This is a pre-copy-editing, author-produced PDF of an article accepted for publication in the *Journal of Logic and Computation* following peer review. The definitive publisher-authenticated version (*Journal of Logic and Computation 2007 17(2):333-409;* doi:10.1093/logcom/exm006) is available online at: http://logcom.oxfordjournals.org/cgi/content/abstract/17/2/333

Using Temporal Logic to Analyse Temporal Logic: A Hierarchical Approach Based on Intervals

Ben Moszkowski*
Software Technology Research Laboratory; Gateway House
De Montfort University; The Gateway; Leicester LE1 9BH; Great Britain
email: benm@dmu.ac.uk

9 January 2007

Abstract

Temporal logic has been extensively utilized in academia and industry to formally specify and verify behavioural properties of numerous kinds of hardware and software. We present a novel way to apply temporal logic to the study of a version of itself, namely, propositional linear-time temporal logic (PTL). This involves a hierarchical framework for obtaining standard results for PTL, including a small model property, decision procedures and axiomatic completeness. A large number of the steps involved are expressed in a propositional version of Interval Temporal Logic (ITL) which is referred to as PITL. It is a natural generalization of PTL and includes operators for reasoning about periods of time and sequential composition. Versions of PTL with finite time and infinite time are both considered and one benefit of the framework is the ability to systematically reduce infinite-time reasoning to finite-time reasoning. The treatment of PTL with the operator $until$ and past time naturally reduces to that for PTL without either one. The interval-oriented methodology differs from other analyses of PTL which typically use sets of formulas and sequences of such sets for canonical models. Instead we represent models as time intervals expressible in PITL. The analysis furthermore relates larger intervals with smaller ones. Being an interval-based formalism, PITL is well suited for sequentially combining and decomposing the relevant formulas. Consequently, we can articulate issues of equal significance in more conventional analyses of PTL but normally only considered at the metalevel. A good example of this is the existence of bounded models with periodic suffixes for PTL formulas which are satisfiable in infinite time. We also describe decision procedures based on binary decision diagrams and exploit some links with finite-state automata.

Beyond the specific issues involving PTL, the research is a significant application of ITL and interval-based reasoning and illustrates a general approach to formally reasoning about sequential and parallel behaviour in discrete linear time. The work also includes some interesting representation theorems. In addition, it has relevance to hardware description and verification since the specification languages PSL/Sugar (IEEE Standard 1850) and "temporal e" (part of IEEE Standard 1647) both contain temporal constructs concerning intervals of time as does

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

the related SystemVerilog Assertion language contained in SystemVerilog (IEEE Standard 1800), an extension of the IEEE 1364-2001 Verilog language.

Keywords: temporal logic, interval temporal logic, small models, decision procedures, axiomatic completeness

1 Introduction

Temporal logic as studied by Prior [80], Rescher and Urquhart [83] and others has its historical roots in philosophy. However, following the seminal paper by Pnueli [78], it has become one of the main formalisms used in computer science for reasoning about the dynamic behaviour of systems [25, 34, 35, 53, 59]. In particular, the version known as *Propositional Linear-Time Temporal Logic* (abbreviated as either PTL or PLTL) and some variants of it have been extensively investigated and applied. In a relatively recent and significant article, Lichtenstein and Pnueli [57] give a detailed analysis of PTL which is meant to largely subsume and supercede earlier ones. Indeed, the work appears to have the rather ambitious goal of coming close to offering the last word on the subject and is perhaps best described in the authors' own words:

> The paper summarizes work of over 20 years and is intended to provide a definitive reference to the version of propositional temporal logic used for the specification and verification of reactive systems.

The kind of PTL considered by Lichtenstein and Pnueli has discrete time and past time. Both decision procedures and axiomatic completeness are discussed and a new simplified axiom system is presented. The approach makes use of semantic tableaux and throughout the presentation the treatment of PTL with past-time operators runs in parallel with the future-only version. The authors choose in particular to use tableaux since they offer a basis for uniformly showing axiomatic completeness and also obtaining a practical decision procedure. The material about past time is distinctly marked so that one can optionally delete it to obtain an analysis limited to the future fragment of PTL.

We present a novel framework for investigating PTL which significantly differs from the methods of Lichtenstein and Pnueli and earlier treatments such as [33, 36, 53, 95]. It is used to obtain standard results such as a small model property, decision procedures and axiomatic completeness. However, instead of relying on semantic tableaux, filtration and other previous techniques, our method is based on an interval-oriented analysis of certain kinds of low-level PTL formulas called *transition configurations*. An important feature of this approach is that it provides a natural hierarchical means of reducing full PTL to this subset and also reduces both PTL with the $until$ operator and past time to versions without them. Therefore the overwhelming bulk of the analysis only involves PTL with neither $until$ nor past time. Moreover, the analysis of PTL with infinite time naturally reduces to that for PTL with just finite time. The low-level formulas also have associated decision procedures, including simple symbolic ones based on binary decision diagrams (BDDs) [10] which we have implemented. Some connections with automata-based decision procedures for PTL are discussed.

The basic version of PTL used here is described in detail in Section 3 but we will now briefly summarize some of the features in order to be able to overview some key aspects of our work. We postpone the treatment of $until$ and past time in order to later handle them in a natural hierarchical manner. Both finite and infinite time are permitted, whereas most versions of PTL deal solely with the latter. One reason for including

finite time is to allow us to naturally capture parts of our infinite-time analysis within PTL formulas concerning finite-time subintervals. The only two primitive temporal operators initially considered are $\bigcirc$ (strong next) and $\Diamond$ (eventually) although some others are definable in terms of them (e.g., $\Box$ (henceforth) and $\Diamond^+$ (strict eventually)).

Our analysis of PTL extensively employs intervals of time which are represented as finite and countably infinite sequences of states and described by formulas in a propositional version of Interval Temporal Logic (ITL) [37, 64–68, 71–73] (see also [49]) referred to as PITL. By using a hierarchical, interval-oriented framework, the approach differs from that of Lichtenstein and Pnueli and previous ones which in general utilize sets of formulas and sequences of such sets (also referred to as *paths*). We instead relate transition configurations to semantically equivalent formulas in PITL. Time intervals facilitate an analysis which naturally relates larger intervals with smaller ones. The process of doing this can be explicitly expressed in PITL in a way not possible within previous frameworks which lack both a formalization of intervals and logical operators concerning various kinds of sequential composition of intervals.

Let us now informally consider as an example a simplified presentation of how we later establish the existence of periodic models for certain kinds of low-level formulas involving infinite time. The analysis for temporal logic formulas involving infinite time needs to consider formulas of the form $\Box \Diamond^+ A$, where A is itself a restricted kind of temporal logic formula. Here $\Box \Diamond^+ A$ is true for an interval, that is, the interval *satisfies* $\Box \Diamond^+ A$, iff the interval has infinite length and A itself is satisfied by an infinite number of the interval's suffixes. We want to show that if $\Box \Diamond^+ A$ is satisfied by some interval, then there also exists a periodic interval which satisfies $\Box \Diamond^+ A$. We first show a sufficient condition motivated by A's restricted syntax which ensures that $\Box \Diamond^+ A$ is semantically equivalent to the PITL formula A^ω. This formula is true on an interval if the interval has infinite length and can be split into an infinite sequence of finite intervals each satisfying A. We then select one of these finite intervals and join ω copies of it together to obtain a periodic interval satisfying A^ω and hence also the original formula $\Box \Diamond^+ A$. Furthermore, after showing the existence of bounded models for A, we can then establish similar properties for A^ω and hence also $\Box \Diamond^+ A$.

We believe that our interval-based analysis complements existing approaches since it provides a notational way to articulate various issues concerning PTL model construction which are equally relevant within a more conventional analysis but are normally only considered at the metalevel. It also illustrates some general techniques for compositional specification and proof in discrete linear time. This all fits nicely with one of the main purposes of a logic which is to provide a notation for explicitly and formally expressing reasoning processes. In addition, a number of the temporal logic formulas encountered can even be used with little or no change as input to a implementation of a PTL decision procedure which supports both finite and infinite time. The analysis itself is performed without the need to add any fundamentally new concepts to PITL but does require a reader's willingness to acquire some familiarity with PITL and various fairly general issues concerning interval-based reasoning.

Another feature of our approach is that it readily generalizes to a finite-time analysis of an important subset of PITL called *Fusion Logic* (FL), which was previously used by us in [73] to hierarchically show the completeness of an axiom system for PITL with finite time. The analysis of FL uses a reduction of FL formulas to PTL ones. The prototype implementation of our decision procedure for PTL with finite time also supports FL. A brief introduction to FL is given in Subsection 13.4 since FL is a natural extension of our framework for studying PTL and furthermore demonstrates another connection between PTL and intervals.

Our previous work in [74] contains an earlier description of this material but was limited to showing axiomatic completeness for PTL without past time. In the mean time, we have significantly extended the notation, methods and their scope of application. The structure of presentation has also been refined. A preliminary version of the current work appears in [75], after which further improvements have been subsequently incorporated.

The use of intervals here seems to go well with a growing general awareness even in industry of the desirability for temporal logics which go beyond conventional point-based constructs to also handle behavioural specifications involving intervals of time. As evidence for this we mention the Property Specification Language PSL/Sugar [81]. This is a modified version of a language Sugar [3] developed at IBM/Haifa. PSL/Sugar has been ratified as IEEE Standard 1850 by the IEEE Standards Association [47] and has the purpose of precisely expressing a hardware system's design properties so that they can then be tested using simulation and model checking. It includes a temporal logic with regular expressions and other operators for sequential composition. The hardware description language SystemVerilog [85] is an extension of the established IEEE Standard 1364 Verilog language and includes temporal assertions similar to those in PSL/Sugar. SystemVerilog was itself ratified as a standard by Accellera Organization, Inc. and has now been approved as IEEE Standard 1800.

In addition, the IEEE Standards Association has more recently approved Verisity Ltd.'s [90] *e* language, which is intended for functional testing and verification, as IEEE Standard 1647 [48][1]. A subset of *e* called *temporal e* contains temporal operators for sequentially composition and was influenced in part by ITL [45, 63, 91].

Structure of presentation

Let us now summarize the structure of the rest of this paper. Section 2 mentions some related work and compares it with our approach. Section 3 presents the version of PTL we use. Section 4 summarizes the propositional version of ITL which we use in the analysis. Section 5 introduces low level PTL formulas called *transition configurations* and relates them to some semantically equivalent propositional ITL formulas which simplify the subsequent analysis. Section 6 proves the existence of small models for transition configurations. Section 7 shows how to relate the satisfiability of the two main kinds of transition configurations with simple interval-oriented tests. Section 8 deals with BDD-based decision procedures for transition configurations. Section 9 concerns axiomatic completeness for an important subset of PTL in which the only temporal operator is $\bigcirc$ (next). Section 10 looks at a PTL axiom system and axiomatic completeness for transition configurations. Section 11 presents formulas called *invariants* and *invariant configurations* which together serve as a bridge between the previously mentioned transition configurations and arbitrary PTL formulas. Section 12 discusses how to generalize the previous results to work with arbitrary PTL formulas. Section 13 hierarchically extends our approach to deal with both the temporal operator $until$ and past time. It also briefly looks at a superset of PTL called *Fusion Logic*. Section 14 concludes with some brief discussion.

[1] Verisity has been acquired by Cadence Design Systems [16].

2 Background

Temporal logics have become a popular topic of study in theoretical computer science and are also being utilized by industry to locate faults in digital circuit designs, communication protocols and other applications. Issues such as small models, proof systems, axiomatic completeness and decision procedures for PTL (with time almost always modelled as discrete and infinite) have been extensively investigated by Gabbay et al. [33], Sistla and Clarke [84], Wolper [95], Kröger [53], Goldblatt [36], Lichtenstein and Pnueli [57], Lange and Stirling [55], Pucella [82] (who also considers PTL with finite time) and others. French [32] elaborates on the presentation by Gabbay et al. [33].

Vardi and Wolper [88] and Bernholtz, Vardi and Wolper [6] give decisions procedures for some temporal logics based on a reduction to ω-automata. They do not consider axiomatic completeness. Wolper [93] presents a tutorial on such a decision procedure for PTL with infinite time.

Ben-Ari et al. [4, 5], Wolper [92, 94] and Banieqbal and Barringer [2] develop closely related proofs of completeness for logics which include PTL as a subset or are branching-time versions of it. The book by Rescher and Urquhart [83] is an early source of tableau-based completeness proofs for temporal logics with various models of time. The survey by Emerson [25] includes material about axiom systems for both linear and branching-time temporal logic. Burgess [15] and Marx, Mikulas and Reynolds [61] consider the axiomatization of versions of temporal logic with linear time but without an assumption of discrete time. The handbooks by Gabbay, Hodkinson and Reynolds [35] and Gabbay, Finger and Reynolds [34] give extensive coverage to various important aspects of temporal logic such as axiomatization.

Fisher [29, 30] (see also later work by Fisher, Dixon and Peim [31] and Bolotov, Fisher and Dixon [9]) presents a normal form for PTL called *Separated Normal Form* (SNF) which consists of formulas having the syntax $\Box \bigwedge_i A_i$, where each A_i can be one of the following:

$$\mathbf{start} \supset \bigvee_c l_c \qquad \bigcirc \bigwedge_a k_a \supset \bigcirc \bigvee_d l_d \qquad \bigcirc \bigwedge_b k_b \supset \Diamond l.$$

Here each particular k_a, k_b, l, l_c and l_d is a literal (i.e., a propositional variable or its negation). Some versions of SNF permit past-time constructs or have other relatively minor differences. Applications include theorem proving, executable specifications and representing ω-automata. We mention SNF here since it is a PTL normal form which somewhat resembles what we call invariants and formally introduce in Section 11.

3 Overview of PTL

This section summarizes the basic version of PTL used here. Later on in Section 13 we augment PTL with the operator $until$ and past time.

3.1 Syntax of PTL

We now describe the syntax of permitted PTL formulas. In what follows, p is any propositional variable and both X and Y denote PTL formulas:

$$p \qquad true \quad \neg X \quad X \vee Y \quad \bigcirc X \text{ (strong next)} \quad \Diamond X \text{ (eventually)}.$$

We include $true$ as a primitive so as to avoid a definition of it which contains some specific variable. This is not strictly necessary. Other conventional logic operators such as $false$, $X \wedge Y$ and $X \supset Y$ (X implies Y) are defined in the usual way. Also, $\Box X$ (henceforth) is defined as $\neg \Diamond \neg X$.

3.2 Semantics of PTL

The version of PTL considered here uses discrete, linear time which is represented by intervals each consisting of a sequence of one or more states. More precisely, an interval σ is any finite or infinite sequence of one or more states σ^0, σ^1, ... Each state σ^i in σ maps each propositional variable p, q, ...to one of the boolean values $true$ and $false$. The value of p in the state σ^i is denoted $\sigma^i(p)$. A finite interval σ has an *interval length* $|\sigma| \geq 0$ which equals the number of states minus 1 and is hence always greater than or equal to 0. We regard the smallest nonzero interval length 1 as a *unit* of (abstract) time. For example, an interval with 6 states has interval length 5 or equivalently 5 time units. These units do not correspond to any particular notion of physical time. The interval length of an infinite interval is taken to be ω. The term *subinterval* refers to any interval obtained from some *contiguous* subsequence of another interval's states.

We call a one-state interval (i.e., one with interval length 0) an *empty interval*. A two-state interval (i.e., one with interval length 1) is called a *unit interval*. Both kinds of intervals play an important role in our analysis.

The notation $\sigma \models X$ denotes that the interval σ *satisfies* the PTL formula X. We now give a definition of this using induction on X's syntax:

- Propositional variable: $\sigma \models p$ iff p is true in the initial state σ^0 (i.e., $\sigma^0(p) = true$).
- True: $\sigma \models true$ trivially holds for any σ.
- Negation: $\sigma \models \neg X$ iff $\sigma \not\models X$.
- Disjunction: $\sigma \models X \vee Y$ iff $\sigma \models X$ or $\sigma \models Y$.
- Next: $\sigma \models \bigcirc X$ iff $\sigma' \models X$,
  where σ contains at least two states and σ' denotes the suffix subinterval $\sigma^1\sigma^2 \ldots$ which starts from second state σ^1 in σ.
- Eventually: $\sigma \models \Diamond X$ iff $\sigma' \models X$,
  for some suffix subinterval σ' of σ (perhaps σ itself).

Table 1 shows a variety of other useful temporal operators which are definable in PTL. It includes operators for testing whether an interval is finite or infinite and whether the interval has exactly one state or two states. Most of the operators only become relevant when finite intervals are permitted. Therefore, readers who are just familiar with conventional PTL with infinite time will have previously encountered only a few of the operators.

Note: Some readers may prefer to skim Table 1 for now and later consult it in more detail when the various operators are actually used.

