of axiomatic completeness for PITL with just finite time permits any valid implication of the form *finite* $\supset A$.

A.1. **Some Basic Properties of Chop.**

We now consider deducing various simple properties of chop and the associated operators $\diamondsuit\!\!\!\!\diamond$, $\boxdot$, $\diamondsuit$ and $\square$ which have a wide range of uses.

T1 $\vdash$ $\boxdot(A \supset A') \supset (A ^\frown B) \supset (A' ^\frown B)$

1	$\vdash$	$B \supset B$		**Prop**
2	$\vdash$	$\square(B \supset B)$		1,$\square$**Gen**
3	$\vdash$	$\boxdot(A \supset A') \wedge \square(B \supset B) \supset (A ^\frown B) \supset (A' ^\frown B)$		**P8**
4	$\vdash$	$\boxdot(A \supset A') \supset (A ^\frown B) \supset (A' ^\frown B)$		2,3,**Prop**

T2 $\vdash$ $\square(B \supset B') \supset (A ^\frown B) \supset (A ^\frown B')$

1	$\vdash$	*finite* $\supset (A \supset A)$		**Prop**
2	$\vdash$	$\boxdot(A \supset A)$		1,$\boxdot$**FGen**
3	$\vdash$	$\boxdot(A \supset A) \wedge \square(B \supset B') \supset (A ^\frown B) \supset (A ^\frown B')$		**P8**
4	$\vdash$	$\square(B \supset B') \supset (A ^\frown B) \supset (A ^\frown B')$		2,3,**Prop**

T3 $\vdash$ $\square(B \equiv B') \supset (A ^\frown B) \equiv (A ^\frown B')$

1	$\vdash$	$\square(B \equiv B') \equiv \square(B \supset B') \wedge \square(B' \supset B)$		**VPTL**
2	$\vdash$	$\square(B \supset B') \supset (A ^\frown B) \supset (A ^\frown B')$		**T2**
3	$\vdash$	$\square(B' \supset B) \supset (A ^\frown B') \supset (A ^\frown B)$		**T2**
4	$\vdash$	$\square(B \equiv B') \supset (A ^\frown B) \equiv (A ^\frown B')$		2,3,**Prop**

The following derived variant of Inference Rule $\boxdot$**FGen** omits the subformula *finite*:

DR4 $\vdash A \Rightarrow \vdash \boxdot A$

1	$\vdash$	A	assump.
2	$\vdash$	*finite* $\supset A$	1,**Prop**
3	$\vdash$	$\boxdot A$	2, $\boxdot$**FGen**

The derived inference rule **DR4** can also be referred to as $\boxdot$**Gen** (analogous to the inference rule $\square$**Gen**).

DR5 $\vdash A \supset A' \Rightarrow \vdash (A ^\frown B) \supset (A' ^\frown B)$

1	$\vdash$	$A \supset A'$	assump.
2	$\vdash$	$\boxdot(A \supset A')$	1,**DR4**
3	$\vdash$	$\boxdot(A \supset A') \supset (A ^\frown B) \supset (A' ^\frown B)$	**T1**
4	$\vdash$	$A ^\frown B \supset A' ^\frown B$	2,3,**MP**

DR6 $\vdash A \equiv A' \Rightarrow \vdash (A ^\frown B) \equiv (A' ^\frown B)$

1	$\vdash$	$A \equiv A'$	assump.
2	$\vdash$	$A \supset A'$	1,**Prop**
3	$\vdash$	$A ^\frown B \supset A' ^\frown B$	2,**DR5**
4	$\vdash$	$A' \supset A$	1,**Prop**
5	$\vdash$	$A' ^\frown B \supset A ^\frown B$	4,**DR5**

6 ⊢ $A {}^\frown B$ ≡ $A' {}^\frown B$ 3,5,**Prop**

DR7   ⊢ $A \supset B$ ⇒   ⊢ $\lozenge\!\!\!\lozenge A \supset \lozenge\!\!\!\lozenge B$
1 ⊢ A ⊃ B assump.
2 ⊢ $A {}^\frown true$ ⊃ $B {}^\frown true$ 1,**DR5**
3 ⊢ $\lozenge\!\!\!\lozenge A$ ⊃ $\lozenge\!\!\!\lozenge B$ 2,def. of $\lozenge\!\!\!\lozenge$

DR8   ⊢ $A \equiv B$ ⇒   ⊢ $\lozenge\!\!\!\lozenge A \equiv \lozenge\!\!\!\lozenge B$
1 ⊢ A ≡ B assump.
2 ⊢ $A {}^\frown true$ ≡ $B {}^\frown true$ 1,**DR6**
3 ⊢ $\lozenge\!\!\!\lozenge A$ ≡ $\lozenge\!\!\!\lozenge B$ 2,def. of $\lozenge\!\!\!\lozenge$

DR9   ⊢ $B \supset B'$ ⇒   ⊢ $(A {}^\frown B) \supset (A {}^\frown B')$
1 ⊢ B ⊃ B' assump.
2 ⊢ $\Box(B \supset B')$ $\Box$**Gen**
3 ⊢ $\Box(B \supset B')$ ⊃ $(A {}^\frown B) \supset (A {}^\frown B')$ **T2**
4 ⊢ $A {}^\frown B$ ⊃ $A {}^\frown B'$ 2,3,**MP**

DR10   ⊢ $B \equiv B'$ ⇒   ⊢ $(A {}^\frown B) \equiv (A {}^\frown B')$
1 ⊢ B ≡ B' assump.
2 ⊢ B ⊃ B' 1,**Prop**
3 ⊢ $A {}^\frown B$ ⊃ $A {}^\frown B'$ 2,**DR9**
4 ⊢ B' ⊃ B 1,**Prop**
5 ⊢ $A {}^\frown B'$ ⊃ $A {}^\frown B$ 4,**DR9**
6 ⊢ $A {}^\frown B$ ≡ $A {}^\frown B'$ 3,5,**Prop**

DR11   ⊢ $A \equiv B$ ⇒   ⊢ $\lozenge A \equiv \lozenge B$
1 ⊢ A ≡ B assump.
2 ⊢ $true {}^\frown A$ ≡ $true {}^\frown B$ 1,**DR10**
3 ⊢ $\lozenge A$ ≡ $\lozenge B$ 2,def. of $\lozenge$

DR12   ⊢ $A \equiv B$ ⇒   ⊢ $\Box A \equiv \Box B$
1 ⊢ A ≡ B assump.
2 ⊢ $\neg A$ ≡ $\neg B$ 1,**Prop**
3 ⊢ $\lozenge \neg A$ ≡ $\lozenge \neg B$ 2,**DR11**
4 ⊢ $\neg \lozenge \neg A$ ≡ $\neg \lozenge \neg B$ 3,**Prop**
5 ⊢ $\Box A$ ≡ $\Box B$ 4,def. of $\Box$