Figure 1 assists in the understanding of Table 1 by illustrating a number of the operators through sample formulas and intervals. In the figure, the logical values $true$ and $false$ are respectively abbreviated as "`t`" and "`f`". In what follows, we frequently

Standard derived PTL *operators:*		
$\Box\, X$	$\stackrel{\text{def}}{\equiv} \neg \Diamond \neg X$	Henceforth
$\Diamond^+ X$	$\stackrel{\text{def}}{\equiv} \bigcirc \Diamond X$	Eventually in strict future
$\Box^+ X$	$\stackrel{\text{def}}{\equiv} \neg \Diamond^+ \neg X$	Henceforth in strict future (not used here)
PTL *operators primarily for finite intervals:*		
$more$	$\stackrel{\text{def}}{\equiv} \bigcirc\, true$	More than one state
$empty$	$\stackrel{\text{def}}{\equiv} \neg more$	Only one state (empty interval)
$\circledcirc_{\text{w}}\, X$	$\stackrel{\text{def}}{\equiv} \neg \bigcirc \neg X$	Weak next (same as $more \supset \bigcirc X$)
$skip$	$\stackrel{\text{def}}{\equiv} \bigcirc\, empty$	Exactly two states (unit interval)
$X?$	$\stackrel{\text{def}}{\equiv} X \wedge empty$	Empty interval with test
$\$\, X$	$\stackrel{\text{def}}{\equiv} X \wedge skip$	Unit interval with test
PTL *operators for finite and infinite intervals:*		
$finite$	$\stackrel{\text{def}}{\equiv} \Diamond\, empty$	Finite interval
inf	$\stackrel{\text{def}}{\equiv} \neg finite$	Infinite interval
$sfin\ X$	$\stackrel{\text{def}}{\equiv} \Diamond(empty \wedge X)$	Strong test of final state
$fin\ X$	$\stackrel{\text{def}}{\equiv} \Box(empty \supset X)$	Weak test of final state
$\diamondsuit_{\text{m}}\, X$	$\stackrel{\text{def}}{\equiv} \Diamond(more \wedge X)$	Sometime before the very end
$\boxdot_{\text{m}}\, X$	$\stackrel{\text{def}}{\equiv} \Box(more \supset X)$	Henceforth except perhaps at very end
$X \leftarrow Y$	$\stackrel{\text{def}}{\equiv} finite\ \supset\ (fin\ X) \equiv Y$	Temporal assignment

Table 1: Some definable PTL operators

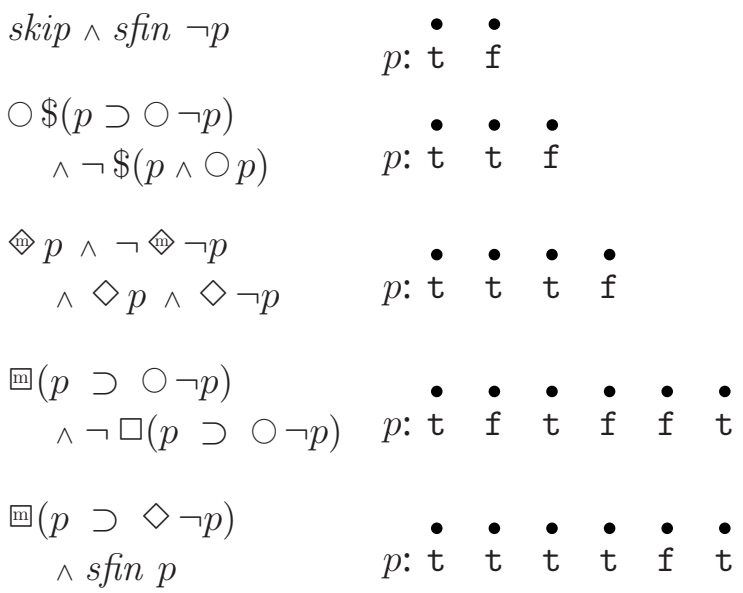


Figure 1: Some examples of formulas with derived PTL operators

use the operator ▣ instead of the more conventional PTL operator $\Box$ when we need to test all pairs of adjacent states in a interval. This is because ▣ is better suited for such tests on finite-time intervals since it does not "run off the end". The fourth example in Figure 1 illustrates this feature. As a consequence, ▣ is often easier to work with in our interval-based analysis as is later shown in Theorem 5.4.

Definition 3.1 (Satisfiability and validity)**.** *For any interval σ and* PTL *formula X, if σ satisfies X (i.e., $\sigma \models X$ holds), then X is said to be* satisfiable*, denoted as $\dashv X$. A formula X satisfied by all intervals is* valid*, denoted as $\models X$.*

We now define an important subset of PTL involving the operator $\bigcirc$:

Definition 3.2 (Next Logic)**.** *The set of* PTL *formulas in which the only primitive temporal operator is $\bigcirc$ is called* Next Logic (NL)*. The subset of* NL *in which no $\bigcirc$ is nested within another $\bigcirc$ is denoted as* NL^1*.*

For example, the NL formula $p \wedge \bigcirc q$ is in NL^1, whereas the NL formula $p \wedge \bigcirc(q \vee \bigcirc p)$ is not.

The variables T, T' and T'' denote formulas in NL^1.

Definition 3.3 (Tautologies)**.** *A* tautology *is any formula which is a substitution instance of some valid nonmodal propositional formula.*

For example, the formula $\bigcirc X \vee \Diamond Y \supset \Diamond Y$ is a tautology since it is a substitution instance of the valid nonmodal formula $p \vee q \supset q$. It is not hard to show that all tautologies are themselves valid since intuitively a tautology is any valid formula which does not require modal reasoning to justify its truth.

Convention for variables denoting individual formulas and sets of formulas: In what follows, the variables w, w' and w'' refer to *state formulas*, that is, formulas with no temporal operators. Furthermore, PROP denotes the set of all state formulas. For any finite set of variables V, PROP$_V$ denotes the set of all state formulas only having

variables in V. Likewise, the set PTL_V denotes the set of all formulas in PTL only containing variables in V and NL^1_V denotes the set of all formulas in NL^1 only having variables in V. For example, the formula $p \wedge \Diamond q$ is in $\text{PTL}_{\{p,q\}}$ but not in $\text{PTL}_{\{p\}}$.

3.3 Example of the hierarchical process

Our analysis of PTL reduces arbitrary PTL formulas to lower level ones with a much more restricted syntax. The next PTL formula serves as a simple example to motivate some of the notation and conventions later introduced:

$$\Box \Diamond p \wedge \Box \Diamond \neg p.$$

This is reducible to the formula $\Box I \wedge w$, where I and w are given below:

$$\begin{aligned} I\colon\ & (r_1 \equiv \Diamond p) \wedge (r_2 \equiv \Diamond \neg r_1) \wedge (r_3 \equiv \Diamond \neg p) \wedge (r_4 \equiv \Diamond \neg r_3) \\ w\colon\ & \neg r_2 \wedge \neg r_4. \end{aligned}$$

The auxiliary variables $r_1, \ldots, r_4$ provide a natural way to restrict the nesting of temporal operators within the conjunction I. We call I an *invariant* and the conjunction $\Box I \wedge w$ an *invariant configuration*. Both are formally introduced later in Section 11. It can be shown that the original formula $\Box \Diamond p \wedge \Box \Diamond \neg p$ is satisfiable iff the invariant configuration $\Box I \wedge w$ is. Similarly, the original formula is satisfiable in finite time iff the invariant configuration $\Box I \wedge w \wedge \mathit{finite}$ is.

When analysing behaviour in finite time, we further transform the invariant configuration $\Box I \wedge w \wedge \mathit{finite}$ to a semantically equivalent formula which is a special kind of conjunction $\Box T \wedge w \wedge \mathit{finite}$, where T and w are as follows:

$$\begin{aligned} T\colon\ & (r_1 \equiv (p \vee \bigcirc r_1)) \wedge (r_2 \equiv (\neg r_1 \vee \bigcirc r_2)) \\ & \wedge (r_3 \equiv (\neg p \vee \bigcirc r_3)) \wedge (r_4 \equiv (\neg r_3 \vee \bigcirc r_4)) \\ w\colon\ & \neg r_2 \wedge \neg r_4. \end{aligned}$$

Here I's first conjunct $r_1 \equiv \Diamond p$ is replaced in T by the $\Diamond$-free formula $r_1 \equiv (p \vee \bigcirc r_1)$. The remaining conjuncts in T similarly avoid having any $\Diamond$ constructs. We call T a *transition formula* and $\Box T \wedge w \wedge \mathit{finite}$ a *transition configuration* (formally defined in Section 5). The formula T is in fact a formula in the important PTL subset called NL^1 (formally defined earlier in Definition 3.2) in which the only temporal constructs are $\bigcirc$ operators not nested within other $\bigcirc$ operators. In addition, in finite-time intervals the PTL formulas $\Box I$ and $\Box T$ are semantically equivalent. Moreover, it can be proved that the original formula $\Box \Diamond p \wedge \Box \Diamond \neg p$ is satisfiable in finite time iff the transition configuration $\Box T \wedge w \wedge \mathit{finite}$ is satisfiable. As is later shown in Section 5, NL^1 formulas such as T play a fundamental role in our analysis of transition configurations.

Sections 6–10 subsequently consider small models, decision procedures and completeness of axiom systems for transition configurations such as $\Box T \wedge w \wedge \mathit{finite}$. For example, in Section 6 in Theorem 6.2, we establish the existence of small models for this kind of transition configuration by showing that it is satisfiable iff it is satisfiable in a finite interval having less than $2^{|V|}$ states, where V is any finite set of variables which includes all variables occurring in T and w. If the formula is indeed satisfiable, the decision procedure in Section 8 can construct an interval of less than $2^{|V|}$ states by first considering all states satisfying w. The algorithm then searches for intervals starting

with such a state and in which each pair of adjacent states, when regarded as a two-state interval, satisfies the transition formula T and additionally the final state, when regarded as a one-state interval, also satisfies T. This is to ensure that the entire interval satisfies the formula $\Box T$. A sample interval can then be generated. On the other hand, if the transition configuration is unsatisfiable and there is sufficient memory available, the decision procedure will eventually terminate with a negative answer.

In Section 10, an axiom system for PTL is given and then a form of axiomatic completeness for the various kinds of transition configurations is established. This is done by proving that any consistent transition configuration, that is, one which is not deducibly false in the axiom system, is satisfiable.

We later in Section 11 formally extend all of this material to handle invariant configurations such as $\Box I \wedge w \wedge \mathit{finite}$ by reducing them to semantically equivalent transition configurations. The results for small models and decision procedures for transition configurations can then be used. Axiomatic completeness for invariant configurations is shown by noting that an invariant configuration such as $\Box I \wedge w \wedge \mathit{finite}$ is deducibly equivalent to its associated transition configuration and then making use of the previously established axiomatic completeness for transition configurations.

Finally in Section 12 we deal with arbitrary formulas such as the sample one $\Box \Diamond p \wedge \Box \Diamond \neg p$ by systematically reducing them to lower-level invariant configurations which capture their semantics and normally contain extra auxiliary variables. The original formula is satisfiable iff the invariant configuration is. Furthermore, any interval satisfying the invariant configuration also satisfies the original formula. The decision procedures for invariant configurations, which, as already mentioned, in fact reduce them to semantically equivalent transition configurations, can then be applied. Axiomatic completeness for arbitrary PTL formulas follows by observing that if such a formula is consistent, then so is the associated invariant configuration. From axiomatic completeness for invariant configurations, it follows that there exists a model for this particular invariant configuration. Finally, this model can also serve as a model for the original PTL formula.

3.4 Notation for accessing parts of conjunctions

From the examples just given it can be seen that we often manipulate formulas which are conjunctions. For the moment we do not make any particular assumptions about the syntax of the individual conjuncts although this will be done in later sections for certain useful kinds of conjunctions. The next three definitions provide some helpful notation for denoting the number of conjuncts of an arbitrary conjunction and for accessing one or more of them. In what follows, C denotes some conjunction of zero or more conjuncts. A conjunction with no conjuncts is denoted as true.

Definition 3.4 (Size of a conjunction)**.** *Let the notation $|C|$ denote the number of C's conjuncts.*

Definition 3.5 (Indexing of a conjunction's conjuncts)**.** *For each $k : 1 \leq k \leq |C|$, we let $C[k]$ denote the k-th conjunct.*

Observe that if a conjunction C has length $|C| = 0$, there are no conjuncts to be indexed.

Definition 3.6 (Parts of a conjunction)**.** *Let k and l be natural numbers such that $1 \leq k \leq |C|$ and $0 \leq l \leq |C|$. The notation $C[k : l]$ denotes the conjunction of*

consecutive conjuncts in C starting with $C[k]$ and finishing with $C[l]$, inclusive, i.e., $C[k] \wedge \cdots \wedge C[l]$ (which contains $l - k + 1$ conjuncts).

For example, below is shown a conjunction with three conjuncts and how to access them:

$$C\colon\ \ (p \supset q)\ \wedge\ \neg(r \equiv \bigcirc \mathit{true})\ \wedge\ (r \vee u)$$
$$|C| = 3 \qquad C[1]\colon\ \ p \supset q \qquad C[2]\colon\ \ \neg(r \equiv \bigcirc \mathit{true}) \qquad C[3]\colon\ \ r \vee u.$$

Note that for any conjunction C, the formula $C[1:0]$ denotes true and $C[1:|C|]$ is identical to C. Also, for any $k : 1 \leq k \leq |C|$, both $C[k]$ and $C[k:k]$ refer to the same conjunct.

Our analysis will frequently make use of conjunctions of equivalences, as illustrated in the previous Subsection 3.3. When doing this, we will sometimes omit the parentheses around each individual equivalence and instead rely on both the context and sufficient spacing between them to avoid ambiguous parsing. Here is an example:

$$r_1 \equiv (r_2 \vee r_3) \quad \wedge \quad r_2 \equiv \bigcirc p \quad \wedge \quad r_3 \equiv \Diamond q.$$

4 Propositional Interval Temporal Logic

We now describe the version of quantifier-free propositional ITL (PITL) used here for systematically analysing transition configurations. More on ITL can be found in [37, 64–68, 71–73] (see also [49]). The same discrete-time intervals are used as in PTL. In addition, all PTL constructs are permitted as well as two other ones. Hence, any PTL formula is also a PITL formula.

Here is the syntax of PITL's two primitive interval-oriented constructs *chop* and *chop-star*, where A and B are themselves PITL formulas:

$$A; B \ \text{(chop)} \qquad A^* \ \text{(chop-star)}.$$

The semantics of the other constructs in PITL is as in PTL and is therefore omitted here.

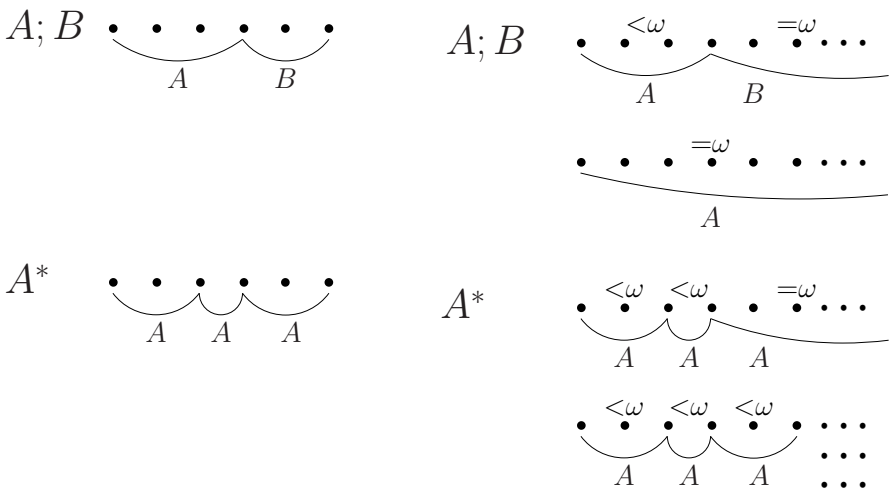
Before defining the semantics of chop and chop-star, we introduce some notation for describing subintervals of an interval σ. For natural numbers i, j with $i \leq j \leq |\sigma|$, let $\sigma^{i:j}$ denotes the subinterval with starting state σ^i and final state σ^j and having interval length $j - i$ (i.e., $j - i + 1$ states). Furthermore, if σ is an infinite interval, let $\sigma^{i:\omega}$ denote the (infinite) suffix subinterval starting with state σ^i.

The formula $A; B$ is true on σ (i.e., $\sigma \models A; B$) iff one of the following holds:

- For some natural number $i : 0 \leq i \leq |\sigma|$, the interval σ can be divided into two subintervals $\sigma^{0:i}$ and $\sigma^{i:|\sigma|}$ sharing the state σ^i such that both $\sigma^{0:i} \models A$ and $\sigma^{i:|\sigma|} \models B$ hold.
- The interval σ itself has infinite length and $\sigma \models A$ holds.

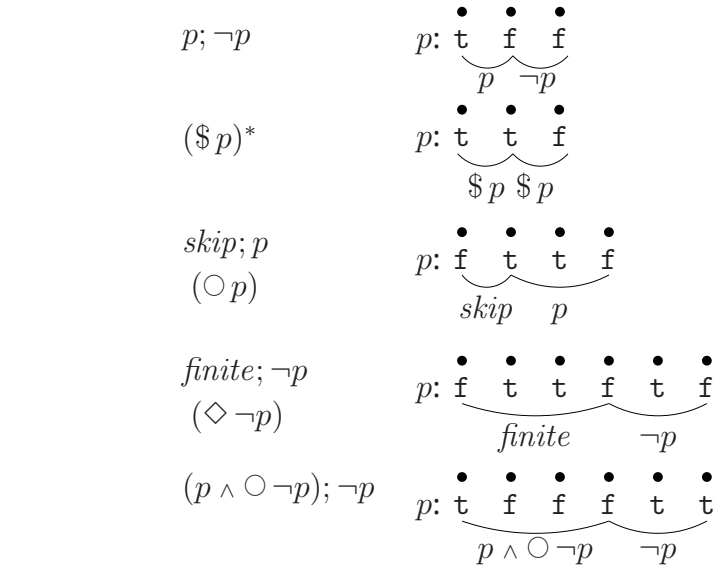
The formula A^* is true on σ (i.e., $\sigma \models A^*$) iff one of the following holds:

- The interval σ has finite length and there exists some natural number $n \geq 0$ and finite sequence of natural numbers $l_0 \leq l_1 \leq \cdots \leq l_n$ where $l_0 = 0$ and $l_n = |\sigma|$, such that for each $i : 0 \leq i < n$, $\sigma^{l_i:l_{i+1}} \models A$ holds.



(a) Informal semantics for finite time

(b) Informal semantics for infinite time



(c) Some finite-time examples

Figure 2: Informal PITL semantics and examples

- The interval σ has infinite length and there exists some $n \geq 0$ and finite sequence of natural numbers $l_0 \leq l_1 \leq \cdots \leq l_n$ where $l_0 = 0$, such that for each $i : 0 \leq i < n$, $\sigma^{l_i:l_{i+1}} \models A$ holds and also $\sigma^{l_n:\omega} \models A$ holds.
- The interval σ has infinite length and there exists some countably infinite strictly ascending sequence of natural numbers $l_0 < l_1 < \cdots$ where $l_0 = 0$, such that for each $i : i \geq 0$, $\sigma^{l_i:l_{i+1}} \models A$ holds.

Figure 2 pictorially illustrates the semantics of chop and chop-star in both finite and infinite time and also shows some simple PITL formulas together with intervals which satisfy them. For some sample formulas we include in parentheses versions using conventional PTL logic operators which were previously introduced in Section 3.