DR13   ⊢ $\Box A \supset B$ ⇒   ⊢ $\Box A \supset \Box B$
1 ⊢ $\Box A$ ⊃ B assump.
2 ⊢ $\Box(\Box A \supset B)$ 1,$\Box$**Gen**
3 ⊢ $\Box(\Box A \supset B)$ ⊃ $(\Box A \supset \Box B)$ **VPTL**
4 ⊢ $\Box A$ ⊃ $\Box B$ 2,3,**MP**

T14   ⊢ $(A \wedge A') {}^\frown B$ ⊃ $A {}^\frown B$
1 ⊢ $A \wedge A'$ ⊃ A **Prop**
2 ⊢ $(A \wedge A') {}^\frown B$ ⊃ $A {}^\frown B$ 1,**DR5**

T15   ⊢ $(A \wedge A') {}^\frown B$ ⊃ $A' {}^\frown B$
1 ⊢ $A \wedge A'$ ⊃ A' **Prop**
2 ⊢ $(A \wedge A') {}^\frown B$ ⊃ $A' {}^\frown B$ 1,**DR5**

T16 $\vdash$ $(A \wedge A')^\frown B$ $\supset$ $(A^\frown B) \wedge (A'^\frown B)$

1	$\vdash$	$(A \wedge A')^\frown B$	$\supset$	$A^\frown B$	**T14**
2	$\vdash$	$(A \wedge A')^\frown B$	$\supset$	$A'^\frown B$	**T15**
3	$\vdash$	$(A \wedge A')^\frown B$	$\supset$	$(A^\frown B) \wedge (A'^\frown B)$	1,2,**Prop**

T17 $\vdash$ $(A \wedge A')^\frown B$ $\equiv$ $(A' \wedge A)^\frown B$

1	$\vdash$	$A \wedge A'$	$\equiv$	$A' \wedge A$	**Prop**
2	$\vdash$	$(A \wedge A')^\frown B$	$\equiv$	$(A' \wedge A)^\frown B$	1,**DR6**

T18 $\vdash$ $(A \vee A')^\frown B$ $\equiv$ $(A^\frown B) \vee (A'^\frown B)$

The proof for $\supset$ is immediate from axiom **P3**. Here is the proof for $\subset$:

1	$\vdash$	A	$\supset$	$A \vee A'$	**Prop**
2	$\vdash$	$A^\frown B$	$\supset$	$(A \vee A')^\frown B$	1,**DR5**
3	$\vdash$	A'	$\supset$	$A \vee A'$	**Prop**
4	$\vdash$	$A'^\frown B$	$\supset$	$(A \vee A')^\frown B$	3,**DR5**
5	$\vdash$	$(A^\frown B) \vee (A^\frown B')$	$\supset$	$(A \vee A')^\frown B$	2,4,**Prop**

T19 $\vdash$ $A^\frown B$ $\supset$ $\lozenge\!\!\!\cdot A$

1	$\vdash$	B	$\supset$	$true$	**Prop**
2	$\vdash$	$A^\frown B$	$\supset$	$A^\frown true$	1,**DR9**
3	$\vdash$	$A^\frown B$	$\supset$	$\lozenge\!\!\!\cdot A$	2,def. of $\lozenge\!\!\!\cdot$

T20 $\vdash$ $\lozenge\!\!\!\cdot empty$

1	$\vdash$	$empty^\frown true$	$\equiv$	$true$	**P5**
2	$\vdash$	$empty^\frown true$	$\supset$	$\lozenge\!\!\!\cdot empty$	**T19**
3	$\vdash$	$\lozenge\!\!\!\cdot empty$			1,2,**Prop**

T21 $\vdash$ $A^\frown B$ $\supset$ $\lozenge B$

1	$\vdash$	A	$\supset$	$true$	**Prop**
2	$\vdash$	$A^\frown B$	$\supset$	$true^\frown B$	1,**DR5**
3	$\vdash$	$A^\frown B$	$\supset$	$\lozenge B$	2,def. of $\lozenge$

A.2. **Some Properties of ▣ involving the Modal System K and Axiom D.**

The two pairs of operators $\square$ and $\lozenge$ and ▣ and $\lozenge\!\!\!\cdot$ obey various standard properties of modal logics. Axiom **VPTL** helps streamline reasoning involving $\square$ and $\lozenge$. The situation with ▣ and $\lozenge\!\!\!\cdot$ is quite different since they lack a comparable axiom. Therefore, it is especially beneficial to review some conventional modal systems which assist in organising various useful deductions involving ▣ and $\lozenge\!\!\!\cdot$.

Table 6 summarises some relevant modal systems, various associated axioms and inference rules. Chellas [Che80] and Hughes and Cresswell [HC96] give more details.

Within PITL, as in PTL, the operator $\square$ can be regarded as the conventional unary *necessity* modality L and the operator $\lozenge$ as the dual *possibility* operator M. The two operators together fulfil the requirements of the modal system *S4*. We do not need to explicitly prove versions of the *S4* axioms in Table 6 for $\square$ and $\lozenge$. Rather, any PITL formula which is a substitution instance of a valid *S4* formula involving $\square$ and $\lozenge$ can be readily deduced using the PITL proof system's Axiom **VPTL**. Similarly, inference rules

System		Axiom or inference rule	Axiom or rule name
K:		$M\,A \ \overset{\text{def}}{\equiv} \ \neg L\,\neg A$	M-def
	plus	$\vdash \ L(A \supset B) \ \supset \ (L\,A \supset L\,B)$	K
	plus	$\vdash \ A \ \Rightarrow \ \vdash \ L\,A$	N
T:	K plus	$\vdash \ L\,A \ \supset \ A$	T
$S4$:	T plus	$\vdash \ L\,A \ \supset \ LL\,A$	4
$KD4$:	K plus 4 and	$\vdash \ L\,A \ \supset \ M\,A$	D

Table 6: Some standard modal systems

based on $S4$ can be obtained with Axiom **VPTL**, Inference Rule $\Box$**Gen** (which corresponds to the inference rule N of $S4$) and modus ponens. Moreover, the PITL proof system's Axiom **VPTL** permits using *any* PITL formula which is a substitution instance of some valid PTL formula which can also contain the PTL operator $\bigcirc$. In view of all this, we do not give much further consideration to aspects of $S4$ with $\Box$ and $\Diamond$.

In contrast to $\Box$, the PITL operator $\boxdot$ does not have a comprehensive axiom analogous to **VPTL**. Therefore, we need to explicitly prove in the PITL axiom system various modal properties of $\boxdot$ and its dual $\diamondsuit$. If only finite time is allowed, then $\boxdot$ and $\diamondsuit$ act as an $S4$ system. However, $\boxdot$ with infinite time permitted does not fulfil the requirements of $S4$, or even those of the weaker modal system T, because Axiom T fails. Instead, $\boxdot$ with infinite time fulfils the requirements of the modal system $KD4$ which is strictly weaker than $S4$.

Here is a list of $KD4$'s axioms and inference rules and related PITL proofs for $\boxdot$:

K	$\vdash \ L(A \supset B) \ \supset \ (L\,A \supset L\,B)$	Theorem **T25**
N	$\vdash \ A \ \Rightarrow \ \vdash \ L\,A$	Derived Inf. Rule **DR4**
D	$\vdash \ L\,A \ \supset \ M\,A$	Theorem **T33**
4	$\vdash \ L\,A \ \supset \ LL\,A$	Theorem **T47**

If only finite time is allowed, then the implication D does not need to be regarded as an explicit axiom since it can be inferred from any proof system for $S4$.

Remark A.1. It is also worth noting that the related operators $\boxdot$ and $\diamondsuit$ (defined using weak chop in Table 1 in Section 2) obey the modal system $S4$ even when infinite time is permitted. However, we prefer to work with $\boxdot$ and $\diamondsuit$ since the use of strong chop simplifies the overall PITL completeness proof.

Conventional model logics usually take L, not M, to be primitive. When we deduce standard modal properties for $\boxdot$ and $\diamondsuit$ in our PITL axiom system, we let M, which corresponds to $\diamondsuit$, be primitive and define L to be M's dual (i.e., $L\,A \overset{\text{def}}{\equiv} \neg M\,\neg A$). This M-based approach goes well with the PITL axioms for chop. Chellas [Che80] discusses some alternative axiomatisations of modal systems with M as the primitive although none correspond directly to ours. For the system K, we can deduce implication (A.1) below for $\boxdot$ and $\diamondsuit$ (see Theorem **T23** later on) and then obtain from it together some other reasoning the more standard axiom K just presented which only mentions L:

$$\vdash \quad L(A \supset B) \ \supset \ (M\,A \ \supset \ M\,B). \tag{A.1}$$

The operators $\Box$ and $◫$ together yield a *multi-modal logic* with two necessity constructs L and L' which are commutative:

$$\vdash \quad L L' A \quad \equiv \quad L' L\, A.$$

This corresponds to our Theorem **T55** given later on.

Below are various theorems and derived inference rules about $◫$ and $\diamond$ for obtaining the axioms M-def (Theorem **T22**) and K (Theorem **T25**) found in the modal system K. The associated inference rule N was already proved above as Derived Inference Rule **DR4**. We also prove the modal axiom D (Theorem **T33**).

In the next proof's final step, recall that $\equiv$-**chain** indicates a chain of equivalences:

T22 $\vdash \quad \diamond A \quad \equiv \quad \neg ◫ \neg A$

1	$\vdash$	$A \equiv \neg\neg A$	**Prop**
2	$\vdash$	$\diamond A \equiv \diamond \neg\neg A$	1,**DR8**
3	$\vdash$	$\diamond \neg\neg A \equiv \neg\neg \diamond \neg\neg A$	**Prop**
4	$\vdash$	$\diamond \neg\neg A \equiv \neg ◫ \neg A$	3,def. of $◫$
5	$\vdash$	$\diamond A \equiv \neg ◫ \neg A$	2,4,$\equiv$-**chain**

T23 $\vdash \quad ◫(A \supset B) \quad \supset \quad \diamond A \supset \diamond B$

1	$\vdash$	$◫(A \supset B) \supset (A^{\frown} true) \supset (B^{\frown} true)$	**T1**
2	$\vdash$	$◫(A \supset B) \supset \diamond A \supset \diamond B$	1,def. of $\diamond$

T24 $\vdash \quad ◫(\neg B \supset \neg A) \quad \supset \quad (◫ A) \supset (◫ B)$

1	$\vdash$	$◫(\neg B \supset \neg A) \supset (\diamond \neg B) \supset (\diamond \neg A)$	**T23**
2	$\vdash$	$◫(\neg B \supset \neg A) \supset (\neg \diamond \neg A) \supset (\neg \diamond \neg B)$	1,**Prop**
3	$\vdash$	$◫(\neg B \supset \neg A) \supset (◫ A) \supset (◫ B)$	2,def. of $◫$

T25 $\vdash \quad ◫(A \supset B) \quad \supset \quad (◫ A) \supset (◫ B)$

1	$\vdash$	$(A \supset B) \supset (\neg B \supset \neg A)$	**Prop**
2	$\vdash$	$\neg(\neg B \supset \neg A) \supset \neg(A \supset B)$	1,**Prop**
3	$\vdash$	$◫\big(\neg(\neg B \supset \neg A) \supset \neg(A \supset B)\big)$	2,**DR4**
4	$\vdash$	$◫\big(\neg(\neg B \supset \neg A) \supset \neg(A \supset B)\big)$ $\supset ◫(A \supset B) \supset ◫(\neg B \supset \neg A)$	**T24**
5	$\vdash$	$◫(A \supset B) \supset ◫(\neg B \supset \neg A)$	3,4,**MP**
6	$\vdash$	$◫(\neg B \supset \neg A) \supset (◫ A) \supset (◫ B)$	**T24**
7	$\vdash$	$◫(A \supset B) \supset (◫ A) \supset (◫ B)$	5,6,$\supset$-**chain**

DR26 $\vdash \quad A \supset B \quad \Rightarrow \quad \vdash \quad ◫ A \supset ◫ B$

1	$\vdash$	$A \supset B$	assump.
2	$\vdash$	$◫(A \supset B)$	1,**DR4**
3	$\vdash$	$◫(A \supset B) \supset (◫ A) \supset (◫ B)$	**T25**
4	$\vdash$	$◫ A \supset ◫ B$	2,3,**MP**

DR27 $\vdash A \equiv B \quad \Rightarrow \quad \vdash ◫ A \equiv ◫ B$

1	$\vdash$	$A \equiv B$	assump.
2	$\vdash$	$A \supset B$	1,**Prop**
3	$\vdash$	$◫ A \supset ◫ B$	2,**DR26**
4	$\vdash$	$B \supset A$	1,**Prop**
5	$\vdash$	$◫ B \supset ◫ A$	4,**DR26**