Remark 4.1. *The behaviour of chop-star on empty intervals is a frequent source of confusion and it is therefore important to note that any formula A^* (including $false^*$)*

A^+	$\stackrel{\text{def}}{\equiv}$	$A; A^*$	Chop-plus
A^ω	$\stackrel{\text{def}}{\equiv}$	$(A \wedge \mathit{finite})^* \wedge \mathit{inf}$	Chop-omega
A^n	$\stackrel{\text{def}}{\equiv}$	$\begin{cases} \mathit{empty} & \text{if } n = 0 \\ A; A^{n-1} & \text{otherwise} \end{cases}$	Fixed iteration
$A^{\leq n}$	$\stackrel{\text{def}}{\equiv}$	$\bigvee_{k \leq n} A^k$	
$A^{<n}$	$\stackrel{\text{def}}{\equiv}$	$\bigvee_{k < n} A^k$	
$\diamondsuit_{\text{i}} A$	$\stackrel{\text{def}}{\equiv}$	$A; \mathit{true}$	A is true in some initial subinterval
$\boxdot_{\text{i}} A$	$\stackrel{\text{def}}{\equiv}$	$\neg \diamondsuit_{\text{i}} \neg A$	A is true in all initial subintervals
$\diamondsuit_{\text{a}} A$	$\stackrel{\text{def}}{\equiv}$	$\mathit{finite}; A; \mathit{true}$	A is true in some subinterval
$\boxdot_{\text{a}} A$	$\stackrel{\text{def}}{\equiv}$	$\neg \diamondsuit_{\text{a}} \neg A$	A is true in all subintervals

Table 2: Some useful derived PITL operators

is true on a one-state interval. This is because in the semantics of chop-star for a one-state interval we can always set $n = 0$ and therefore ignore the values of variables in the interval.

Table 2 shows several especially useful derived PITL operators, including some variants of chop-star.

The notions of satisfiability and validity already introduced in Definition 3.1 for PTL naturally generalize to PITL.

Let PITL_V be the set of all PITL formulas only having variables in V.

The next definition introduces a special kind of state formula which is indispensable for interval-based reasoning. It plays the role that sets of formulas typically do in other analyses of PTL.

Definition 4.2 (Atoms and V-atoms)**.** *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 set of all atoms is denoted $Atoms$. The Greek letters α, β and γ denote individual atoms. For any finite set of propositional variables V, let $Atoms_V$ be some set of $2^{|V|}$ logically distinct atoms containing exactly the variables in V. We refer to such atoms as V-*atoms*.*

For example, we can let $Atoms_{\{p,q\}}$ be the set of the four logically distinct 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. For any finite set of variables V, this immediately leads to a suitable set of $2^{|V|}$ different V-atoms.

5 Transition configurations

Starting with a finite set of variables V, an NL^1_V formula T and a state formula init in PROP_V, we consider small models, decision procedures and axiomatic completeness

for certain low-level formulas referred to here as *transition configurations*. These formulas play a central role in our approach. The analysis of arbitrary PTL formulas can be ultimately reduced to that of transition configurations.

Before actually formally defining transition configurations, we need to introduce the concept of a *conditional liveness formula* which is a specific kind of conjunction necessary for reasoning about liveness properties involving infinite time. The definition therefore makes use of some general notation already introduced in Definitions 3.4–3.6 for manipulating conjunctions.

Definition 5.1 (Conditional liveness formulas, enabling tests and liveness tests)**.** *A* conditional liveness formula L *is a conjunction of* $|L|$ *implications* $L[1] \wedge \cdots \wedge L[|L|]$. *Each implication has the form* $w \supset \diamondm w'$, *where* w *and* w' *are two state formulas. For convenience, we let* $\eta_{L[k]}$ *denote the left operand of* L*'s* k*-th implication* $L[k]$. *Similarly,* $\theta_{L[k]}$ *denotes the operand of the* $\diamondm$ *formula on the right side of* L*'s* k*-th implication. It follows from all of this that for each* $k : 1 \leq k \leq |L|$, *the implications* $L[k]$ *and* $\eta_{L[k]} \supset \diamondm \theta_{L[k]}$ *denote the same formula. In addition, every* $\eta_{L[i]}$ *and* $\theta_{L[i]}$ *is a state formula.*

Each $\eta_{L[k]}$ *is called an* enabling test.

Each $\theta_{L[k]}$ *is called a* liveness test.

For any V*-atom* α *and any* $k : 1 \leq k \leq |L|$, *if the formula* $\alpha \wedge \eta_{L[k]}$ *is satisfiable, we say that* α enables L*'s* k*-th implication* $L[k]$.

Here is a sample conditional liveness formula:

$$((p \vee \neg q) \supset \diamondm \neg p) \quad \wedge \quad (q \supset \diamondm(p \equiv \neg q)) \quad \wedge \quad (true \supset \diamondm(p \supset q)). \qquad (1)$$

If we denote the overall formula as L, then, for example, the enabling test $\eta_{L[2]}$ is q and liveness test $\theta_{L[3]}$ is $p \supset q$.

Note that $\diamondm$ behaves the same as $\Diamond$ on infinite intervals. However, in finite intervals $\diamondm$, like its dual $\boxm$, ignores the final state. In principle, either $\diamondm$ or $\Diamond$ can be used in conditional liveness formulas and the choice between them appears to be largely a matter of taste. Nevertheless, we choose to use $\diamondm$ in part because it facilitates an interesting generalization of both conditional liveness formulas and another kind of formula called an *invariant* which is introduced later in Section 11. This generalization will be mentioned in Subsection 13.3. In addition, the application of $\diamondm$ naturally complements our extensive use of its dual $\boxm$.

Here is the definition of transition configurations:

Definition 5.2 (Transition configurations)**.** *A* transition configuration *is a formula of the form* $\Box T \;\; \wedge \;\; X$, *where the formula* T *is in* NL^1_V, *and the* PTL_V *formula* X *has one of the four forms shown below:*

Type of transition configuration	*Syntax of* X
Finite-time	$init \wedge finite$
Infinite-time	$init \wedge \Box \Diamond^+ L$
Final	$w \wedge empty$
Periodic	$\alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L)$

Here $init$ *is a state formula in* PROP_V *which corresponds to some* initial *condition,* w *is some state formula in* PROP_V, L *is a conditional liveness formula in* PTL_V *and* α *is a* V*-atom. If* $init$ *is the formula* $true$, *it can be omitted. The same applies with* w.

Type of transition configuration	PITL$_V$ formula	Where proved
Finite-time	$((\$\,T)^* \wedge \mathit{init} \wedge \mathit{finite}); (T \wedge \mathit{empty})$	Theorem 5.10
Infinite-time	$((\$\,T)^* \wedge \mathit{init} \wedge \mathit{finite}); \big((\$\,T)^* \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^\omega$	Theorem 5.19
Final	$T \wedge w \wedge \mathit{empty}$	straightforward
Periodic	$((\$\,T)^* \wedge \alpha \wedge L)^\omega$	Theorem 5.17

Table 3: Reduction of transition configurations to PITL$_V$ formulas

For example, the conjunction $\Box(\mathit{more} \supset (p \equiv \bigcirc p)) \wedge p \wedge \mathit{finite}$ is a finite-time transition configuration which is true exactly for finite intervals in which p is always true.

Note: In the course of analysing transition configurations, we will assume that V, T, init and L are fixed.

We will show that finite-time and infinite-time transition configurations are equivalent to certain PITL$_V$ formulas for which we can more readily establish such things as the existence of periodic models, small models, decision procedures and axiomatic completeness. Table 3 shows the corresponding PITL$_V$ formula for each kind of transition configuration and where the equivalence of the two is proved. Recall the definition of chop-omega in Table 2. Also, in Table 3 and elsewhere the formula $\vec{V} \leftarrow \vec{V}$ denotes that the initial value of each variable occurring in the set of variables V equals its final value. It is a natural generalization of the temporal assignment operator $\leftarrow$ previously introduced in Table 1 and can be defined as follows within PTL$_V$:

$$\vec{V} \leftarrow \vec{V} \quad \stackrel{\text{def}}{\equiv} \quad \mathit{finite} \supset \bigwedge_{v \in V} ((\mathit{fin}\ v) \equiv v).$$

Consequently, $\vec{V} \leftarrow \vec{V}$ is semantically equivalent to the disjunction given below:

$$\bigvee_{\alpha \in \mathit{Atoms}_V} (\alpha \wedge \mathit{fin}\ \alpha).$$

In addition to the theorems summarized in Table 3, Theorem 5.29 will establish that an infinite-time transition configuration is satisfiable iff the next PTL formula is satisfiable in finite time:

$$\boxdot T \wedge \mathit{init} \wedge \Diamond(L \wedge \mathit{finite} \wedge \mathit{more} \wedge (\vec{V} \leftarrow \vec{V})).$$

In order to perform interval-based analysis on transition configurations, we need to relate $\Box\, T$ to the PITL formula $(\$\,T)^*$. Now the PTL formula $\boxdot T$, which is very similar to $\Box\, T$, was previously defined in Table 1 to be true on an interval iff T is true in all of the interval's nonempty suffix subintervals. It turns out that due to T being in NL^1, the formula $(\$\,T)^*$ is semantically equivalent to $\boxdot T$. Intuitively, this is because an NL^1 formula cannot probe past the second state of an interval. The next lemma formalizes this:

Lemma 5.3. *Let σ and σ' be two nonempty intervals which share the same first two states (i.e., $\sigma^0 = (\sigma')^0$ and $\sigma^1 = (\sigma')^1$). Then, for any formula T in* NL^1*, σ satisfies T iff σ' satisfies T.*

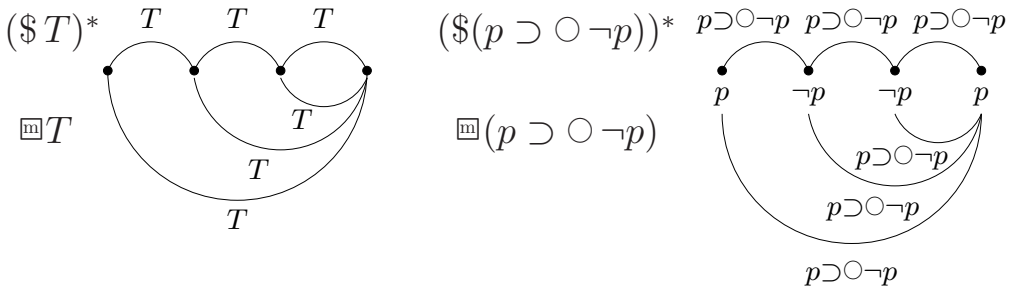


Figure 3: Illustration of equivalence of $(\$\,T)^*$ and $\boxed{\text{m}}\,T$

Proof. Induction on T's syntax ensures that it cannot distinguish between σ and σ'. □

Consequently, if two nonempty intervals share the same first two states, then the truth value of T for both intervals is identical. Figure 3 illustrates this with two instances of an interval containing 4 states. The second version uses the concrete NL^1 formula $p \supset \bigcirc \neg p$ and shows specific values for the propositional variable p. Both $(\$\,T)^*$ and $\boxed{\text{m}}\,T$ test each pair of adjacent states. The equivalence consequently permits us to express $(\$\,T)^*$ in PTL by means of $\boxed{\text{m}}\,T$. In addition, it is often useful to express $\boxed{\text{m}}\,T$ as $(\$\,T)^*$ because the later turns out to be much more suitable for interval-based reasoning involving sequential composition and decomposition.

We now formally establish the semantic equivalence of the formulas $(\$\,T)^*$ and $\boxed{\text{m}}\,T$:

Theorem 5.4. *For any* NL^1 *formula* T*, the* PITL *formula* $(\$\,T)^*$ *and the* PTL *formula* $\boxed{\text{m}}\,T$ *are semantically equivalent and hence the equivalence* $(\$\,T)^* \equiv \boxed{\text{m}}\,T$ *is valid.*

Proof. Given an interval σ, we can put each two-state (unit) subinterval in one-to-one correspondence with the suffix (nonempty) subinterval which shares the same first two states. Now σ satisfies $(\$\,T)^*$ iff T is true on all of σ's unit subintervals. Similarly, σ satisfies $\boxed{\text{m}}\,T$ iff T is true on all of σ's nonempty suffix subintervals. By the previous Lemma 5.3 a given unit subinterval satisfies T iff the matching suffix (nonempty) subinterval satisfies T. Consequently, the overall interval satisfies $(\$\,T)^*$ iff it satisfies $\boxed{\text{m}}\,T$. □

It is not hard to check that on a one-state (empty) interval, $\boxed{\text{m}}\,T$ is trivially true. On a two-state (unit) interval, it is semantically equivalent to the formula T itself.

Also note that the PTL formula $\Box\,T$ is semantically equivalent to the PTL formula $\boxed{\text{m}}\,T \wedge \mathit{fin}\ T$. This fact and Theorem 5.4 together establish that $\Box\,T$ is also semantically equivalent to the PITL formula $(\$\,T)^* \wedge \mathit{fin}\ T$. Therefore, the formula $\Box\,T$ in transition configurations can be readily re-expressed in PITL as the conjunction $(\$\,T)^* \wedge \mathit{fin}\ T$. This will assist our interval-based analysis of transition configurations.

Remark 5.5. *We have discussed the important semantic equivalence of the formulas* $(\$\,T)^*$ *and* $\boxed{\text{m}}\,T$ *with quite a few people who themselves have a considerable amount of experience with both* PTL *and* PITL*. Originally we thought that this amounted to a straightforward application of temporal logic. However, to our surprise, these people found the equivalence and its applications to be nontrivial and interesting. For this reason, we have designated the statement of the equivalence of* $(\$\,T)^*$ *and* $\boxed{\text{m}}\,T$ *to be a theorem (i.e., the previous Theorem 5.4), rather than merely a lemma.*

It is also worth considering a more syntactic approach to demonstrating the semantic equivalence of $(\$\,T)^*$ *and* $\boxed{\text{m}}\,T$ *since some readers might find this alternative way helpful. Recall the unary* ITL *operator* $\diamondsuit$ *defined in Table 2 for testing whether its operand is true in some initial subinterval. Now observe that* $(\$\,T)^*$ *is semantically equivalent to the* PITL *formula* $\boxed{\text{m}}\diamondsuit(T \wedge \mathit{skip})$ *since both formulas examine* T *in all two-state subintervals. Furthermore, owing to* T *being in* NL[1]*, the formulas* T *and* $\diamondsuit(T \wedge \mathit{skip})$ *are semantically equivalent on any nonempty interval since neither of them examines beyond the second state (see Lemma 5.3). Consequently,* $\boxed{\text{m}}\diamondsuit(T \wedge \mathit{skip})$ *and* $\boxed{\text{m}}\,T$ *are equivalent since* $\boxed{\text{m}}$*'s sole operand is only tested on nonempty subintervals. Therefore* $(\$\,T)^*$ *and* $\boxed{\text{m}}\,T$ *are indeed semantically equivalent.*

Here is a corollary of Theorem 5.4 for infinite time:

Corollary 5.6. *The two formulas* $\Box\,T$ *and* $(\$\,T)^*$ *are semantically equivalent on infinite intervals and hence the implication* $\mathit{inf} \supset \Box\,T \equiv (\$\,T)^*$ *is valid.*

Proof. This readily follows from Theorem 5.4 and the semantic equivalence of $\boxed{\text{m}}\,T$ and $\Box\,T$ on infinite intervals. □

The next two Lemmas 5.7 and 5.8 subsequently provide a basis for relating finite-time transition configurations to final ones and also for relating infinite-time transition configurations to periodic ones.

Lemma 5.7. *For any* PITL *formula* A*, the next equivalence is valid:*

$$\models \quad \Box\,T \wedge \diamondsuit\,A \;\equiv\; (\$\,T)^* \wedge \diamondsuit(\Box\,T \wedge A).$$

Proof. We first establish the validity of the PTL formula $\Box\,p \;\equiv\; \boxed{\text{m}}\,p \wedge \diamondsuit\,\Box\,p$ which itself leads to the validity of the formula $\Box\,p \wedge \diamondsuit\,q \;\equiv\; \boxed{\text{m}}\,p \wedge \diamondsuit(\Box\,p \wedge q)$. We then substitute T into p and A into q. Finally, Theorem 5.4 permits us to replace $\boxed{\text{m}}\,T$ by $(\$\,T)^*$. □

Lemma 5.8. *For any state formula* w *and* PITL *formula* A*, the next equivalence is valid:*

$$\Box\,T \wedge w \wedge \diamondsuit\,A \quad\equiv\quad ((\$\,T)^* \wedge w \wedge \mathit{finite}); (\Box\,T \wedge A). \tag{2}$$

Proof. Lemma 5.7 ensures that $\Box\,T \wedge \diamondsuit\,A$ is semantically equivalent to the conjunction $(\$\,T)^* \wedge \diamondsuit(\Box\,T \wedge A)$. This is itself semantically equivalent to the next PITL formula:

$$((\$\,T)^* \wedge \mathit{finite}); ((\$\,T)^* \wedge \Box\,T \wedge A).$$

Now $\Box\,T$ trivially implies $\boxed{\text{m}}\,T$ which by Theorem 5.4 is semantically equivalent to $(\$\,T)^*$. This consequently permits us to simplify the subformula $(\$\,T)^* \wedge \Box\,T$ into $\Box\,T$ to obtain the next valid equivalence:

$$\models \quad \Box\,T \wedge \diamondsuit\,A \quad\equiv\quad ((\$\,T)^* \wedge \mathit{finite}); (\Box\,T \wedge A).$$

Simple temporal reasoning permits us to suitably add the state formula w to each side to obtain the validity of the formula (2). □

5.1 Analysis of finite-time behaviour

The following Lemma 5.9 and Theorem 5.10 concern reducing a finite-time transition configuration to the associated semantically equivalent PITL formula in Table 3 which is easier to later analyse.

Lemma 5.9. *The following equivalence is valid for finite-time transition configurations and relates them to final configurations:*

$$\models \quad \Box\, T \wedge \mathit{init} \wedge \mathit{finite} \;\equiv\; ((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}); (\Box\, T \wedge \mathit{empty}). \qquad (3)$$

Proof. The PTL formula finite is defined to be $\Diamond\, \mathit{empty}$ in Table 1. Lemma 5.8 then ensures the validity of the equivalence (3). □

Theorem 5.10 builds on Lemma 5.9 by reducing a finite-time transition configuration to a chop formula in PITL which is even easier to analysis because its righthand operand is in NL[1]:

Theorem 5.10. *The following equivalence is valid for finite-time transition configurations:*

$$\models \quad \Box\, T \wedge \mathit{init} \wedge \mathit{finite} \;\equiv\; ((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}); (T \wedge \mathit{empty}).$$

Proof. This readily follows from Lemma 5.9 and the fact that in an empty interval, the formulas $\Box\, T$ and T are equivalent. □

Note that we can use the unary PTL operator ? (previously defined in Table 1) to alternatively express the PITL formula $((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}); (T \wedge \mathit{empty})$ as the semantically equivalent PITL formulas $\mathit{init}?; ((\$\, T)^* \wedge \mathit{finite}); T?$ and $\mathit{init} \wedge (\$\, T)^* \wedge \mathit{sfin}\ T$. Each form has its benefits. We prefer $T \wedge \mathit{empty}$ over the equivalent $T?$ since some readers might get confused upon seeing the operator ? with an operand which is itself a temporal formula even though this is permitted.

5.2 Analysis of infinite-time behaviour

We now turn to analysing infinite-time transition configurations. The first step involves relating them to periodic transition configurations. The next Lemma 5.11 does this:

Lemma 5.11. *The following equivalence is valid for infinite-time transition configurations:*

$$\begin{aligned} &\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L \\ &\quad \equiv \quad ((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}); \textstyle\bigvee_{\alpha \in \mathit{Atoms}_V} (\Box\, T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)). \end{aligned} \qquad (4)$$

Proof. Observe that in an infinite interval if the conditional liveness formula L is always eventually true then for at least one of the finite number of V-atoms, the conjunction $\alpha \wedge L$ is also always eventually true. Therefore simple temporal reasoning yields that $\Box \Diamond^+ L$ is semantically equivalent to the disjunction $\bigvee_{\alpha \in \mathit{Atoms}_V} \Box \Diamond^+ (\alpha \wedge L)$. The subformula $\Box \Diamond^+ (\alpha \wedge L)$ can be re-expressed as $\Diamond(\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L))$ so the next equivalence concerning $\Box \Diamond^+ L$ is valid:

$$\models \quad \Box \Diamond^+ L \;\equiv\; \textstyle\bigvee_{\alpha \in \mathit{Atoms}_V} \Diamond(\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)).$$

We can then swap the instances of $\bigvee$ and $\Diamond$ to obtain the following valid equivalence:

$$\models \quad \Box \Diamond^+ L \;\equiv\; \Diamond \bigvee_{\alpha \in Atoms_V} (\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)).$$

This permits us to re-express the infinite-time transition configuration $\Box T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ by means of the next valid equivalence:

$$\begin{aligned} \models \quad & \Box T \wedge \mathit{init} \wedge \Box \Diamond^+ L \\ & \equiv \; \Box T \wedge \mathit{init} \wedge \Diamond \bigvee_{\alpha \in Atoms_V} (\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)). \end{aligned} \quad (5)$$

We invoke Lemma 5.8 on the righthand operand of this equivalence to establish the validity of the equivalence below:

$$\begin{aligned} \models \quad & \Box T \wedge \mathit{init} \wedge \Diamond \bigvee_{\alpha \in Atoms_V} (\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)) \\ \equiv \quad & ((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}); \big(\Box T \wedge \bigvee_{\alpha \in Atoms_V} (\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L))\big). \end{aligned} \quad (6)$$

The righthand operand of the chop construct in (6) can be re-expressed as shown by the valid equivalence now given:

$$\begin{aligned} \models \quad & \Box T \wedge \bigvee_{\alpha \in Atoms_V} (\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)) \\ & \equiv \; \bigvee_{\alpha \in Atoms_V} \big(\Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)\big). \end{aligned}$$

The justification of this only involves conventional propositional reasoning. Consequently, the equivalence (6) can be itself re-expressed as follows:

$$\begin{aligned} \models \quad & \Box T \wedge \mathit{init} \wedge \Diamond \bigvee_{\alpha \in Atoms_V} (\alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)) \\ \equiv \quad & ((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}); \bigvee_{\alpha \in Atoms_V} \big(\Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L))\big). \end{aligned} \quad (7)$$

The second operand of equivalence (5) is identical to the first operand of equivalence (7). Consequently, the conjunction of these two equivalences implies the initial equivalence (4). Hence, the validity of (5) and (7) yields our goal which is the validity of (4). □

5.2.1 Reduction using chop-omega operator

Much of the remainder of the analysis of transition configurations consists of showing how to further reduce a periodic transition configuration $\Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)$ to the semantically equivalent PITL formula $((\$\, T)^* \wedge \alpha \wedge L)^\omega$. A general class of formulas which includes $\alpha \wedge L$ will now be described. For any PITL formula A in this class, the two formulas $A \wedge \Box \Diamond^+ A$ and A^ω will be shown to be semantically equivalent in Theorem 5.16. We first need to introduce a derived PITL operator which turns out to be useful for analysing periodic behaviour in infinite intervals.

Definition 5.12 (The operator $\Diamond\!\!\!\!\cdot$)**.** *For any* PITL *formula* A*, let the* PITL *formula* $\Diamond\!\!\!\!\cdot\, A$ *is defined to be* $(A \wedge \mathit{finite}); \mathit{true}$*. Therefore,* $\Diamond\!\!\!\!\cdot\, A$ *is true on an interval iff* A *is true on some finite subinterval starting at the beginning of the overall interval.*

Note that $\Diamond\!\!\!\!\cdot\, A$ can also be expressed with the derived operator $\Diamond\!\!\!\!\cdot$ (itself previously defined in Table 2) as $\Diamond\!\!\!\!\cdot (A \wedge \mathit{finite})$.

It is worthwhile to define a notion of fixpoints of the operator $\Diamond\!\!\!\!\cdot$:

Definition 5.13 (Fixpoints of the operator $\diamondsuit$)**.** *A* PITL *formula A is a fixpoint of $\diamondsuit$ iff the equivalence $A \equiv \diamondsuit A$ is valid.*

Fixpoints of $\diamondsuit$ are easier to move out of subintervals than are arbitrary formulas. Incidentally, for any PITL formula A, the formula $\diamondsuit A$ is a trivial fixpoint of $\diamondsuit$ since $\diamondsuit A$ and $\diamondsuit \diamondsuit A$ are semantically equivalent. We will shortly show that all conditional liveness formulas are $\diamondsuit$-fixpoints and later use this in the analysis of infinite intervals.

We extensively investigate fixpoints of various temporal operators and their application to compositional reasoning in [68–71].

The next lemma characterizes a broad syntactic class of formulas which are $\diamondsuit$-fixpoints and is easy to check:

Lemma 5.14. *Every state formula is a $\diamondsuit$-fixpoint. Furthermore, if the* PITL *formulas A and B are $\diamondsuit$-fixpoints, then so are the* PITL *formulas $A \wedge B$, $A \vee B$, $\bigcirc A$ and $\Diamond A$.*

Lemma 5.15. *Every conditional liveness formula is a $\diamondsuit$-fixpoint.*

Proof. By Definition 5.1, a conditional liveness formula is a conjunction of implications each which has the form $w \supset \circledcirc w'$ for some state formulas w and w'. If we replace $\supset$ and $\circledcirc$ by their definitions, then the implication reduces to the formula $\neg w \vee \Diamond((\bigcirc true) \wedge w')$. Lemma 5.14 then ensures that this is a $\diamondsuit$-fixpoint. Consequently, the original implication $w \supset \circledcirc w'$ is one as well. Therefore by Lemma 5.14, the conjunction of such implications which constitutes a conditional liveness formula is also a $\diamondsuit$-fixpoint. □

Observe that by Lemmas 5.14 and 5.15, the formula $\alpha \wedge L$ is itself a $\diamondsuit$-fixpoint because both α and L are $\diamondsuit$-fixpoints.

Now the formula $\alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L)$ is itself an instance of the PITL formula $A \wedge \Box \Diamond^+ A$. We now proof in Theorem 5.16 that if A is a $\diamondsuit$-fixpoint, then the formula $A \wedge \Box \Diamond^+ A$ can be re-expressed as the semantically equivalent PITL formula A^ω. This will let us re-express $\alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L)$ as the semantically equivalent PITL formula $(\alpha \wedge L)^\omega$. The establishment of this equivalence is a key step in the reduction of reasoning about infinite time behaviour to finite time behaviour and consequently proving the existence of periodic models for satisfiable periodic transition configurations.

Theorem 5.16. *For any* PITL *formula A which is a $\diamondsuit$-fixpoint, the next equivalence is valid:*

$$\models \quad A \wedge \Box \Diamond^+ A \quad \equiv \quad A^\omega.$$

Proof. Left side implies right side: Suppose that an interval σ satisfies $A \wedge \Box \Diamond^+ A$. Now this conjunction is semantically equivalent to the formula $\diamondsuit A \wedge \Box \Diamond^+ \diamondsuit A$ because A is a $\diamondsuit$-fixpoint. Therefore σ also satisfies the formula $\diamondsuit A \wedge \Box \Diamond^+ \diamondsuit A$. Furthermore, σ is clearly an infinite interval due to the conjunct containing $\Box \Diamond^+$. Therefore, σ has an infinite number of finite subintervals which all satisfy A including at least one starting with σ's initial state σ^0. An infinite sequence of nonoverlapping finite-length subintervals all satisfying A can then be selected with the first one commencing at the beginning of σ. Consequently, σ satisfies the PITL formula $((A \wedge finite); true)^\omega$ which is the same as $(\diamondsuit A)^\omega$. This and the assumption that A is a $\diamondsuit$-fixpoint together yield that σ satisfies A^ω.

Right side implies left side: Suppose that an interval σ satisfies A^ω. Therefore σ is an infinite interval and has an infinite number of finite subintervals all satisfying A,

including one starting with σ's initial state. From this we can readily obtain the valid PITL implication shown below:

$$\models \quad A^\omega \quad \supset \quad (A \wedge \mathit{finite}); \mathit{true} \ \wedge \ \Box \Diamond^+((A \wedge \mathit{finite}); \mathit{true}).$$

This can be re-expressed using $\diamondsuit$ as follows:

$$\models \quad A^\omega \quad \supset \quad \diamondsuit A \ \wedge \ \Box \Diamond^+ \diamondsuit A.$$

The assumption that A is a $\diamondsuit$-fixpoint then yields the desired validity of the semantically equivalent implication $A^\omega \ \supset \ A \wedge \Box \Diamond^+ A$. □

The following Theorem 5.17 relates any periodic transition configuration with its associated PITL formula shown in Table 3:

Theorem 5.17. *The next equivalence concerning a periodic transition configuration is valid:*

$$\models \quad \Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L) \ \equiv \ ((\$\, T)^* \wedge \alpha \wedge L)^\omega. \tag{8}$$

Proof. Lemmas 5.14 and 5.15 ensure that the formula $\alpha \ \wedge \ L$ is itself a $\diamondsuit$-fixpoint because both α and L are $\diamondsuit$-fixpoints. Therefore Theorem 5.16 yields the validity of the following equivalence:

$$\models \quad \alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L) \quad \equiv \quad (\alpha \wedge L)^\omega.$$

We then conjoin $\Box T$ to each side of the equivalence. Recall the fact that $\Box T$ and $(\$\, T)^*$ are semantically equivalent in infinite time (Corollary 5.6) so the equivalence below is valid:

$$\models \quad \Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L) \ \equiv \ (\$\, T)^* \wedge (\alpha \wedge L)^\omega.$$

Now $(\$\, T)^* \wedge (\alpha \wedge L)^\omega$ is an instance of the PITL formula $(\$\, B)^* \wedge C^\omega$ which itself is semantically equivalent to $((\$\, B)^* \ \wedge \ C)^\omega$. The intuition here is that both of them use $\$\, B$ to test exactly all the two-state subintervals of the overall interval. Finally, we use this to re-express $(\$\, T)^* \wedge (\alpha \wedge L)^\omega$ as $((\$\, T)^* \wedge \alpha \wedge L)^\omega$, thereby obtaining the validity of formula (8). □

The following Lemma 5.18 concerning a disjunction of periodic transition configurations is needed to justify our reduction of the satisfiability of a infinite-time transition configuration to the associated PITL formula shown in Table 3:

Lemma 5.18. *The next equivalence is valid:*

$$\models \quad \bigvee\nolimits_{\alpha \in Atoms_V} (\Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L)) \\ \equiv \quad \big((\$\, T)^* \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^\omega. \tag{9}$$

Proof. Theorem 5.17 ensures that the equivalence given below is valid:

$$\models \quad \Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+(\alpha \wedge L) \quad \equiv \quad ((\$\, T)^* \wedge \alpha \wedge L)^\omega.$$

We then use some simple temporal reasoning to establish that this equivalence's right-hand operand $((\$\, T)^* \wedge \alpha \wedge L)^\omega$ can be re-expressed as shown in the next equivalence:

$$\models \quad ((\$\, T)^* \wedge \alpha \wedge L)^\omega \quad \equiv \quad \alpha \ \wedge \ \big((\$\, T)^* \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^\omega.$$

These two equivalences are now combined to obtain the valid one given below:

$$\models \quad \Box T \wedge \alpha \wedge L \wedge \Box \Diamond^{+}(\alpha \wedge L) \quad \equiv \quad \alpha \ \wedge \ \big((\$\, T)^{*} \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^{\omega}.$$

This and some further simple reasoning about the operator $\bigvee$ yields the validity of the following equivalence:

$$\models \quad \bigvee_{\alpha \in Atoms_V} (\Box T \wedge \alpha \wedge L \wedge \Box \Diamond^{+}(\alpha \wedge L))$$
$$\equiv \quad \bigvee_{\alpha \in Atoms_V} \Big(\alpha \ \wedge \ \big((\$\, T)^{*} \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^{\omega}\Big).$$

The righthand side can be re-expressed as $\big((\$\, T)^{*} \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^{\omega}$ and thereby yields the validity of the equivalence (9). □

The equivalence of an infinite-time transition configuration with the associated PITL formula shown in Table 3 is now established:

Theorem 5.19. *The following equivalence is valid for infinite-time transition configurations:*

$$\models \quad \Box T \wedge \mathit{init} \wedge \Box \Diamond^{+} L$$
$$\equiv \quad ((\$\, T)^{*} \wedge \mathit{init} \wedge \mathit{finite}); \big((\$\, T)^{*} \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^{\omega}.$$

Proof. This readily follows from Lemma 5.11 which relates infinite-time transition configurations to periodic transition configurations and Lemma 5.18 which re-expresses the disjunction of several periodic transition configurations using chop-omega. □

5.2.2 Fusion and canonical intervals

Let us consider some general concepts and techniques concerning PITL and its notion of intervals. They will be extensively used later on.

Definition 5.20 (Fusion)**.** *Let σ and σ' be two intervals. The definition of the* fusion *of them, denoted $\sigma \circ \sigma'$, has two cases, depending on whether σ has finite length or not:*

- *If σ has finite length, we require that the last state of σ equals the first state of σ'. The fusion of σ with σ' is then the interval obtained by appending the two intervals together so as to include only one copy of the shared state.*
- *Otherwise, the fusion is σ itself, no matter what σ' is.*

For example, suppose s_1, s_2 and s_3 are states. If σ is the interval $s_1 s_2$ and σ' is the interval $s_2 s_3$, then their fusion $\sigma \circ \sigma'$ equals the three-state interval $s_1 s_2 s_3$, rather than the four-state interval $s_1 s_2 s_2 s_3$ which concatenation yields. Note that when σ has finite length and σ and σ' do not share the relevant state, then their fusion is undefined. If both σ and σ' are finite and compatible, then the interval $\sigma \circ \sigma'$ contains the total sum of states in σ and σ' minus one. Hence the interval length of $\sigma \circ \sigma'$ equals the sum of the interval lengths of σ and σ'. Pratt first defined fusion for describing the semantics of a process logic [79] and called it *fusion product* (see Harel, Kozen and Tiuryn [40, Section 17.3] for a tutorial on process logics).

It is worth comparing chop and fusion. Fusion is a general operation definable for such things as strings (i.e., sequences of letters) or intervals (i.e., sequences of states).

As used here, it starts with two suitable intervals and joins them together. In contrast, chop is a logical operator which starts with an overall interval and then tests for the existence of a way to split it into two fusible subintervals. Furthermore, the semantics of the chop operator can be defined using fusion, whereas fusion is for our purposes a semantic concept, not a logical construct.

Here is a lemma relating chop with fusion:

Lemma 5.21. *A* PITL *formula $A; B$ is satisfiable iff there exist two intervals σ and σ' such that the fusion of them $\sigma \circ \sigma'$ is defined and one of the following is true:*

- *The interval σ has finite length, it satisfies A and the interval σ' satisfies B.*
- *The interval σ has infinite length and it satisfies A.*

This lemma provides a way to reduce the problem of constructing an interval satisfying $A; B$ to that of constructing intervals satisfying A and B.

Before further reducing transition configurations involving infinite time, we introduce the notion of *canonical intervals* and discuss their use in relating the satisfiability of chop and chop-omega formulas with satisfiability of their operands.

The next definition of a notion of canonical states and intervals together with the subsequent Lemma 5.23 will be extensively utilized to facilitate reasoning about intervals.

Definition 5.22 (Canonical states and intervals)**.** *For any finite set of variables V and state s, we say that s is a V-*state *if s assigns each variable* not *in V the value $false$.*

*Similarly, for any finite set of variables V and interval σ, we say that σ is a V-*interval *if σ's states all assign each variable* not *in V the value $false$.*

Furthermore, for any set of variables V, we can denote a V-state by the unique V-atom which the state satisfies. In addition, a V-interval can be denoted by the unique sequence of V-atoms associated with its V-states.