6	$\vdash$	$\Box A \;\equiv\; \Box B$	3,5,**Prop**

T28 $\quad\vdash\quad \Box(A \land B) \;\equiv\; \Box A \land \Box B$

1	$\vdash$	$(A \land B) \;\supset\; A$	**Prop**
2	$\vdash$	$\Box(A \land B) \;\supset\; \Box A$	1,**DR26**
3	$\vdash$	$(A \land B) \;\supset\; B$	**Prop**
4	$\vdash$	$\Box(A \land B) \;\supset\; \Box B$	3,**DR26**
5	$\vdash$	$A \;\supset\; (B \supset (A \land B))$	**Prop**
6	$\vdash$	$\Box A \;\supset\; \Box(B \supset (A \land B))$	5,**DR26**
7	$\vdash$	$\Box(B \supset (A \land B)) \;\supset\; \big(\Box B \supset \Box(A \land B)\big)$	**T25**
8	$\vdash$	$\Box A \land \Box B \;\supset\; \Box(A \land B)$	6,7,**Prop**
9	$\vdash$	$\Box(A \land B) \;\equiv\; \Box A \land \Box B$	2,4,8,**Prop**

T29 $\quad\vdash\quad \Box(A \equiv B) \;\equiv\; \Box(A \supset B) \land \Box(B \supset A)$

1	$\vdash$	$(A \equiv B) \;\equiv\; (A \supset B) \land (B \supset A)$	**Prop**
2	$\vdash$	$\Box(A \equiv B) \;\equiv\; \Box\big((A \supset B) \land (B \supset A)\big)$	1,**DR27**
3	$\vdash$	$\Box\big((A \supset B) \land (B \supset A)\big) \;\equiv\; \Box(A \supset B) \land \Box(B \supset A)$	**T28**
4	$\vdash$	$\Box(A \equiv B) \;\equiv\; \Box(A \supset B) \land \Box(B \supset A)$	2,3,$\equiv$-**chain**

T30 $\quad\vdash\quad \Box(A \equiv A') \;\supset\; (A ^\frown B) \equiv (A' ^\frown B)$

1	$\vdash$	$\Box(A \equiv A') \;\equiv\; \Box(A \supset A') \land \Box(A' \supset A)$	**T29**
2	$\vdash$	$\Box(A \supset A') \;\supset\; (A ^\frown B) \supset (A' ^\frown B)$	**T1**
3	$\vdash$	$\Box(A' \supset A) \;\supset\; (A' ^\frown B) \supset (A ^\frown B)$	**T1**
4	$\vdash$	$\Box(A \equiv A') \;\supset\; (A ^\frown B) \equiv (A' ^\frown B)$	1–3,**Prop**

T31 $\quad\vdash\quad \Box(A \equiv B) \;\supset\; \Diamond A \equiv \Diamond B$

1	$\vdash$	$\Box(A \equiv B) \;\supset\; (A ^\frown true) \equiv (B ^\frown true)$	**T30**
2	$\vdash$	$\Box(A \equiv B) \;\supset\; \Diamond A \equiv \Diamond B$	1,def. of $\Diamond$

DR32 $\quad\vdash finite \supset (A \equiv B) \;\;\Rightarrow\;\; \vdash \Diamond A \equiv \Diamond B$

1	$\vdash$	$finite \;\supset\; (A \equiv B)$	assump.
2	$\vdash$	$\Box(A \equiv B)$	1,$\Box$**FGen**
3	$\vdash$	$\Box(A \equiv B) \;\supset\; \Diamond A \equiv \Diamond B$	**T31**
4	$\vdash$	$\Diamond A \;\equiv\; \Diamond B$	2,3,**MP**

T33 $\quad\vdash\quad \Box A \;\supset\; \Diamond A$

1	$\vdash$	$A \;\supset\; (empty \supset A)$	**Prop**
2	$\vdash$	$\Box A \;\supset\; \Box(empty \supset A)$	1,**DR26**
3	$\vdash$	$\Box(empty \supset A) \;\supset\; (\Diamond empty \supset \Diamond A)$	**T23**
4	$\vdash$	$\Box A \;\supset\; (\Diamond empty \supset \Diamond A)$	2,3,$\supset$-**chain**
5	$\vdash$	$\Diamond empty$	**T20**
6	$\vdash$	$\Box A \;\supset\; \Diamond A$	4,5,**Prop**

T34 $\quad\vdash\quad \Diamond(A \lor B) \;\equiv\; \Diamond A \lor \Diamond B$

1	$\vdash$	$(A \lor B) ^\frown true \;\equiv\; (A ^\frown true) \lor (B ^\frown true)$	**T18**
2	$\vdash$	$\Diamond(A \lor B) \;\equiv\; \Diamond A \lor \Diamond B$	1,def. of $\Diamond$

T35 $\quad\vdash\quad \Box A \land (A' ^\frown B) \;\supset\; (A \land A') ^\frown B$

1	$\vdash$	$A \;\supset\; (A' \supset A \land A')$	**Prop**
2	$\vdash$	$\Box A \;\supset\; \Box(A' \supset A \land A')$	1,**DR26**

3 ⊢ $\boxdot(A' \supset A \wedge A') \quad \supset \quad (A'^\frown B) \supset (A \wedge A')^\frown B$ **T1**
4 ⊢ $\boxdot A \wedge (A'^\frown B) \quad \supset \quad (A \wedge A')^\frown B$ 2,3,**Prop**

A.3. **Some Properties of Chop, $\diamondsuit$ and $\boxdot$ with State Formulas.**

T36 ⊢ $\diamondsuit w \quad \equiv \quad w$
PROOF FOR $\supset$.
1 ⊢ $\neg w \quad \supset \quad \boxdot \neg w$ **P7**
2 ⊢ $\neg w \quad \supset \quad \neg \diamondsuit \neg \neg w$ 1,def. of $\boxdot$
3 ⊢ $\diamondsuit \neg \neg w \quad \supset \quad w$ 2,**Prop**
4 ⊢ $w \quad \supset \quad \neg \neg w$ **Prop**
5 ⊢ $\diamondsuit w \quad \supset \quad \diamondsuit \neg \neg w$ 4,**DR7**
6 ⊢ $\diamondsuit w \quad \supset \quad w$ 3,5,$\supset$-**chain**
PROOF FOR $\subset$.
1 ⊢ $w \quad \supset \quad \boxdot w$ **P7**
2 ⊢ $\boxdot w \quad \supset \quad \diamondsuit w$ **T33**
3 ⊢ $w \quad \supset \quad \diamondsuit w$ 1,2,$\supset$-**chain**