For example, for any V-atoms α and β, the two-atom sequence $\alpha\beta$ denotes a finite V-interval with V-states denoted by α and β, respectively. Hence, $\alpha\beta \models X$ denotes that the two-state V-interval $\alpha\beta$ satisfies the formula X. If X is in PTL_V, then $\alpha\beta \models X$ holds iff the conjunction $\alpha \wedge \bigcirc\beta? \wedge X$ is satisfiable. Furthermore a single V-atom can be regarded as a one-state V-interval. For example, $\alpha \models X$ denotes that the one-state V-interval α satisfies X. For any X in PTL_V, this is the case iff the conjunction $\alpha \wedge X \wedge \mathit{empty}$ is satisfiable. Similarly, the notation $\alpha\beta\alpha \models X$ denotes that the V-interval $\alpha\beta\alpha$, which has two identical states, satisfies the formula X.

The next lemma ensures that any satisfiable PITL_V formula is satisfied by some V-interval.

Lemma 5.23. *An interval σ satisfies a* PITL_V *formula A iff there exists a V-interval with the same number of states as σ, agrees with σ on the values of the variables in V and moreover satisfies A.*

Proof. Let σ' be the V-interval obtained from σ by setting all variables not in the set V to false in each state. The semantics in PITL of A ignores such variables. □

The following lemma employs V-atoms and the PTL construct finite to express a simple sufficient condition which ensures that any two intervals which respectively satisfy the two operands in a chop formula with a particular syntax given in the lemma can be fused together into an interval which satisfies the overall chop formula.

Lemma 5.24. *For any V-atom α and* PITL$_V$ *formulas A and B, the following are equivalent:*

(a) The formula $(A \wedge \mathit{finite}); (\alpha \wedge B)$ is satisfiable.

(b) The formulas $A \wedge \mathit{sfin}\ \alpha$ and $\alpha \wedge B$ are satisfiable.

Proof. $(a) \Rightarrow (b)$*:* If some interval σ satisfies the formula $(A \wedge \mathit{finite}); (\alpha \wedge B)$, then by the semantics of chop there exist two subintervals of σ denoted here as σ' and σ'' such that the subinterval σ' satisfies $A \wedge \mathit{finite}$ and moreover if σ' has finite length, then σ'' satisfies $\alpha \wedge B$. The right subformula finite in $A \wedge \mathit{finite}$ ensures that σ' is indeed finite and therefore σ'' does satisfies $\alpha \wedge B$.

$(b) \Rightarrow (a)$*:* If the two formulas $A \wedge \mathit{sfin}\ \alpha$ and $\alpha \wedge B$ are satisfiable, then by Lemma 5.23 some *V*-intervals σ and σ' satisfy them. Now σ is finite due to the subformula $\mathit{sfin}\ \alpha$. Also, the last state of σ and the first state of σ' both equal the *V*-state denoted by the *V*-atom α. Hence σ and σ' can be fused and the fusion $\sigma \circ \sigma'$ satisfies the formula $(A \wedge \mathit{finite}); (\alpha \wedge B)$. □

5.2.3 Periodic models and reduction to finite-time behaviour

The remaining material in this section deals with relating transition configurations involving infinite time to other formulas involving periodicity as well as to formulas about finite time. The connections are interesting in themselves and also later utilized.

The next Lemmas 5.25 and 5.26 help to establish small models, decidability and axiomatic completeness for periodic transition configurations:

Lemma 5.25. *For any V-atom α and* PITL$_V$ *formula A, the following are equivalent:*

(a) The formula $(\alpha \wedge A)^\omega$ is satisfiable.

(b) The formula $(\alpha \wedge A)^\omega$ has a periodic model.

(c) The formula $\alpha \wedge A \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable (in some finite-time interval).

Proof. $(a) \Rightarrow (c)$*:* Suppose the interval σ satisfies $(\alpha \wedge A)^\omega$. We can assume each iteration of $\alpha \wedge A$ occurs in a nonempty, finite interval as expressed by the next valid equivalence:

$$\models \quad (\alpha \wedge A)^\omega \ \equiv\ (\alpha \wedge A \wedge \mathit{finite} \wedge \mathit{more})^\omega.$$

Furthermore, each pair of adjacent iterations share a common state satisfying α and hence all have α true at the beginning and end as is captured by the following valid equivalence:

$$\models \quad (\alpha \wedge A)^\omega \ \equiv\ (\alpha \wedge A \wedge \mathit{finite} \wedge \mathit{more} \wedge \mathit{fin}\ \alpha)^\omega.$$

Therefore the subformula $\alpha \wedge A \wedge \mathit{finite} \wedge \mathit{more} \wedge \mathit{fin}\ \alpha$ is satisfiable (in some finite-time interval) and hence the semantically equivalent formula $\alpha \wedge A \wedge \bigcirc \mathit{sfin}\ \alpha$ is also satisfiable.

$(c) \Rightarrow (b)$*:* Suppose the interval σ satisfies $\alpha \wedge A \wedge \bigcirc \mathit{sfin}\ \alpha$. As a consequence of α being a *V*-atom and A being a PITL$_V$ formula together with Lemma 5.23, we can assume without loss of generality that σ is a *V*-interval. We then readily fuse ω instances of σ together to obtain a periodic interval satisfying the formula $(\alpha \wedge A)^\omega$.

$(b) \Rightarrow (a)$*:* Clearly if some periodic interval satisfies $(\alpha \wedge A)^\omega$, then this formula is satisfiable. □

Lemma 5.26 shows that any satisfiable periodic transition configuration has a periodic model. Subsequently, Theorem 5.29 establishes that any satisfiable infinite-time transition configuration has an ultimately periodic model (i.e., an interval with a periodic suffix):

Lemma 5.26. *For any V-atom α, the following are equivalent:*

(a) The periodic transition configuration $\Box\, T \wedge \alpha \wedge L \wedge \Box \Diamond^{+}(\alpha \wedge L)$ is satisfiable.

(b) The periodic transition configuration $\Box\, T \wedge \alpha \wedge L \wedge \Box \Diamond^{+}(\alpha \wedge L)$ has a periodic model.

(c) The formula $(\$\, T)^{*} \wedge \alpha \wedge L \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable (in some finite-time interval).

Proof. Theorem 5.17 reduces the periodic transition configuration to the semantically equivalent PITL$_V$ formula $((\$\, T)^{*} \wedge \alpha \wedge L)^{\omega}$. We then utilize Lemma 5.25. □

Lemma 5.27. *For any V-atom α and* PITL$_V$ *formulas A and B, the following are equivalent:*

(a) The formula $(A \wedge \mathit{finite}); (\alpha \wedge B)^{\omega}$ is satisfiable.

(b) The formula $(A \wedge \mathit{finite}); (\alpha \wedge B)^{\omega}$ has an ultimately periodic model (i.e., an interval with a periodic suffix).

(c) The formula $(A \wedge \mathit{finite}); (\alpha \wedge B \wedge \bigcirc \mathit{sfin}\ \alpha)$ is satisfiable (in some finite-time interval).

Proof. $(a) \Rightarrow (c)$*:* If the formula $(A \wedge \mathit{finite}); (\alpha \wedge B)^{\omega}$ is satisfiable then the PITL$_V$ formula $(A \wedge \mathit{finite}); (\alpha \wedge B \wedge \bigcirc \mathit{sfin}\ \alpha)^{\omega}$ is also satisfiable. From this readily follows the satisfiability of the formula $(A \wedge \mathit{finite}); (\alpha \wedge B \wedge \bigcirc \mathit{sfin}\ \alpha)$.

$(c) \Rightarrow (b)$*:* If the formula $(A \wedge \mathit{finite}); (\alpha \wedge B \wedge \bigcirc \mathit{sfin}\ \alpha)$ is satisfiable then Lemma 5.24 ensures that the two formulas $A \wedge \mathit{finite} \wedge \mathit{fin}\ \alpha$ and $\alpha \wedge B \wedge \bigcirc \mathit{sfin}\ \alpha$ are also satisfiable. Lemma 5.25 then yields that the formula $(\alpha \wedge B)^{\omega}$ has a periodic model. Suppose the interval σ satisfies $A \wedge \mathit{finite} \wedge \mathit{fin}\ \alpha$ and the interval σ' is a periodic model of $(\alpha \wedge B)^{\omega}$. Lemma 5.23 permits us to assume that σ and σ' are V-intervals. We can fuse σ together with σ' to obtain an ultimately periodic model for $(A \wedge \mathit{finite}); (\alpha \wedge B)^{\omega}$.

$(b) \Rightarrow (a)$*:* Clearly if some ultimately periodic interval satisfies $(A \wedge \mathit{finite}); (\alpha \wedge B)^{\omega}$, then this formula is satisfiable. □

Lemma 5.28. *For any* PITL$_V$ *formulas A and B, the following are equivalent:*

(a) The formula $(A \wedge \mathit{finite}); (B \wedge (\vec{V} \leftarrow \vec{V}))^{\omega}$ is satisfiable.

(b) The formula $(A \wedge \mathit{finite}); (B \wedge (\vec{V} \leftarrow \vec{V}))^{\omega}$ has an ultimately periodic model.

(c) The formula $(A \wedge \mathit{finite}); (B \wedge \mathit{more} \wedge \mathit{finite} \wedge (\vec{V} \leftarrow \vec{V}))$ is satisfiable (in some finite-time interval).

Proof. This follows from Lemma 5.27 and simple temporal reasoning involving chop and the operator $\bigvee$. We also make use of the following valid equivalences concerning $\vec{V} \leftarrow \vec{V}$, the formula B and any V-atom α:

$$\begin{array}{lll} \models & \alpha \wedge B \wedge \bigcirc \mathit{sfin}\ \alpha & \equiv \quad \alpha \wedge B \wedge \mathit{more} \wedge \mathit{finite} \wedge (\vec{V} \leftarrow \vec{V}) \\ \models & (\alpha \wedge B)^{\omega} & \equiv \quad \alpha \wedge (B \wedge (\vec{V} \leftarrow \vec{V}))^{\omega}. \end{array}$$

□

Type of transition configuration	Upper bounds	Where proved
Finite-time	Interval length less than $\lvert Atoms_V \rvert$	Theorem 6.2
Infinite-time	Initial part $< \lvert Atoms_V \rvert$, Period $\leq (\lvert L \rvert + 1) \cdot \lvert Atoms_V \rvert$	Theorem 6.9
Final	Interval length is 0	straightforward
Periodic	Period $\leq (\lvert L \rvert + 1) \cdot \lvert Atoms_V \rvert$	Lemma 6.8

Table 4: Summary of upper bounds of intervals for transition configurations

Theorem 5.29. *The following are equivalent:*

(a) The infinite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ *is satisfiable.*

(b) The infinite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ *has an ultimately periodic model.*

(c) The PITL$_V$ *formula* $\big((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}\big); \big((\$\, T)^* \wedge L \wedge \mathit{more} \wedge \mathit{finite} \wedge (\vec{V} \leftarrow \vec{V})\big)$ *is satisfiable (in some finite-time interval).*

(d) The PTL$_V$ *formula* $\boxdot T \wedge \mathit{init} \wedge \Diamond(L \wedge \mathit{finite} \wedge \mathit{more} \wedge (\vec{V} \leftarrow \vec{V}))$ *is satisfiable (in some finite-time interval).*

Proof. We need to obtain formulas which are in a form suitable for Lemma 5.28. First of all, Theorem 5.19 permits us to re-express the infinite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ as the formula $\big((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}\big); \big((\$\, T)^* \wedge L \wedge (\vec{V} \leftarrow \vec{V})\big)^{\omega}$. Recall that Theorem 5.4 shows the semantic equivalence of the formulas $\boxdot T$ and $(\$\, T)^*$. Therefore, simple interval-based temporal reasoning ensures that formulas in (c) and (d) are semantically equivalent. We complete the proof by invoking Lemma 5.28. □

6 Small models for transition configurations

We now turn to giving upper bounds on small models for satisfiable transition configurations. This is later used in Section 8 to construct a decision procedure for them. Table 4 summarizes the upper bounds for intervals satisfying the various kinds of transition configurations and where the results are proved.

It will be necessary to employ the fact (e.g., in Theorem 6.2 and Lemma 6.6) that the formula $\alpha \wedge (\$\, T)^* \wedge \mathit{sfin}\ \beta$ is satisfiable iff a simple variant of it is satisfiable in an interval of bounded interval length. The following lemma deals with this:

Lemma 6.1. *For any V-atoms* α *and* β*, the formula* $\alpha \wedge (\$\, T)^* \wedge \mathit{sfin}\ \beta$ *is satisfiable iff the formula* $\alpha \wedge (\$\, T)^{<\lvert Atoms_V \rvert} \wedge \mathit{sfin}\ \beta$ *is satisfiable. Hence, the formula* $\alpha \wedge (\$\, T)^* \wedge \mathit{sfin}\ \beta$ *is satisfiable iff it is satisfiable in an interval having interval length less than* $\lvert Atoms_V \rvert$*.*

Proof. Any interval satisfying $\alpha \wedge (\$\, T)^{<\lvert Atoms_V \rvert} \wedge \mathit{sfin}\ \beta$ can be readily seen to also satisfy $\alpha \wedge (\$\, T)^* \wedge \mathit{sfin}\ \beta$. Let us now establish the converse by doing a proof by contradiction. Suppose $\alpha \wedge (\$\, T)^* \wedge \mathit{sfin}\ \beta$ is satisfiable but $\alpha \wedge (\$\, T)^{<\lvert Atoms_V \rvert} \wedge \mathit{sfin}\ \beta$ is not. Let σ be any interval which has the smallest length of those which satisfy

$\alpha \wedge (\$\,T)^* \wedge \mathit{sfin}\ \beta$. Lemma 5.23 permits us to assume that σ is a V-interval. Now σ's length is greater than or equal to $|Atoms_V|$ and therefore contains at least $|Atoms_V|+1$ states. Consequently, some V-state occurs at least twice in σ. Let the V-atom γ denote this state. It follows that σ satisfies the following PITL$_V$ formula:

$$\alpha \ \wedge\ \big((\$\,T)^*; \gamma?; (\$\,T)^+; \gamma?; (\$\,T)^*\big)\ \wedge\ \mathit{sfin}\ \beta.$$

Therefore σ contains two proper subintervals σ' and σ'' which respectively satisfy the PITL$_V$ formulas $\alpha \wedge (\$\,T)^* \wedge \mathit{sfin}\ \gamma$ and $\gamma \wedge (\$\,T)^* \wedge \mathit{sfin}\ \beta$. In addition, the last state of σ' is the same as the first one of σ'' so σ' and σ'' can be fused together. The fusion $\sigma' \circ \sigma''$ has length strictly less than that of σ and furthermore, like σ, satisfies the formula $\alpha \wedge (\$\,T)^* \wedge \mathit{sfin}\ \beta$. But this violates the assumption that σ was amongst the shortest such intervals and yields a contradiction. □

Theorem 6.2. *If a finite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \mathit{finite}$ is satisfiable, then it is satisfied by some finite interval of length less than $|Atoms_V|$.*

Proof. Theorem 5.10 ensures that the finite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \mathit{finite}$ is semantically equivalent to the formula $((\$\,T)^* \wedge \mathit{init} \wedge \mathit{finite}); (T \wedge \mathit{empty})$. This is satisfiable iff for some V-atom α, the formula $((\$\,T)^* \wedge \mathit{init} \wedge \mathit{finite}); (\alpha \wedge T \wedge \mathit{empty})$ is satisfiable. Now Lemma 5.24 ensures that this itself is satisfiable iff the formulas $(\$\,T)^* \wedge \mathit{init} \wedge \mathit{sfin}\ \alpha$ and $\alpha \wedge T \wedge \mathit{empty}$ are both satisfiable. By Lemma 6.1, the first of these is satisfiable iff the formula $(\$\,T)^{<|Atoms_V|} \wedge \mathit{init} \wedge \mathit{sfin}\ \alpha$ is satisfiable. Lemma 5.23 permits us to assume without loss of generality that the intervals satisfying the formulas $(\$\,T)^{<|Atoms_V|} \wedge \mathit{init} \wedge \mathit{sfin}\ \alpha$ and $\alpha \wedge T \wedge \mathit{empty}$ are V-intervals. We then fuse the intervals together to obtain one of interval length less than $|Atoms_V|$ which satisfies $((\$\,T)^* \wedge \mathit{init} \wedge \mathit{finite}); (T \wedge \mathit{empty})$ and hence also satisfies the semantically equivalent finite-time transition configuration. □

The next definition is required for analysing infinite-time transition configurations and makes use of the earlier Definitions 3.4–3.6 concerning conjunctions and Definition 5.1 concerning conditional liveness formulas:

Definition 6.3 (Enabled liveness formulas)**.** *An* enabled liveness formula En *is a conjunction of $|En|$ formulas in which for each $k : 1 \leq k \leq |En|$, the subformula $En[k]$ is of the form $\diamondsuit\, w$, for some state formula w. The state formulas $\theta_{En[1]}, \ldots, \theta_{En[|En|]}$ denote the $|En|$ liveness tests in En so that $En[k]$ and $\diamondsuit\, \theta_{En[k]}$ refer to the same formula.*

For any V-atom α and conditional liveness formula L, we will also define $En_{L,\alpha}$ to be the enabled liveness formula containing exactly the liveness tests in L which are enabled by α (recall Definition 5.1). Let S be the set of indices of L's implications which are enabled by α. Then $En_{L,\alpha}$ is the conjunction $\bigwedge_{j \in S} \diamondsuit\, \theta_{L[j]}$.

For example, suppose V is the set $\{p, q\}$, α is the V-atom $\neg p \wedge q$ and L is the conditional liveness formula $((p \vee \neg q) \supset \diamondsuit \neg p) \wedge (q \supset \diamondsuit(p \equiv \neg q)) \wedge (\mathit{true} \supset \diamondsuit(p \supset q))$ mentioned earlier as formula (1) in Section 5. Then $En_{L,\alpha}$ is the conjunction $\diamondsuit(p \equiv \neg q) \wedge \diamondsuit(p \supset q)$.

Lemma 6.4. *For any V-atom α and conditional liveness formula L in* PTL$_V$*, the conjunctions $\alpha \wedge L$ and $\alpha \wedge En_{L,\alpha}$ are semantically equivalent.*

Not surprisingly, the hardest part of the proof of existence of small models for infinite-time transition configurations involves finding small models for periodic transition configurations. Recall that Lemma 5.26 relates the satisfiability of the periodic transition configuration $\Box T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)$ to that of the PITL$_V$ formula $(\$ T)^* \wedge \alpha \wedge L \wedge \bigcirc \mathit{sfin}\ \alpha$. We will use the equivalence of $\alpha \wedge L$ and $\alpha \wedge \mathit{En}_{L,\alpha}$ to assist in the analysis of bounded models of $(\$ T)^* \wedge \alpha \wedge L \wedge \bigcirc \mathit{sfin}\ \alpha$. These can then be used to obtain a bounded periodic model for the original periodic transition configuration.

Lemma 6.5. *For any V-atom α and conditional liveness formula L in* PTL$_V$*, the following equivalence is valid:*

$$\models \quad (\$ T)^* \wedge \alpha \wedge L \wedge \bigcirc \mathit{sfin}\ \alpha \quad \equiv \quad (\$ T)^* \wedge \alpha \wedge \mathit{En}_{L,\alpha} \wedge \bigcirc \mathit{sfin}\ \alpha.$$

Proof. This readily follows from the earlier Lemma 6.4 concerning the semantic equivalence of the formulas $\alpha \wedge L$ and $\alpha \wedge \mathit{En}_{L,\alpha}$. □

The next Lemma 6.6 shortens the nonempty, finite model expressed by the formula $(\$ T)^* \wedge \alpha \wedge \mathit{En} \wedge \bigcirc \mathit{sfin}\ \alpha$ to one having a bounded length by adapting the technique presented earlier in Lemma 6.1 concerning a bounded model for the formula $(\$ T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta$.

Lemma 6.6. *For any V-atom α and enabled liveness formula En in* PTL$_V$*, if the formula $(\$ T)^* \wedge \alpha \wedge \mathit{En} \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable, then it is satisfied by a interval having interval length at most $(|\mathit{En}| + 1)\,|\mathit{Atoms}_V|$.*

Proof. If the formula $(\$ T)^* \wedge \alpha \wedge \mathit{En} \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable, then by Lemma 5.23 there exists some satisfying V-interval. We can fuse $|\mathit{En}| + 1$ copies of this interval together to obtain a V-interval σ which satisfies the formula $\big((\$ T)^* \wedge \alpha \wedge \mathit{En} \wedge \mathit{finite}\big)^{|\mathit{En}|+1} \wedge \bigcirc \mathit{sfin}\ \alpha$. It is not hard to check than σ itself satisfies the original formula $(\$ T)^* \wedge \alpha \wedge \mathit{En} \wedge \bigcirc \mathit{sfin}\ \alpha$ since each liveness test in En is satisfied somewhere in σ prior to the last state. Furthermore, there exist a sequence of $|\mathit{En}|$ V-atoms $\gamma_1, \ldots, \gamma_{|\mathit{En}|}$ such that for each $j : 1 \leq j \leq |\mathit{En}|$, the state formula $\gamma_j \wedge \theta_{\mathit{En}[j]}$ is satisfied by some state prior to the last one and the V-interval σ satisfies the next formula:

$$\alpha \ \wedge \ \big((\$ T)^*; \gamma_1?; \ldots; (\$ T)^*; \gamma_{|\mathit{En}|}?; (\$ T)^+\big) \ \wedge \ \bigcirc \mathit{sfin}\ \alpha.$$

If a gap between two of the $|\mathit{En}|$ selected states satisfying their respective liveness tests has interval length of at least $|\mathit{Atoms}_V|$, then within the gap, some state occurs twice. Such a gap can then be shortened in the manner of Lemma 6.1. By means of this we obtain from the V-interval σ another V-interval having bounded length and satisfying the formula below:

$$\alpha \ \wedge \ \big((\$ T)^{<|\mathit{Atoms}_V|}; \gamma_1?; \ldots; (\$ T)^{<|\mathit{Atoms}_V|}; \gamma_{|\mathit{En}|}?; (\$ T)^{\leq |\mathit{Atoms}_V|}\big) \\ \wedge \ \bigcirc \mathit{sfin}\ \alpha.$$

The resulting new interval is nonempty and has interval length not exceeding $(|\mathit{En}| + 1)\,|\mathit{Atoms}_V|$. Moreover it still satisfies $(\$ T)^* \wedge \alpha \wedge \mathit{En} \wedge \bigcirc \mathit{sfin}\ \alpha$. □

Lemma 6.7. *If the formula $(\$ T)^* \wedge \alpha \wedge L \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable, then it is satisfiable on a finite, nonempty interval with interval length at most $(|L| + 1)\,|\mathit{Atoms}_V|$.*

Proof. From Lemma 6.6 we have that if the formula $(\$\,T)^* \wedge \alpha \wedge \mathit{En}_{L,\alpha} \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable, then it is satisfiable on a finite, nonempty interval having interval length at most $(|\mathit{En}_{L,\alpha}| + 1)\,|\mathit{Atoms}_V|$. Lemma 6.4 ensures that the conjunctions $\alpha \wedge L$ and $\alpha \wedge \mathit{En}_{L,\alpha}$ are semantically equivalent. In addition, we have $|\mathit{En}_{L,\alpha}| \leq |L|$. Therefore, if the formula $(\$\,T)^* \wedge \alpha \wedge L \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable, then it is satisfiable on a finite, nonempty interval with interval length at most $(|L| + 1)\,|\mathit{Atoms}_V|$. □

Lemma 6.8. *If the periodic transition configuration* $\Box\, T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)$ *is satisfiable, then it is satisfied by a periodic interval with period of interval length at most* $(|L| + 1)\,|\mathit{Atoms}_V|$.

Proof. Lemma 5.26 ensures that if the periodic transition configuration is satisfiable, then the formula $(\$\,T)^* \wedge \alpha \wedge L \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable. By Lemma 6.7, if this is satisfiable, then it has a satisfying interval having interval length at most $(|L| + 1)\,|\mathit{Atoms}_V|$. Lemma 5.23 permits us to assume without loss of generality that the interval is a V-interval. We can then fuse ω copies of it together to obtain a periodic interval which has a period with interval length at most $(|L| + 1)\,|\mathit{Atoms}_V|$ and also satisfies the formula $((\$\,T)^* \wedge \alpha \wedge L)^\omega$. Theorem 5.17 establishes that this formula is equivalent to the original periodic transition configuration. □

Theorem 6.9. *If the infinite-time transition configuration* $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ *is satisfiable, then it is satisfied by an ultimately periodic interval consisting of an initial segment having interval length less than* $|\mathit{Atoms}_V|$ *fused with a periodic interval having a period with interval length of at most* $(|L| + 1)\,|\mathit{Atoms}_V|$.

Proof. If some interval satisfies the formula $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$, then Lemma 5.11 ensures that the interval also satisfies the next semantically equivalent formula:

$$((\$\,T)^* \wedge \mathit{init} \wedge \mathit{finite}); \bigvee\nolimits_{\alpha \in \mathit{Atoms}_V} (\Box\, T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)). \qquad (10)$$

Lemma 5.24 and simple temporal reasoning establish that for some V-atom α the two formulas $(\$\,T)^* \wedge \mathit{init} \wedge \mathit{sfin}\ \alpha$ and $\Box\, T \wedge \alpha \wedge L \wedge \Box \Diamond^+ (\alpha \wedge L)$ are satisfiable. By Lemma 6.1, the first formula is satisfiable in some interval σ having interval length less than $|\mathit{Atoms}_V|$. Lemma 6.8 yields some periodic interval σ' which satisfies the second formula and possesses a period with interval length of at most $(|L| + 1)\,|\mathit{Atoms}_V|$. Lemma 5.23 permits us to assume that σ and σ' are V-intervals. Therefore the last state of σ is the same as the first one of σ' since both states satisfy α. The fusion $\sigma \circ \sigma'$ is itself ultimately periodic and satisfies the formula (10). Hence it also satisfies the semantically equivalent original infinite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ as well. In addition, the interval $\sigma \circ \sigma'$ has an initial segment having interval length less than $|\mathit{Atoms}_V|$ fused with a periodic interval with period of interval length at most $(|L| + 1)\,|\mathit{Atoms}_V|$. □

7 Decomposition of transition configurations

We now prove the two Theorems 7.1 and 7.4 which respectively relate the satisfiability of finite-time and infinite-time transition configurations with simple interval-oriented tests involving finite time. These theorems are later used in Section 8 as part of the justification of the PTL decision procedures and in Section 10 as part of the completeness proof of an axiom system for PTL.

Theorem 7.1 (Decomposing finite-time transition configurations)**.** *The following are equivalent:*

(a) The finite-time configuration $\Box\, T \wedge \mathit{init} \wedge \mathit{finite}$ *is satisfiable.*

(b) For some V-atoms α *and* β*, the three formulas below are satisfiable:*

$$\alpha \wedge \mathit{init} \qquad (\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta \qquad T \wedge \beta \wedge \mathit{empty}.$$

Proof. Theorem 5.10 ensures that the finite-time configuration is semantically equivalent to the next PITL$_V$ formula:

$$((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}); (T \wedge \mathit{empty}).$$

Now simple interval-based reasoning guarantees that this is satisfiable iff for some V-atoms α and β, the next formula is satisfiable:

$$((\$\, T)^* \wedge \alpha \wedge \mathit{init} \wedge \mathit{finite}); (T \wedge \beta \wedge \mathit{empty}).$$

Lemma 5.24 ensures that this is itself satisfiable iff the next two formulas are:

$$(\$\, T)^* \wedge \alpha \wedge \mathit{init} \wedge \mathit{sfin}\ \beta \qquad T \wedge \beta \wedge \mathit{empty}.$$

Finally, simple temporal reasoning ensures that the first of these is itself is satisfiable iff the following two formulas are satisfiable:

$$\alpha \wedge \mathit{init} \qquad (\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta. \qquad \square$$

We now turn to decomposing an infinite-time transition configuration:

Lemma 7.2. *The infinite-time transition configuration* $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ *is satisfiable iff for some V-atoms* α *and* β*, the following formulas are satisfiable:*

$$(\$\, T)^* \wedge \alpha \wedge \mathit{init} \wedge \mathit{sfin}\ \beta \qquad (\$\, T)^* \wedge \beta \wedge \mathit{En}_{L,\beta} \wedge \bigcirc\, \mathit{sfin}\ \beta. \tag{11}$$

Proof. Theorem 5.29 ensures that the infinite-time configuration is satisfiable iff the next PITL$_V$ formula is satisfiable (in some finite-time interval):

$$\big((\$\, T)^* \wedge \mathit{init} \wedge \mathit{finite}\big); \big((\$\, T)^* \wedge L \wedge \mathit{more} \wedge \mathit{finite} \wedge (\vec{V} \leftarrow \vec{V})\big).$$

Simple interval-based temporal reasoning ensures that this itself is satisfiable iff for some V-atoms α and β, next formula is satisfiable:

$$\big((\$\, T)^* \wedge \alpha \wedge \mathit{init} \wedge \mathit{finite}\big); \big((\$\, T)^* \wedge \beta \wedge L \wedge \bigcirc\, \mathit{sfin}\ \beta\big). \tag{12}$$

Now Lemma 6.4 guarantees the semantic equivalence of the conjunctions $\beta \wedge L$ and $\beta \wedge \mathit{En}_{L,\beta}$. We therefore can replace L by $\mathit{En}_{L,\beta}$ in formula (12). Finally, Lemma 5.24 yields that the resulting formula is itself satisfiable iff the two formulas in (11) are satisfiable. $\square$

The next lemma concerning enabled liveness formulas is shortly used in Theorem 7.4 to analyse the satisfiability of infinite-time configurations:

Lemma 7.3. *For any V-atom* α *and enabled liveness formula* En*, the following are equivalent:*

(a) The formula $(\$\,T)^* \wedge \alpha \wedge \mathit{En} \wedge \bigcirc \mathit{sfin}\ \alpha$ *is satisfiable.*

(b) For some $|\mathit{En}|$ *V-atoms* $\gamma_1, \ldots, \gamma_{|\mathit{En}|}$ *(not necessarily distinct), the following are all satisfiable:*

$$(\$\,T)^* \wedge \alpha \wedge \bigcirc \mathit{sfin}\ \alpha$$
$$\text{for each } \gamma_i\text{:}\quad (\$\,T)^* \wedge \alpha \wedge \mathit{sfin}\ \gamma_i \quad \gamma_i \wedge \theta_{\mathit{En}[i]} \quad (\$\,T)^* \wedge \gamma_i \wedge \mathit{sfin}\ \alpha.$$

Proof. Induction on the length of En and simple interval-based reasoning can be used to demonstrate that the formula $(\$\,T)^* \wedge \alpha \wedge \mathit{En} \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable iff the formula $(\$\,T)^* \wedge \alpha \wedge \bigcirc \mathit{sfin}\ \alpha$ is satisfiable and also for some V-atoms $\gamma_1, \ldots, \gamma_{|\mathit{En}|}$, for each γ_i the following formula is satisfiable:

$$(\$\,T)^* \wedge \alpha \wedge \Diamond(\gamma_i \wedge \theta_{\mathit{En}[i]}) \wedge \mathit{sfin}\ \alpha. \tag{13}$$

This guarantees that for each liveness test $\theta_{\mathit{En}[i]}$ in En, the V-atom α can reach some V-atom γ_i which satisfies $\theta_{\mathit{En}[i]}$ and this V-atom γ_i itself can reach back to α. We can re-express (13) as the semantically equivalent formula below:

$$\big((\$\,T)^* \wedge \alpha \wedge \mathit{finite}\big); \big((\$\,T)^* \wedge \gamma_i \wedge \theta_{\mathit{En}[i]} \wedge \mathit{sfin}\ \alpha\big).$$

Lemma 5.24 ensures that this is satisfiable iff the next two formulas are:

$$(\$\,T)^* \wedge \alpha \wedge \mathit{sfin}\ \gamma_i \qquad (\$\,T)^* \wedge \gamma_i \wedge \theta_{\mathit{En}[i]} \wedge \mathit{sfin}\ \alpha.$$

The second one is satisfiable iff the two formulas shown below are satisfiable:

$$\gamma_i \wedge \theta_{\mathit{En}[i]} \qquad (\$\,T)^* \wedge \gamma_i \wedge \mathit{sfin}\ \alpha. \qquad \square$$

Theorem 7.4 (Decomposing infinite-time transition configurations)**.** *The following are equivalent:*

(a) The infinite-time configuration $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ *is satisfiable.*

(b) For some V-atoms α, β *and* $\gamma_1, \ldots, \gamma_{|\mathit{En}_{L,\beta}|}$ *(not necessarily distinct), the following are all satisfiable:*

$$\alpha \wedge \mathit{init} \qquad (\$\,T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta \qquad (\$\,T)^* \wedge \beta \wedge \bigcirc \mathit{sfin}\ \beta$$
$$\text{for each } \gamma_i\text{:}\quad (\$\,T)^* \wedge \beta \wedge \mathit{sfin}\ \gamma_i \quad \gamma_i \wedge \theta(\mathit{En}_{L,\beta}, i) \quad (\$\,T)^* \wedge \gamma_i \wedge \mathit{sfin}\ \beta.$$

Proof. Lemma 7.2 establishes that the infinite-time configuration $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$ is satisfiable iff there exist some V-atoms α and β for which the next two formulas are satisfiable:

$$(\$\,T)^* \wedge \alpha \wedge \mathit{init} \wedge \mathit{sfin}\ \beta \qquad (\$\,T)^* \wedge \beta \wedge \mathit{En}_{L,\beta} \wedge \bigcirc \mathit{sfin}\ \beta. \tag{14}$$

Now simple temporal reasoning ensures that the first of these is itself is satisfiable iff the following two formulas are satisfiable:

$$\alpha \wedge \mathit{init} \qquad (\$\,T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta.$$

Furthermore, Lemma 7.3 guarantees that the second formula in (14) is satisfiable iff the formula $(\$\,T)^* \wedge \beta \wedge \bigcirc \mathit{sfin}\ \beta$ is satisfiable and furthermore for some V-atoms γ_1, $\ldots$, $\gamma_{|\mathit{En}_{L,\beta}|}$ (not necessarily distinct), the following are all satisfiable for each γ_i:

$$(\$\,T)^* \wedge \beta \wedge \mathit{sfin}\ \gamma_i \qquad \gamma_i \wedge \theta(\mathit{En}_{L,\beta}, i) \qquad (\$\,T)^* \wedge \gamma_i \wedge \mathit{sfin}\ \beta.$$

$\square$

Type of transition config.	Max. # of variables to represent a state
Finite-time	$\|V\|$
Infinite-time (see §8.1)	$2\|V\| + \|L\|$
Infinite-time (see §8.2)	$\|V\| + 2\|L\|$
Infinite-time (also in §8.2)	$\|V\| + \|L\| + \lceil \log_2(\|L\| + 1) \rceil$

Table 5: Number of variables used by decision procedures in BDDs

8 Decision procedures

We now describe decision procedures for finite-time and infinite-time transition configurations. They are based on binary decision diagrams (BDDs) [10, 11] which provide an efficient basis for performing many computational tasks involving reductions to reasoning about formulas in propositional logic. The approach taken here demonstrates how to use interval-based techniques to naturally describe and analyse PTL decision procedures. We had little difficulty implementing ones for finite and infinite time using the popular Colorado University Decision Diagram Package (CUDD) [24] developed by Somenzi. Our prototype tool consists of a front-end coded in the CLISP [20] implementation of Common Lisp [1] as well as a back-end coded in Perl [76]. The back-end employs a Perl-oriented interface to CUDD written by Somenzi and called PerlDD [77]. The front-end accepts arbitrary PTL formulas and converts them to transition configurations using methods later described in Sections 11 and 12. The transition configurations are then passed to the back-end which analyses them using BDDs. In this section we describe the basis for performing this analysis.

The remainder of this section assumes that the reader already has some familiarity with BDDs.

For the convenience of readers, Table 5 gives a summary of the maximum number of variables required to represent an individual state in BDDs for the three kinds of decision procedures later discussed as well as for a variant of the third one. We include this table here since it reflects the size of the state space and therefore gives some idea of the relative efficiency of the techniques. For example, the decision procedure for finite time has a state space containing up to $2^{|V|}$ states. In contrast, the first decision procedure for infinite time requires more than twice as many variables and can therefore take significantly more time and space to run. On the other hand, the remaining decision procedures for infinite time only require extra variables in proportion to the number of liveness tests. In each decision procedure, as in typical applications of BDD-based reachability analysis, some of the constructed BDDs represent binary relations between pairs of states and therefore require twice the number of variables.