T37 ⊢ $\boxdot w \quad \equiv \quad w$
1 ⊢ $\diamondsuit \neg w \quad \equiv \quad \neg w$ **T36**
2 ⊢ $\neg \diamondsuit \neg w \quad \equiv \quad w$ 1,**Prop**
3 ⊢ $\boxdot w \quad \equiv \quad w$ 2,def. of $\boxdot$

T38 ⊢ $w^\frown A \quad \supset \quad w$
1 ⊢ $w^\frown A \quad \supset \quad \diamondsuit w$ **T19**
2 ⊢ $\diamondsuit w \quad \equiv \quad w$ **T36**
3 ⊢ $w^\frown A \quad \supset \quad w$ 1,2,**Prop**

T39 ⊢ $(w \wedge A)^\frown B \quad \supset \quad w$
1 ⊢ $w \wedge A \quad \supset \quad w$ **Prop**
2 ⊢ $(w \wedge A)^\frown B \quad \supset \quad w^\frown B$ 1,**DR5**
3 ⊢ $w^\frown B \quad \supset \quad w$ **T38**
4 ⊢ $(w \wedge A)^\frown B \quad \supset \quad w$ 2,3,$\supset$-**chain**

The following lets us move a state formula into the left side of chop:

T40 ⊢ $w \wedge (A^\frown B) \quad \supset \quad (w \wedge A)^\frown B$
1 ⊢ $w \quad \supset \quad \boxdot w$ **P7**
2 ⊢ $w \wedge (A^\frown B) \quad \supset \quad \boxdot w \wedge (A^\frown B)$ 1,**Prop**
3 ⊢ $\boxdot w \wedge (A^\frown B) \quad \supset \quad (w \wedge A)^\frown B$ **T35**
4 ⊢ $w \wedge (A^\frown B) \quad \supset \quad (w \wedge A)^\frown B$ 2,3,$\supset$-**chain**

We can easily combine this with theorem **T39** to deduce the equivalence below:

T41 ⊢ $(w \wedge A)^\frown B \quad \equiv \quad w \wedge (A^\frown B)$
1 ⊢ $(w \wedge A)^\frown B \quad \supset \quad w$ **T39**
2 ⊢ $(w \wedge A)^\frown B \quad \supset \quad (w^\frown B) \wedge (A^\frown B)$ **T16**
3 ⊢ $(w \wedge A)^\frown B \quad \supset \quad w \wedge (A^\frown B)$ 1,2,**Prop**
4 ⊢ $w \wedge (A^\frown B) \quad \supset \quad (w \wedge A)^\frown B$ **T40**

5   ⊢ $w \wedge (A^\frown B)$ ≡ $(w \wedge A)^\frown B$ 3,4,**Prop**

Below is a useful corollary of **T41** used in decomposing the left side of chop:

T42   ⊢ $(w \wedge empty)^\frown A$ ≡ $w \wedge A$
1   ⊢ $(w \wedge empty)^\frown A$ ≡ $w \wedge (empty^\frown A)$ **T41**
2   ⊢ $empty^\frown A$ ≡ A **P5**
3   ⊢ $(w \wedge empty)^\frown A$ ≡ $w \wedge A$ 1,2,**Prop**

A.4. Some Properties of □ involving the Modal System K4.

We now consider how to establish for the PITL operator □ the axiom "4" (PITL Theorem **T47**) found in the modal systems $K4$ and $S4$.

T43   ⊢ $\Diamond \Diamond A$ ≡ $\Diamond A$
1   ⊢ $(A^\frown true)^\frown true$ ≡ $A^\frown(true^\frown true)$ **P2**
2   ⊢ $\Diamond\, true$ ≡ $true$ **T36**
3   ⊢ $(true^\frown true)$ ≡ $true$ 2,def. of ◇
4   ⊢ $A^\frown(true^\frown true)$ ≡ $A^\frown true$ 3,**DR6**
5   ⊢ $(A^\frown true)^\frown true$ ≡ $A^\frown true$ 1,4,≡-**chain**
6   ⊢ $\Diamond \Diamond A$ ≡ $\Diamond A$ 5,def. of ◇

T44   ⊢ $\Diamond \neg A$ ≡ $\neg □ A$
1   ⊢ $□ A$ ≡ $\neg \Diamond \neg A$ def. of □
2   ⊢ $\Diamond \neg A$ ≡ $\neg □ A$ 1,**Prop**

T45   ⊢ $\Diamond \Diamond \neg A$ ≡ $\neg □ □ A$
1   ⊢ $\Diamond \neg A$ ≡ $\neg □ A$ **T44**
2   ⊢ $\Diamond \Diamond \neg A$ ≡ $\Diamond \neg □ A$ 1,**DR8**
3   ⊢ $\Diamond \neg □ A$ ≡ $\neg □ □ A$ **T44**
4   ⊢ $\Diamond \Diamond \neg A$ ≡ $\neg □ □ A$ 2,3,≡-**chain**

T46   ⊢ $□ □ A$ ≡ $□ A$
1   ⊢ $\Diamond \Diamond \neg A$ ≡ $\Diamond \neg A$ **T43**
2   ⊢ $\Diamond \Diamond \neg A$ ≡ $\neg □ □ A$ **T45**
3   ⊢ $\neg □ □ A$ ≡ $\Diamond \neg A$ 1,2,**Prop**
4   ⊢ $\Diamond \neg A$ ≡ $\neg □ A$ **T44**
5   ⊢ $\neg □ □ A$ ≡ $\neg □ A$ 3,4,≡-**chain**
6   ⊢ $□ □ A$ ≡ $□ A$ 5,**Prop**

T47   ⊢ $□ A$ ⊃ $□ □ A$
1   ⊢ $□ □ A$ ≡ $□ A$ **T46**
2   ⊢ $□ A$ ⊃ $□ □ A$ 1,**Prop**

A.5. Properties Involving the PTL Operator ○.

T48   ⊢ $(○ A)^\frown B$ ≡ $○(A^\frown B)$
1   ⊢ $(skip^\frown A)^\frown B$ ≡ $skip^\frown(A^\frown B)$ **P2**
2   ⊢ $(○ A)^\frown B$ ≡ $○(A^\frown B)$ 1,def. of ○

T49 $\vdash$ $(w \wedge \bigcirc A)^\frown B$ $\equiv$ $w \wedge \bigcirc(A^\frown B)$
1 $\vdash$ $(w \wedge \bigcirc A)^\frown B$ $\equiv$ $w \wedge \big((\bigcirc A)^\frown B\big)$ **T41**
2 $\vdash$ $(\bigcirc A)^\frown B$ $\equiv$ $\bigcirc(A^\frown B)$ **T48**
3 $\vdash$ $(w \wedge \bigcirc A)^\frown B$ $\equiv$ $w \wedge \bigcirc(A^\frown B)$ 1,2,**Prop**

T50 $\vdash$ $\diamondsuit\!\!\!\!\diamond\,(w \wedge \bigcirc w')$ $\equiv$ $w \wedge \bigcirc w'$
1 $\vdash$ $(w \wedge \bigcirc w')^\frown true$ $\equiv$ $w \wedge \bigcirc(w'^\frown true)$ **T49**
2 $\vdash$ $\diamondsuit\!\!\!\!\diamond\,(w \wedge \bigcirc w')$ $\equiv$ $w \wedge \bigcirc \diamondsuit\!\!\!\!\diamond\, w'$ 1,def. of $\diamondsuit\!\!\!\!\diamond\,$
3 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, w'$ $\equiv$ w' **T36**
4 $\vdash$ $skip^\frown \diamondsuit\!\!\!\!\diamond\, w'$ $\equiv$ $skip^\frown w'$ 3,**DR10**
5 $\vdash$ $\bigcirc \diamondsuit\!\!\!\!\diamond\, w'$ $\equiv$ $\bigcirc w'$ 4,def. of $\bigcirc$
6 $\vdash$ $\diamondsuit\!\!\!\!\diamond\,(w \wedge \bigcirc w')$ $\equiv$ $w \wedge \bigcirc w'$ 2,5,**Prop**

A.6. **Some Properties of $\boxdot$ Together with $\square$.**

We make use of the following analogue of Theorem **T44** for $\diamondsuit$ and $\square$:

T51 $\vdash$ $\diamondsuit \neg A$ $\equiv$ $\neg \square A$
1 $\vdash$ $\diamondsuit \neg A$ $\equiv$ $\neg \square A$ **VPTL**

T52 $\vdash$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, A$ $\equiv$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, A$
1 $\vdash$ $(true^\frown A)^\frown true$ $\equiv$ $true^\frown(A^\frown true)$ **P2**
2 $\vdash$ $(\diamondsuit A)^\frown true$ $\equiv$ $\diamondsuit(A^\frown true)$ 1,def. of $\diamondsuit$
3 $\vdash$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, A$ $\equiv$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, A$ 2,def. of $\diamondsuit\!\!\!\!\diamond\,$

T53 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, \diamondsuit \neg A$ $\equiv$ $\neg \boxdot \square A$
1 $\vdash$ $\diamondsuit \neg A$ $\equiv$ $\neg \square A$ **T51**
2 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, \diamondsuit \neg A$ $\equiv$ $\diamondsuit\!\!\!\!\diamond\, \neg \square A$ **1,DR8**
3 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, \neg \square A$ $\equiv$ $\neg \boxdot \square A$ **T44**
4 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, \diamondsuit \neg A$ $\equiv$ $\neg \boxdot \square A$ 2,3,$\equiv$-**chain**

T54 $\vdash$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, \neg A$ $\equiv$ $\neg \square \boxdot A$
1 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, \neg A$ $\equiv$ $\neg \boxdot A$ **T44**
2 $\vdash$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, \neg A$ $\equiv$ $\diamondsuit \neg \boxdot A$ **1,DR11**
3 $\vdash$ $\diamondsuit \neg \boxdot A$ $\equiv$ $\neg \square \boxdot A$ **T51**
4 $\vdash$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, \neg A$ $\equiv$ $\neg \square \boxdot A$ 2,3,$\equiv$-**chain**

T55 $\vdash$ $\boxdot \square A$ $\equiv$ $\square \boxdot A$
1 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, \diamondsuit \neg A$ $\equiv$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, \neg A$ **T52**
2 $\vdash$ $\diamondsuit\!\!\!\!\diamond\, \diamondsuit \neg A$ $\equiv$ $\neg \boxdot \square A$ **T53**
3 $\vdash$ $\diamondsuit \diamondsuit\!\!\!\!\diamond\, \neg A$ $\equiv$ $\neg \square \boxdot A$ **T54**
4 $\vdash$ $\boxdot \square A$ $\equiv$ $\square \boxdot A$ 1-3,**Prop**

A.7. **Some Properties of Chop-Star.**

We now consider some theorems and derived rules concerning chop-star.

DR56 $\vdash A \supset more$ $\Rightarrow$ $\vdash A^\star \equiv empty \vee (A^\frown A^\star)$
1 $\vdash$ A $\supset$ $more$ assump.

2	$\vdash$	$A \wedge more \;\equiv\; A$	1,**Prop**
3	$\vdash$	$(A \wedge more)^\frown A^\star \;\equiv\; A^\frown A^\star$	2,**DR6**
4	$\vdash$	$A^\star \;\equiv\; empty \vee \big((A \wedge more)^\frown A^\star\big)$	**P9**
5	$\vdash$	$A^\star \;\equiv\; empty \vee (A^\frown A^\star)$	3,4,**Prop**

DR57 $\vdash A \supset more \;\;\Rightarrow\;\; \vdash A^{\star\frown}B \equiv B \vee \big(A^\frown (A^{\star\frown}B)\big)$

1	$\vdash$	$A \;\supset\; more$	assump.
2	$\vdash$	$A^\star \;\equiv\; empty \vee (A^\frown A^\star)$	1,**DR56**
3	$\vdash$	$A^{\star\frown}B \;\equiv\; (empty \vee (A^\frown A^\star))^\frown B$	2,**DR6**
4	$\vdash$	$(empty \vee (A^\frown A^\star))^\frown B \;\equiv\; (empty^\frown B) \vee ((A^\frown A^\star)^\frown B)$	**T18**
5	$\vdash$	$empty^\frown B \;\equiv\; B$	**P5**
6	$\vdash$	$(A^\frown A^\star)^\frown B \;\equiv\; A^\frown (A^{\star\frown}B)$	**P2**
7	$\vdash$	$A^{\star\frown}B \;\equiv\; B \vee \big(A^\frown (A^{\star\frown}B)\big)$	3–6,**Prop**

T58 $\vdash A^\star \;\equiv\; (A^{\star\frown}empty) \vee A^\omega$

1	$\vdash$	$finite \vee \neg finite$	**Prop**
2	$\vdash$	$finite \vee inf$	1,def. of inf
3	$\vdash$	$finite \;\supset\; (A^{\star\frown}empty) \equiv A^\star$	**P6**
4	$\vdash$	$inf \;\supset\; A^\star \equiv (A^\star \wedge inf)$	**Prop**
5	$\vdash$	$inf \;\supset\; A^\star \equiv A^\omega$	4,def. of chop-omega
6	$\vdash$	$A^\star \;\equiv\; (A^{\star\frown}empty) \vee A^\omega$	2,3,5,**Prop**

A.8. Some Properties Involving a Reduction to PITL with Finite Time.