Our algorithm for finite-time transition configurations adapts methods for *symbolic state space traversal* described by Coudert, Berthet and Madre [21–23] (see also Kropf [54]) for use with BDD-based representations of formulas in propositional logic. Furthermore, it greatly benefits from closely related methods first employed by McMillan in symbolic model checking [14,19,62] which also include the automatic generation of counterexamples for unsatisfiable formulas and, similarly, witnesses for satisfiable ones.

Recall that Theorem 7.1 shows that the finite-time transition configuration $\Box T \wedge init \wedge finite$ is satisfiable iff for some V-atoms α and β, the next three formulas are satisfiable:

$$\alpha \wedge \mathit{init} \qquad (\$\,T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta \qquad T \wedge \beta \wedge \mathit{empty}.$$

We can readily search for suitable V-atoms using BDDs. Three BDDs Γ_1, Γ_2 and Γ_3 are initially constructed. In what follows, please recall the notion $\nvDash X$ introduced in Definition 3.1 to denote that the formula X is satisfiable. We first describe the roles of the BDDs Γ_1, Γ_2 and Γ_3 before actually constructing them:

- The BDD Γ_1 represents the state formula init and hence the set of V-atoms satisfying init (i.e., the set $\{\alpha \in \mathit{Atoms}_V : \alpha \models \mathit{init}\}$). This is the same as the set $\{\alpha \in \mathit{Atoms}_V : \nvDash \alpha \wedge \mathit{init}\}$.

- The second BDD Γ_2 captures all pairs of V-atoms corresponding to unit (i.e., two-state) intervals satisfying T. In other words, it corresponds to the set $\{\langle \alpha, \beta \rangle \in \mathit{Atoms}_V^2 : \alpha\beta \models T\}$. This is the same as the set $\{\langle \alpha, \beta \rangle \in \mathit{Atoms}_V^2 : \nvDash T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta\}$.

  Note that the formula $T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$ can also be expressed using the operator \$ as $\$\,T \wedge \alpha \wedge \mathit{sfin}\ \beta$. Some readers may prefer this second form since it more closely resembles the frequently occurring formula $(\$\,T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta$.

- The third BDD Γ_3 captures the behaviour of T in an empty interval. Therefore Γ_3 represents the set of all V-atoms satisfying the formula $T \wedge \mathit{empty}$ (i.e., the set $\{\beta \in \mathit{Atoms}_V : \beta \models T\}$). This is the same as the set $\{\beta \in \mathit{Atoms}_V : \nvDash T \wedge \beta \wedge \mathit{empty}\}$.

In the course of manipulating the BDDs we make use of two finite sets of propositional variables to represent binary relations between states. They include the original ones (e.g., p, r_1, ..., r_4) as well as primed versions (e.g., p', r_1', ..., r_4'). Therefore, the BDDs contain at most $2|V|$ distinct variables.

For convenience, we often do not distinguish between a BDD and the propositional logic formula it represents.

Let V and V' respectively denote the two sets of variables. We now construct the BDDs Γ_1, Γ_3 and Γ_2 as follows:

- Let Γ_1 be the formula init.

- Obtain Γ_2 from the formula T by replacing all variables in the scope of any $\bigcirc$ constructs by corresponding ones in V' and then deleting all $\bigcirc$ operators (but not the associated operands) to obtain a formula in conventional propositional logic. We refer to this process of constructing Γ_2 from T by the term *flattening*.

- Obtain Γ_3 from the formula T by replacing each $\bigcirc$ construct by false.

The BDDs Γ_1 and Γ_3 both only can contain variables in V whereas Γ_2 can contain variables in V and V'.

Suppose T and init are the following formulas mentioned earlier in Subsection 3.3 (where the variable w is used there in place of init):

$$\begin{aligned} T: \quad & (r_1 \equiv (p \vee \bigcirc r_1)) \ \wedge \ (r_2 \equiv (\neg r_1 \vee \bigcirc r_2)) \\ & \wedge \ (r_3 \equiv (\neg p \vee \bigcirc r_3)) \ \wedge \ (r_4 \equiv (\neg r_3 \vee \bigcirc r_4)) \\ \mathit{init}: \quad & \neg r_2 \wedge \neg r_4. \end{aligned}$$

Here are the associated Γ_1, Γ_2 and Γ_3 for these T and $init$:

$$\begin{aligned}
\Gamma_1 &: \quad \neg r_2 \wedge \neg r_4 \\
\Gamma_2 &: \quad (r_1 \equiv (p \vee r_1')) \ \wedge \ (r_2 \equiv (\neg r_1 \vee r_2')) \\
&\qquad \wedge \ (r_3 \equiv (\neg p \vee r_3')) \ \wedge \ (r_4 \equiv (\neg r_3 \vee r_4')) \\
\Gamma_3 &: \quad (r_1 \equiv (p \vee \mathit{false})) \ \wedge \ (r_2 \equiv (\neg r_1 \vee \mathit{false})) \\
&\qquad \wedge \ (r_3 \equiv (\neg p \vee \mathit{false})) \ \wedge \ (r_4 \equiv (\neg r_3 \vee \mathit{false})).
\end{aligned}$$

The connection between the BDDs for Γ_1 and Γ_3 and the previously mentioned sets of V-atoms they are meant to capture is straightforward. In order to justify the less intuitive relationship between the construction for Γ_2 and the earlier associated set of pairs of V-atoms, we shortly present Lemma 8.2 relating Γ_2 with T. However, the following lemma concerning NL^1 formulas is first given since it is used in the proof of Lemma 8.2.

Lemma 8.1. *The following are equivalent for any* NL^1 *formula* T*:*

(a) The formula T is satisfiable in some nonempty interval.

(b) The formula $\mathit{skip} \wedge T$ is satisfiable.

Proof. $(a) \Rightarrow (b)$*:* Suppose some nonempty interval σ satisfies the formula T. Now σ contains at least two states. Let σ' denote the subinterval consisting the first two states in σ. Now σ' satisfies the formula skip. Furthermore, the formula T is in NL^1. Lemma 5.3 consequently ensures that the interval σ', like σ, satisfies the formula T because both two intervals share the same first two states. Therefore σ' satisfies the formula $\mathit{skip} \wedge T$.

$(b) \Rightarrow (a)$*:* If some interval σ satisfies the PTL formula $\mathit{skip} \wedge T$, then σ is clearly nonempty and also satisfies T. □

Lemma 8.2. *For any V-atoms α and β, the following are equivalent:*

(a) The formula $T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$ is satisfiable (i.e., $\alpha\beta \models T$).

(b) The propositional logic formula $\Gamma_2 \wedge \alpha \wedge \beta_V^{V'}$ is satisfiable.

Proof. $(a) \Rightarrow (b)$*:* Suppose the formula $T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$ is satisfiable. Then the flattening of T into Γ_2 readily yields that the formula $\Gamma_2 \wedge \alpha \wedge \beta_V^{V'}$ is satisfiable.

$(b) \Rightarrow (a)$*:* If the propositional logic formula $\Gamma_2 \wedge \alpha \wedge \beta_V^{V'}$ is satisfiable, then the flattening of $\bigcirc$ constructs in Γ_2 readily yields that the NL^1 formula $T \wedge \alpha \wedge \bigcirc\beta$ is satisfiable. Clearly any interval satisfying it has at least two states. Hence by the previous Lemma 8.1 the formula $\mathit{skip} \wedge T \wedge \alpha \wedge \bigcirc\beta$ is satisfiable. Simple temporal reasoning then ensures that the semantically equivalent formula $T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$ is also satisfiable. □

We use Γ_2 together with the first BDD Γ_1 to iteratively calculate a sequence of BDDs $\Delta_0, \ldots, \Delta_k, \ldots$ so that for any k, Δ_k describes all V-atoms which can be reached from one which satisfies $init$ in exactly k steps. In other words, Δ_k represents the following set:

$$\{\, \beta \in \mathit{Atoms}_V : \text{for some } \alpha \in \mathit{Atoms}_V, \dashv (\$\, T)^k \wedge \alpha \wedge \mathit{init} \wedge \mathit{sfin}\ \beta \,\}.$$

We set Δ_0 to be Γ_1. Therefore, every variable in Δ_0 is in V. Each Δ_{k+1} is calculated to be semantically equivalent to the next quantified propositional logic formula in which renaming ensures that all free variables are in V:

$$\big(\exists V.\,(\Delta_k \wedge \Gamma_2)\big)^{V}_{V'}. \tag{15}$$

Because of the final renaming, the sole variables left in the BDD Δ_{k+1} are elements of V. The only BDD operations required to calculate Δ_{k+1} from (15) are logical-and, existential quantification (which actually yields a BDD representing a semantically equivalent quantifier-free formula) and renaming which are all standard ones.

Remark 8.3. *Within the CUDD system, the entire calculation for obtaining* $\exists V.\,(\Delta_k \wedge \Gamma_2)$ *can even be done by a single CUDD operation tailored to handle this specific kind of common BDD manipulation. Furthermore, the renaming of variables in* V' *to those in* V *is actually achieved by taking the BDD obtained for* $\exists V.\,(\Delta_k \wedge \Gamma_2)$ *and then performing a single CUDD operation which yields another BDD in which the variables in* V *are swapped with the corresponding ones in* V'*.*

For any given Δ_k which has been calculated, we next determine the logical-and of it with Γ_3 (i.e., the BDD $\Delta_k \wedge \Gamma_3$) and then proceed as follows:

1. If the logical-and is not false, then there is some V-atom β satisfying $T \wedge \mathit{empty}$ which can be reached in k steps from a V-atom α satisfying init. Therefore the next three formulas are all satisfiable:

   $$\alpha \wedge \mathit{init} \qquad (\$\,T)^k \wedge \alpha \wedge \mathit{sfin}\ \beta \qquad T \wedge \beta \wedge \mathit{empty}.$$

   Now the second formula ensures the satisfiability of the formula $(\$\,T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta$. Therefore Theorem 7.1 can be invoked to obtain the satisfiability of the original finite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \mathit{finite}$. We therefore do not need to calculate any further Δ_k's.

2. Otherwise, the logical-and is false so we must continue to iterate.

During the iteration process, we maintain a BDD representing the set of all V-atoms so far reachable from one satisfying init. This BDD corresponds to the formula $\bigvee_{0 \le i \le k} \Delta_i$ which equals the next set:

$$\{\, \beta \in \mathit{Atoms}_V : \text{for some } \alpha \in \mathit{Atoms}_V, \dashv\ (\$\,T)^{\le k} \wedge \alpha \wedge \mathit{init} \wedge \mathit{sfin}\ \beta \,\}.$$

If no such β exists which also satisfies $T \wedge \mathit{empty}$, the BDD eventually converges to a value corresponding to the set of all V-atoms reachable from V-atoms which satisfy init. The following set denotes this:

$$\{\, \beta \in \mathit{Atoms}_V : \text{ for some } \alpha \in \mathit{Atoms}_V, \dashv\ (\$\,T)^* \wedge \alpha \wedge \mathit{init} \wedge \mathit{sfin}\ \beta \,\}. \tag{16}$$

We then terminate the algorithm with a report that the original transition configuration $\Box\, T \wedge \mathit{init} \wedge \mathit{finite}$ is unsatisfiable. Even though Theorem 6.2 bounds the number of iterations, in some cases convergence takes too long. This necessitates a preset iteration limit or a facility for manual intervention in order to force premature termination of the loop. It can also happen that memory is exhausted before termination occurs.

If for some n, the algorithm succeeds after n iterations and determines that the transition configuration is satisfiable, then a sample V-interval having $n+1$ states and

which satisfies the formula can be calculated. This involves standard BDD methods for constructing such examples, also referred to as *witnesses*, and is done by working backward through the BDDs Δ_n, Δ_{n-1}, $\ldots \Delta_0$ to find a suitable sequence of $n+1$ V-atoms to serve as a V-interval satisfying the transition configuration. The algorithm can be also readily adapted to only determine values for a subset of the variables in V.

8.1 Dealing with infinite time

For testing an infinite-time transition configuration $\Box\, T \wedge \mathit{init} \wedge \Box \Diamond^+ L$, we can make use of Theorem 5.29 which guarantees that this formula is satisfiable iff the next PTL_V formula is satisfiable:

$$\boxplus\, T \wedge \mathit{init} \wedge \Diamond(L \wedge \mathit{finite} \wedge \mathit{more} \wedge (\vec{V} \leftarrow \vec{V})). \tag{17}$$

The previously described satisfiability algorithm for finite-time can therefore be utilized. However, we must first transform this second formula to some suitable finite-time transition configuration using techniques later described in Sections 11 and 12 for reducing arbitrary PTL formulas to them.

Quite sophisticated and efficient algorithms can presumably be employed to analyse the infinite-time transition configuration by adapting existing BDD-based techniques such as those we will later mention in Subsection 8.2 when we consider our second decision procedure for infinite time. However, we first present a method which is relatively straightforward to describe and which we implemented with not much more difficulty than for the version for finite-time transition configurations. Within this approach for infinite time, once a formula is determined to be satisfiable, it is relatively easy to determine a witness of it. The decision procedure for finite time can even be used to naturally justify some of the interval-based reasoning involved. However, a major computational disadvantage of this particular decision procedure for infinite time is that it requires the introduction of a significant number of extra variables, namely, $|V| + |L|$. Therefore the total number of variables used to represent a single state is $2|V| + |L|$, and the number to represent a binary relation between pairs of states is double this. Thus at least twice as many variables are required as for the decision procedure for finite time. On the other hand, the second, more intricate decision procedure later presented in Subsection 8.2 requires at most $2|L|$ extra variables and possibly fewer. Consequently, this second infinite-time decision procedure appears to lend itself to more efficient implementations, although we ourselves have not yet carried out one.

The basic idea in the first, simpler decision procedure for infinite time is to use BDDs to solve for atoms α and β which ensure that the following three formulas are satisfied:

$$\alpha \wedge \mathit{init} \qquad (\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta \qquad (\$\, T)^+ \wedge \beta \wedge L \wedge (\vec{V} \leftarrow \vec{V}). \tag{18}$$

The third formula guarantees that the atom β has some associated periodic transition configuration (see the earlier Lemma 5.26). Note the use of chop-plus here in $(\$\, T)^+$ to force a nonempty interval. As in the case for finite-time, we first calculate a BDD which denotes the previously described set (16) of all atoms reachable in 0 or more steps via T from one satisfying the formula init. Let us call this BDD Δ'.

The next step is to search for some element β of Δ' with an associated periodic transition configuration. This involves finding a nonempty finite interval satisfying the third formula in (18) containing as conjuncts both L and $\vec{V} \leftarrow \vec{V}$. However these conjuncts cannot be directly used for reachability analysis since neither is expressed by means of

an accessibility relation between adjacent states. We remedy this by constructing two suitable groups of formulas. Each contains three formulas and does permit the required reachability analysis:

- The first group consists of the state formula $init'$, the transition configuration T' and the state formula w' and ensures that the formula $\vec{V} \leftarrow \vec{V}$ holds.
- The second group is similarly made up of the state formula $init''$, the transition configuration T'' and the state formula w'' and ensures that L is true.

The details of constructions will be given shortly. The techniques for handling $\vec{V} \leftarrow \vec{V}$ are somewhat easier to understand and we have therefore associated them with the first of the two groups of formulas. After obtaining both groups, we can search for a suitable atom β by doing the following:

1. Use BDDs to calculate all atoms in the state formula $\Delta' \wedge init' \wedge init''$.
2. Next, determine the BDD equalling the set of all atoms reachable by one or more steps from any of these via the transition configuration $T \wedge T' \wedge T''$.
3. Finally, logically-and the resulting BDD with the state formula $w' \wedge w''$ to obtain the set of all atoms which can serve as β.

Let us now look in more detail at the construction of each group of three formulas and how they work.

Construction of first group of formulas $init'$, T' and w' The first group of three formulas must ensure that the following implication concerning finite time is valid:

$$\models \quad (\$\, T')^* \ \wedge \ init' \ \wedge \ sfin\ w' \quad \supset \quad \vec{V} \leftarrow \vec{V}. \tag{19}$$

Without loss of generality, assume that the variables in the set V are $p_1, \ldots, p_{|V|}$. Let $q_1, \ldots, q_{|V|}$ be $|V|$ distinct propositional variables not occurring in V. Then let $init'$, T' and w' be the following formulas:

$$\begin{array}{ll} init': & q_1 \equiv p_1 \quad \wedge \quad \cdots \quad \wedge \quad q_{|V|} \equiv p_{|V|} \\ T': & (\bigcirc\, q_1) \equiv q_1 \quad \wedge \quad \cdots \quad \wedge \quad (\bigcirc\, q_{|V|}) \equiv q_{|V|} \\ w': & \text{same as } init'. \end{array}$$

The purpose of each q_i is to record the initial value of the corresponding p_i so as to ensure that the initial and final values the p_i are equal. It is not hard to see that the use of the extra propositional variables $q_1, \ldots, q_{|V|}$ together with the use of the formulas $init'$, T' and w' force each p_i's initial and final values to be equal. The desired behaviour for any pair of variables p_i and q_i is captured by the valid implication now given which contains just two propositional variables p and q:

$$\models \quad q \equiv p \ \wedge \ \boxdot((\bigcirc\, q) \equiv q) \ \wedge \ sfin\,(q \equiv p) \quad \supset \quad p \leftarrow p.$$

Note that the instance $p \leftarrow p$ of the binary *temporal assignment* operator $\leftarrow$ is, following the definition earlier in Table 1, the same as the PTL formula $finite \ \supset \ (fin\ p) \equiv p$. It tests that p's initial and final values are equal in finite intervals. Consequently, the implication (19) involving the variables in V and $q_1, \ldots, q_{|V|}$ is indeed valid. In fact, for any given triple of T', $init'$ and w' we can employ the PTL decision procedure for

finite time (described later in Section 12) to check validity of the next formula which is semantically equivalent to (19) in finite intervals:

$$\models \quad \boxed{\text{m}}\, T' \ \wedge\ \mathit{init}' \ \wedge\ \mathit{sfin}\ w' \quad \supset \quad \vec{V} \leftarrow \vec{V}.$$

Here we make use of Lemma 5.4 concerning the semantical equivalence of the formulas $\boxed{\text{m}}\, T$ and $(\$\, T)^*$.