We now present some derived inference rules which come in useful when completeness for PITL with finite time is assumed (see Theorem 2.2). Recall that any valid implication of the form $finite \supset A$ is allowed and that we designate such a step by using **PITLF**. PITL Theorem **T61** below illustrates this technique.

DR59 $\vdash finite \supset (A \supset B) \;\;\Rightarrow\;\; \vdash \boxdot A \supset \boxdot B$

1	$\vdash$	$finite \;\supset\; (A \supset B)$	assump.
2	$\vdash$	$\boxdot(A \supset B)$	1,$\boxdot$**FGen**
3	$\vdash$	$\boxdot(A \supset B) \;\supset\; (\boxdot A \supset \boxdot B)$	**T25**
4	$\vdash$	$\boxdot A \;\supset\; \boxdot B$	2,3,**MP**

DR60 $\vdash finite \supset (A \equiv B) \;\;\Rightarrow\;\; \vdash \boxdot A \equiv \boxdot B$

1	$\vdash$	$finite \;\supset\; (A \equiv B)$	assump.
2	$\vdash$	$finite \;\supset\; (A \supset B)$	1,**Prop**
3	$\vdash$	$\boxdot A \;\supset\; \boxdot B$	2,**DR59**
4	$\vdash$	$finite \;\supset\; (B \supset A)$	1,**Prop**
5	$\vdash$	$\boxdot B \;\supset\; \boxdot A$	4,**DR59**
6	$\vdash$	$\boxdot A \;\equiv\; \boxdot B$	3,5,**Prop**

The next theorem's proof involves the application of the previous derived inference rule together with completeness for PITL with just finite time:

T61 $\vdash \;\boxdot fin\, w \;\equiv\; \Box\, w$

1	$\vdash$	$\boxdot \boxdot fin\, w \;\equiv\; \boxdot fin\, w$	**T46**
2	$\vdash$	$\boxdot fin\, w \;\equiv\; \boxdot \boxdot fin\, w$	1,**Prop**

3	$\vdash$	$\textit{finite} \ \supset \ \big((\boxdot \textit{fin } w) \equiv \Box\, w\big)$	**PITLF**
4	$\vdash$	$\boxdot \boxdot \textit{fin } w \ \equiv \ \boxdot \Box\, w$	3,**DR60**
5	$\vdash$	$\boxdot \Box\, w \ \equiv \ \Box \boxdot w$	**T55**
6	$\vdash$	$\boxdot w \ \equiv \ w$	**T37**
7	$\vdash$	$\Box \boxdot w \ \equiv \ \Box\, w$	6,**DR12**
8	$\vdash$	$\boxdot \textit{fin } w \ \equiv \ \Box\, w$	2,4,5,7,$\equiv$-**chain**

An alternative proof of Theorem **T61** can be given without **PITLF** by first deducing the dual equivalence $\big(\lozenge\,\diamond(\textit{empty} \wedge w)\big) \equiv \diamond\, w$, for any state formula w.

A.9. **Some Properties of Skip, Next And Until.** Recall from §5.1 that NL^1 formulas are exactly those PTL formulas in which the only temporal operators are unnested $\bigcirc$s (e.g., $p \vee \bigcirc \neg p$ but not $p \vee \bigcirc \bigcirc \neg p$). The next theorem holds for any NL^1 formula T:

T62 $\vdash$ $\lozenge(\textit{more} \wedge T) \ \equiv \ \textit{more} \wedge T$

Proof.. We use Axiom **VPTL** to re-express $\textit{more} \wedge T$ as a logically equivalent disjunction $\bigvee_{1 \le i \le n}(w_i \wedge \bigcirc w_i')$ for some natural number $n \ge 1$ and n pairs of state formulas w_i and w_i':

$$\vdash \quad \textit{more} \wedge T \quad \equiv \quad \bigvee_{1 \le i \le n} (w_i \wedge \bigcirc w_i'). \tag{A.2}$$

Now by Theorem **T50** any conjunction $w \wedge \bigcirc w'$ is deducibly equivalent to $\lozenge(w \wedge \bigcirc w')$. Therefore the disjunction in (A.2) can be re-expressed as $\bigvee_{1 \le i \le n} \lozenge(w_i \wedge \bigcirc w_i')$:

$$\vdash \quad \bigvee_{1 \le i \le n} (w_i \wedge \bigcirc w_i') \quad \equiv \quad \bigvee_{1 \le i \le n} \lozenge(w_i \wedge \bigcirc w_i'). \tag{A.3}$$

Then by $n - 1$ applications of Theorem **T34** and some simple propositional reasoning, the righthand operand of this equivalence is itself is deducibly equivalent to $\lozenge\big(\bigvee_{1 \le i \le n}(w_i \wedge \bigcirc w_i')\big)$:

$$\vdash \quad \bigvee_{1 \le i \le n} \lozenge(w_i \wedge \bigcirc w_i') \quad \equiv \quad \lozenge\big(\bigvee_{1 \le i \le n} (w_i \wedge \bigcirc w_i')\big). \tag{A.4}$$

The chain of the three equivalences (A.2)–(A.4) yields the following:

$$\vdash \quad \textit{more} \wedge T \quad \equiv \quad \lozenge\big(\bigvee_{1 \le i \le n} (w_i \wedge \bigcirc w_i')\big).$$

We then apply Derived Rule **DR8** to the first equivalence (A.2):

$$\vdash \quad \lozenge(\textit{more} \wedge T) \quad \equiv \quad \lozenge\big(\bigvee_{1 \le i \le n} (w_i \wedge \bigcirc w_i')\big).$$

The last two equivalences with simple propositional reasoning yield our goal **T62**. $\square$

Here is a corollary of the previous PITL Theorem **T62** for any NL^1 formula T:

T63 $\vdash$ $\Box(more \supset T)$ $\equiv$ $more \supset T$

1	$\vdash$	$\Box(more \supset T)$	$\equiv$	$\neg\Diamond\neg(more \supset T)$	def. of $\Box$
2	$\vdash$	$\neg(more \supset T)$	$\equiv$	$more \wedge \neg T$	**Prop**
3	$\vdash$	$\Diamond\neg(more \supset T)$	$\equiv$	$\Diamond(more \wedge \neg T)$	2,**DR8**
4	$\vdash$	$\Diamond(more \wedge \neg T)$	$\equiv$	$more \wedge \neg T$	**T62**
5	$\vdash$	$\Diamond\neg(more \supset T)$	$\equiv$	$more \wedge \neg T$	3,4,$\equiv$-**chain**
6	$\vdash$	$\Box(more \supset T)$	$\equiv$	$\neg(more \wedge \neg T)$	1,5,**Prop**
7	$\vdash$	$\neg(more \wedge \neg T)$	$\equiv$	$more \supset T$	**Prop**
8	$\vdash$	$\Box(more \supset T)$	$\equiv$	$more \supset T$	6,7,$\equiv$-**chain**

T64 $\vdash$ $more \wedge T$ $\supset$ $\Box(more \supset T)$

1	$\vdash$	$\Box(more \supset T)$	$\equiv$	$more \supset T$	**T63**
2	$\vdash$	$more \wedge T$	$\supset$	$\Box(more \supset T)$	1,**Prop**

T65 $\vdash$ $\Box(skip \supset A) \wedge \bigcirc B$ $\supset$ $(skip \wedge A)^\frown B$

1	$\vdash$	$\Box(skip \supset A) \wedge (skip^\frown B)$	$\supset$	$\big((skip \supset A) \wedge skip\big)^\frown B$	**T35**
2	$\vdash$	$(skip \supset A) \wedge skip$	$\supset$	$skip \wedge A$	**Prop**
3	$\vdash$	$\big((skip \supset A) \wedge skip\big)^\frown B$	$\supset$	$(skip \wedge A)^\frown B$	2,**DR5**
4	$\vdash$	$\Box(skip \supset A) \wedge (skip^\frown B)$	$\supset$	$(skip \wedge A)^\frown B$	1,3,**Prop**
5	$\vdash$	$\Box(skip \supset A) \wedge \bigcirc B$	$\supset$	$(skip \wedge A)^\frown B$	4,def. of $\bigcirc$

T66 $\vdash$ $\Box(more \supset A)$ $\supset$ $\Box(skip \supset A)$

1	$\vdash$	$more$	$\supset$	$skip$	**VPTL**
2	$\vdash$	$(more \supset A)$	$\supset$	$(skip \supset A)$	1,**Prop**
3	$\vdash$	$\Box(more \supset A)$	$\supset$	$\Box(skip \supset A)$	2,**DR26**

T67 $\vdash$ $\Box(more \supset A) \wedge \bigcirc B$ $\supset$ $(skip \wedge A)^\frown B$

1	$\vdash$	$\Box(more \supset A)$	$\supset$	$\Box(skip \supset A)$	**T66**
2	$\vdash$	$\Box(skip \supset A) \wedge \bigcirc B$	$\supset$	$(skip \wedge A)^\frown B$	**T65**
3	$\vdash$	$\Box(more \supset A) \wedge \bigcirc B$	$\supset$	$(skip \wedge A)^\frown B$	1,2,**Prop**

T68 $\vdash$ $\Diamond(skip \wedge T)$ $\equiv$ $more \wedge T$

1	$\vdash$	$finite$	$\supset$	$\Diamond(skip \wedge T) \equiv (more \wedge T)$	**PITLF**
2	$\vdash$	$\Diamond\Diamond(skip \wedge T)$	$\equiv$	$\Diamond(more \wedge T)$	1,**DR32**
3	$\vdash$	$\Diamond\Diamond(skip \wedge T)$	$\equiv$	$\Diamond(skip \wedge T)$	**T43**
4	$\vdash$	$\Diamond(more \wedge T)$	$\equiv$	$more \wedge T$	**T62**
5	$\vdash$	$\Diamond(skip \wedge T)$	$\equiv$	$more \wedge T$	2–4,**Prop**

T69 $\vdash$ $(skip \wedge T)^\frown A$ $\equiv$ $T \wedge \bigcirc A$

PROOF FOR $\supset$.

1	$\vdash$	$(skip \wedge T)^\frown A$	$\supset$	$\Diamond(skip \wedge T)$	**T19**
2	$\vdash$	$\Diamond(skip \wedge T)$	$\equiv$	$more \wedge T$	**T68**
3	$\vdash$	$(skip \wedge T)^\frown A$	$\supset$	T	1,2,**Prop**
4	$\vdash$	$(skip \wedge T)^\frown A$	$\supset$	$skip^\frown A$	**T14**
5	$\vdash$	$(skip \wedge T)^\frown A$	$\supset$	$\bigcirc A$	4,def. of $\bigcirc$
6	$\vdash$	$(skip \wedge T)^\frown A$	$\supset$	$T \wedge \bigcirc A$	3,5,**Prop**

PROOF FOR $\subset$.

1	$\vdash$	$\bigcirc A$	$\supset$	$more$	**VPTL**

2	$\vdash$	$more \ \wedge \ T \quad \supset \quad \boxdot(more \supset T)$	**T64**
3	$\vdash$	$T \ \wedge \ \bigcirc A \quad \supset \quad \boxdot(more \supset T)$	1,2,**Prop**
4	$\vdash$	$\boxdot(more \supset T) \ \wedge \ \bigcirc A \quad \supset \quad (skip \wedge T)^\frown A$	**T67**
5	$\vdash$	$T \ \wedge \ \bigcirc A \quad \supset \quad (skip \wedge T)^\frown A$	3,4,**Prop**

T70	$\vdash$	$T \ until \ A \quad \equiv \quad A \vee \big(T \wedge \bigcirc(T \ until \ A)\big)$	
1	$\vdash$	$skip \wedge T \quad \supset \quad more$	**VPTL**
2	$\vdash$	$(skip \wedge T)^{\star\frown} A \quad \equiv \quad A \vee \big((skip \wedge T)^\frown((skip \wedge T)^{\star\frown}A)\big)$	1,**DR57**
3	$\vdash$	$T \ until \ A \quad \equiv \quad A \vee \big((skip \wedge T)^\frown(T \ until \ A)\big)$	2,def. of $until$
4	$\vdash$	$(skip \wedge T)^\frown(T \ until \ A) \quad \equiv \quad T \ \wedge \ \bigcirc(T \ until \ A)$	**T69**
5	$\vdash$	$T \ until \ A \quad \equiv \quad A \vee \big(T \wedge \bigcirc(T \ until \ A)\big)$	3–4,**Prop**

T71	$\vdash$	$T \ until \ A \quad \supset \quad \diamondsuit A$	
1	$\vdash$	$(skip \wedge T)^{\star\frown} A \quad \supset \quad \diamondsuit A$	**T21**
2	$\vdash$	$T \ until \ A \quad \supset \quad \diamondsuit A$	1,def. of $until$