It is straightforward to add a kind of existential quantification to PTL and PITL to hide one or more variables. In the case, of PTL, the resulting logic is called commonly referred to as *Quantified PTL* (QPTL) or alternatively as *Quantified Propositional Linear-Time Temporal Logic* (QPLTL) (see, e.g., [25, 51, 58]). For any interval σ, propositional variable p and formula A, we say that σ satisfies the formula $\exists p.\, A$ iff there exists some interval σ' which is identical to σ in its length and assignments to variables, except that the values in σ' for p can be different. If we use existential quantification around the lefthand formula in the valid implication (19) to hide the variables $q_1, \ldots, q_{|V|}$, then the resulting formula (itself in quantified PITL) and the one on the righthand side of the implication are in fact semantically equivalent in finite intervals:

$$\models \quad \mathit{finite} \quad \supset \quad \Big(\exists q_1 \ldots q_{|V|}.\, \big((\$\, T')^* \wedge \mathit{init}' \wedge \mathit{sfin}\ w'\big)\Big) \ \equiv\ \vec{V} \leftarrow \vec{V}. \qquad (20)$$

Construction of second group of formulas $init''$, T'' and w'' The second group must ensure the implication below concerning finite time is valid:

$$\models \quad (\$\, T'')^* \ \wedge\ \mathit{init}'' \ \wedge\ \mathit{sfin}\ w'' \quad \supset \quad L. \qquad (21)$$

Now let $u_1, \ldots, u_{|L|}$ be $|V|$ distinct propositional variables not occurring in V nor amongst $q_1, \ldots, q_{|V|}$. Here are the definitions of $init''$, T'' and w'':

$$\begin{array}{lllllll}
\mathit{init}'': & u_1 \equiv \neg\eta_{L[1]} & \wedge & \cdots & \wedge & u_{|L|} \equiv \neg\eta_{L[|L|]} \\
T'': & (\bigcirc u_1) \equiv (u_1 \vee \theta_{L[1]}) & \wedge & \cdots & \wedge & (\bigcirc u_{|L|}) \equiv (u_{|L|} \vee \theta_{L[|L|]}) \\
w'': & u_1 & \wedge & \cdots & \wedge & u_{|L|}.
\end{array}$$

The purpose of each u_i is to guarantee that the liveness requirements imposed by the conditional liveness formula L's conjunct $\eta_{L[i]} \supset \diamondsuit\, \theta_{L[i]}$ are fulfilled. Every u_i is initially set to the value of $\neg\eta_{L[i]}$. Subsequently, the values of the state formulas u_i and $\theta_{L[i]}$ in a state uniquely determine the value of each u_i in the next state. We now show that the temporal assignment $u_i \leftarrow (\eta_{L[i]} \supset \diamondsuit\, \theta_{L[i]})$ is true for the overall interval. There are two cases to consider:

- If the enabling test $\eta_{L[i]}$ is initially false, then u_i is initially set to true and remains so for the rest of the interval. In this case we have $u_i \leftarrow \mathit{true}$. Hence, we also trivially have $u_i \leftarrow (\eta_{L[i]} \supset \diamondsuit\, \theta_{L[i]})$ since $\neg\eta_{L[i]}$ is initially true and hence so is the implication $\eta_{L[i]} \supset \diamondsuit\, \theta_{L[i]}$.
- Otherwise, if $\eta_{L[i]}$ is initially true, then u_i is initially set to false and is true in the last state iff the liveness test $\theta_{L[i]}$ is true sometime strictly before the last state. In this case we have $u_i \leftarrow \diamondsuit\, \theta_{L[i]}$, and consequently also $u_i \leftarrow (\eta_{L[i]} \supset \diamondsuit\, \theta_{L[i]})$ since $\eta_{L[i]}$ is initially true.

The next valid implication combines the two cases:

$$\models \quad \mathit{finite} \ \wedge\ (u_i \equiv \neg\eta_{L[i]}) \ \wedge\ \boxed{\text{m}}\big((\bigcirc u_i) \equiv (u_i \vee \theta_{L[i]})\big) \quad \supset \quad u_i \leftarrow (\eta_{L[i]} \supset \diamondsuit\, \theta_{L[i]}).$$

It is natural to view this implication as a substitution instance of a simpler PTL formula which is itself valid and has only three propositional variables p_1, p_2 and p_3. Here p_1 represents $\eta_{L[i]}$, p_2 represents $\theta_{L[i]}$ and p_3 represents u_i:

$$\models \quad \mathit{finite} \ \wedge \ (p_3 \equiv \neg p_1) \ \wedge \ \boxdot\big((\bigcirc p_3) \equiv (p_3 \vee p_2)\big) \quad \supset \quad p_3 \leftarrow (p_1 \supset \diamondsuit_{\mathrm{m}} p_2).$$

One benefit of this formula is that it can be readily tested for validity using the decision procedure for finite time extended to handle full PTL as described later in Section 12. We can also employ the decision procedure to check the validity of (21) for any given instances of L, T'', init'' and w''. Lemma 5.4, which deals with the semantical equivalence of the formulas $\boxdot T$ and $(\$\, T)^*$, permits us to express (21) in PTL as follows:

$$\models \quad \boxdot T'' \ \wedge \ \mathit{init}'' \ \wedge \ \mathit{sfin}\ w'' \quad \supset \quad L.$$

Below is a formula which uses existential quantification to hide $u_1, \ldots, u_{|L|}$ and thereby expresses the equivalence of the two sides in finite intervals.

$$\models \quad \mathit{finite} \quad \supset \quad \Big(\exists u_1 \ldots u_{|L|}.\, \big((\$\, T'')^* \wedge \mathit{init}'' \wedge \mathit{sfin}\ w''\big)\Big) \ \equiv \ L. \qquad (22)$$

This is similar to the earlier implication (20) concerning $\vec{V} \leftarrow \vec{V}$.

We can now regard the BDD Δ', which represents the set (16), as a state formula and locate suitable periodic transition configurations by calculating the set of all atoms reachable from Δ' in one or more steps using the binary relation over atoms associated with the NL[1] formula $T \wedge T' \wedge T''$. This is done in a similar manner as before. Next, the resulting BDD is logically and-ed with the test $w' \wedge w''$. Let the BDD Δ'' denote the resulting set of atoms. An atom β is consequently in Δ'' iff the following formula is satisfiable:

$$(\$(T \wedge T' \wedge T''))^+ \wedge \Delta' \wedge \mathit{init}' \wedge \mathit{init}'' \wedge \mathit{sfin}\,(\beta \wedge w' \wedge w''). \qquad (23)$$

The reasoning so far given ensures that this is satisfiable iff the original conjunction associated with a periodic transition configuration (i.e., the third formula in (18)) is itself satisfiable. If we use existential quantification to hide the variables $q_1, \ldots, q_{|V|}$ and $u_1, \ldots, u_{|L|}$ in the formula (23) (as in formulas (20) and (22)) and we also omit the atom β, then the resulting formula is semantically equivalent to the one shown below:

$$(\$\, T)^+ \wedge \Delta' \wedge (\vec{V} \leftarrow \vec{V}) \wedge L.$$

Furthermore, the BDD Δ'' can be seen to be the set of all atoms β in Δ' for which the following formula is satisfiable in finite time.

$$(\$\, T)^+ \wedge \beta \wedge L \wedge (\vec{V} \leftarrow \vec{V}).$$

Hence Δ'' does not equal false iff the three formulas in (18) are all satisfiable. As we mentioned earlier, they in turn are satisfiable iff the original infinite-time transition configuration is satisfiable.

8.2 A decision procedure for infinite time motivated by automata

The BDD-based decision procedure for infinite-time just presented has been successfully used on a range of simple examples. However, a different and presumably more efficient decision procedure for infinite-time transition configurations can be developed

which reflects the connection between PTL formulas and the important class of finite-state automata over infinite words known as nondeterministic Büchi automata [12] (see also Thomas [86, 87]). They are like conventional nondeterministic finite-state automata but have a set of accepting states with a different kind of acceptance condition. An infinite word is accepted iff there exists a run for it in which some element of the set of accepting states occurs infinitely often. The link between Büchi automata and temporal logic was originally observed by Vardi, Wolper and Sistla [88, 89, 96].

Our second decision procedure for infinite-time transition configurations can alternatively be obtained by an analysis which totally avoids reference to Büchi automata. We instead construct from an infinite-time transition configuration the previously discussed formula (17) which can be tested for satisfiability in finite-time intervals. However, rather than using this formula itself, we transform it into another one which has a simpler kind of conditional liveness formula. The result of the transformation can be checked for finite-time satisfiability.

The decision procedure has the advantage over the previous one for infinite time in Subsection 8.1 of only requiring for the representation of states at most $2|L|$ extra variables (i.e., $|V| + 2|L|$ variables in total), rather than $|V| + |L|$ extra variables. It therefore is potentially much more efficient to execute. However, it is more complicated to explain and implement and we do not yet have a working version. The earlier decision procedures for finite and infinite time can be used to check some of the steps in the construction as we later show.

Because of the popularity of Büchi automata, in most of the presentation of our decision procedure we will use PTL formulas which have a close structural similarity to them. Nevertheless, unlike the work of Vardi, Wolper and Sistla, an understanding of the role of the corresponding PTL formulas does not at all require the formal introduction of such automata. Rather, we can simply view our approach as converting an infinite-time transition configuration into another in which the conditional liveness formula is expressible as a state formula. Such transition configurations are easier to analyse. Consequently, we omit a formal definition of Büchi automata here. Later on, Remark 8.4 outlines the alternative approach which totally avoids Büchi automata.

The presentation of this decision procedure can be skipped without a loss of continuity in the rest of this work.

Let us now consider the construction in more detail. We can take an infinite-time transition configuration with transition formula T, initial condition $init$ and conditional liveness formula L and transform it into another transition configuration closely related to conventional nondeterministic Büchi automata:

$$\Box(T \wedge T_L) \ \wedge \ (init \wedge init_L) \ \wedge \ \Box \Diamond^+ w_L. \tag{24}$$

Here the transition formula T_L and state formulas $init_L$ and w_L are specially constructed from the conditional liveness formula L and do not depend on T and $init$. This is not an actual finite-state automaton since there is no division of variables into those which serve as the automaton state and those which serve as the input. Although it is possible to existentially quantify over some of the variables to sharpen the automata-theoretic connection, we will not do this here.

Note that strictly speaking the new transition configuration (24) is not a well-formed infinite-time transition configuration because the state formula w_L is itself not a well-formed conditional liveness formula (recall Definition 5.1). However, w_L is semantically equivalent to the simple conditional liveness formula $\neg w_L \ \supset \ \Diamond\!\!\!\!-\!\!\!\!\cdot \ false$. This equivalence can be justified as a substitution instance of the next PTL equivalence

which is itself valid:

$$\models \quad p \equiv (\neg p \supset \diamondsuit_{\text{fin}} false). \tag{25}$$

The validity of the equivalence (25) uses the fact that the subformula $\diamondsuit_{\text{fin}} false$ (the same as $\diamondsuit(more \wedge false)$ by the definition of $\diamondsuit_{\text{fin}}$ in Table 1) is itself semantically equivalent to $\diamondsuit false$ and hence also to $false$. This is combined together with the simple propositional tautology $p \equiv (\neg p \supset false)$. We can check the validity of (25) by using the two decision procedures already presented for finite and infinite time and extending them to handle arbitrary PTL formulas as later shown in Section 12.

It seems reasonable to call any transition configuration such as (24) a *Büchi transition configuration* if its rightmost conjunct is a PTL formula of the form $\Box \diamondsuit^+ w$, for some state formula w since $\Box \diamondsuit^+ w$ is true on an infinite-time interval iff some atom in $Atoms_V$ which satisfies w occurs infinitely often in the interval. This in a sense mimics the standard Büchi acceptance condition which, as already noted, requires that some element of the set of accepting states occurs infinitely many times for an automaton run to be considered an accepting run. Sophisticated techniques applicable to BDDs such as those of Emerson and Lei [26] for the μ-calculus (see also Clarke et al. [19]) and more recent ones by Hardin et al. [38], Xie and Beere [97] and Bloem, Gabow and Somenzi [8] for analysing strongly connected components could presumably be applied to test for the satisfiability of the Büchi transition configuration (24) by adapting methods for checking for the emptiness of Büchi-automata and constructing witnesses when appropriate.

The technique we later describe for actually obtaining the Büchi transition configuration (24) is comparable to the automata-theoretic method of Vardi, Wolper and Sistla [88, 89, 96] which combines what they refer to as a *local* Büchi automaton for checking just the behaviour of adjacent states with another *eventuality* Büchi automaton which checks for liveness requirements. We can regard within our framework the original infinite-time transition configuration's subformula $\boxed{\text{fin}}\, T \wedge init$ as being the analogue of a local Büchi automaton. Similarly the automaton-like construction later obtained by us to simulate the formula $\Box \diamondsuit^+ L$ corresponds to an eventuality Büchi automaton.

A simplified illustration of the construction The technique employed here can be demonstrated by means of a simplified example which focuses on the key ideas. We will encode the infinite-time behaviour of the PTL formula $\Box(\diamondsuit p_1 \wedge \diamondsuit p_2 \wedge \diamondsuit p_3)$ within another one which only contains a single $\diamondsuit$ and makes use of three extra propositional variables u'_1, u'_2 and u'_3. Let T' be the transition configuration given below:

$$\begin{aligned} T'\colon \quad & (\bigcirc u'_1) \equiv \big(if\ (u'_1 \wedge u'_2 \wedge u'_3)\ then\ p_1\ else\ (u'_1 \vee p_1)\big) \\ \wedge \quad & (\bigcirc u'_2) \equiv \big(if\ (u'_1 \wedge u'_2 \wedge u'_3)\ then\ p_2\ else\ (u'_2 \vee p_2)\big) \\ \wedge \quad & (\bigcirc u'_3) \equiv \big(if\ (u'_1 \wedge u'_2 \wedge u'_3)\ then\ p_3\ else\ (u'_3 \vee p_3)\big). \end{aligned}$$

Each u'_i tracks the corresponding p_i and records whether it becomes true. As with u_i in the previous Subsection 8.1, the values of the variables in any state uniquely determine the value of each u'_i in the next state. Whenever p_1, p_2 and p_3 have all become true, all three variables u'_1, u'_2 and u'_3 are simultaneously true and are then reinitialized to track another potential set of occurrences of p_1, p_2 and p_3.

The following valid implication concerning finite, nonempty intervals holds for

each $i\colon 1 \leq i \leq 3$ and illustrates how u'_i tracks the behaviour of p_i:

$$\models \quad \boxed{\text{m}}\, T' \ \wedge\ u'_1 \ \wedge\ u'_2 \ \wedge\ u'_3 \ \wedge\ \bigcirc \boxed{\text{m}}\, \neg(u'_1 \wedge u'_2 \wedge u'_3) \supset \quad \boxed{\text{i}}(\mathit{more} \ \supset\ u'_i \leftarrow \diamondsuit_{\text{m}}\, p_i)). \quad (26)$$

This uses the PITL operator $\boxed{\text{i}}$ defined in Table 2 to test all initial subintervals. Here is a variant of this in PTL which does not contain $\boxed{\text{i}}$:

$$\models \quad \boxed{\text{m}}\, T' \ \wedge\ u'_1 \ \wedge\ u'_2 \ \wedge\ u'_3 \ \wedge\ \bigcirc \boxed{\text{m}}\, \neg(u'_1 \wedge u'_2 \wedge u'_3) \quad \supset \quad u'_i \leftarrow \diamondsuit_{\text{m}}\, p_i. \quad (27)$$

The decision procedure for finite time extended to handle full PTL (see later in Section 12) can be used to confirm the implication's validity.

The valid formula below concerns one cycle of p_1, p_2 and p_3 being true:

$$\models \quad \boxed{\text{m}}\, T' \ \wedge\ u'_1 \ \wedge\ u'_2 \ \wedge\ u'_3 \ \wedge\ \mathit{more} \supset \quad \big(\diamondsuit^+(u'_1 \wedge u'_2 \wedge u'_3) \ \equiv\ (\diamondsuit_{\text{m}}\, p_1 \wedge \diamondsuit_{\text{m}}\, p_2 \wedge \diamondsuit_{\text{m}}\, p_3)\big). \quad (28)$$

It shows how the testing for the three $\diamondsuit_{\text{m}}$-formulas involving p_1, p_2 and p_3 can be simultaneously carried out by means of a single $\diamondsuit^+$ formula containing a conjunction of the variables u'_1, u'_2 and u'_3. We can use the previous two decision procedures for finite time and infinite time extended to handle full PTL to check the validity of (28).

It can be established from this and induction over time that the next implication is valid:

$$\models \quad \mathit{inf} \ \wedge\ \boxed{\text{m}}\, T' \ \wedge\ u'_1 \ \wedge\ u'_2 \ \wedge\ u'_3 \supset \quad (\Box\diamondsuit^+(u'_1 \wedge u'_2 \wedge u'_3) \ \equiv\ \Box(\diamondsuit\, p_1 \wedge \diamondsuit\, p_2 \wedge \diamondsuit\, p_3)). \quad (29)$$

The first decision procedure for infinite time, when extended to handle full PTL, can be used to confirm the validity of this.

Construction of the Büchi transition configuration Let us now turn to constructing the formulas T_L, init_L and w_L. This is more complicated than in the example because a conditional liveness formula L contains enabling tests $\eta_{L[1]}, \ldots, \eta_{L[|L|]}$ which can alternatively enable and disable the corresponding liveness tests $\theta_{L[1]}, \ldots, \theta_{L[|L|]}$.

Suppose $r'_1, \ldots, r'_{|L|}$ and $u'_1, \ldots, u'_{|L|}$ are $2|L|$ propositional variables not occurring in V. Here are the roles played by each of these groups of variables:

- The variables $r'_1, \ldots, r'_{|L|}$ can be seen as guessing in the initial state some subset of the enabling tests $\eta_{L[1]}, \ldots, \eta_{L[|L|]}$ in the conditional liveness formula L. From then on, each r'_i remains stable. This permits us to subsequently check whether there are infinitely many states in which exactly these particular enabling tests are simultaneously true (see below the second half of definition of w_L) and whether each corresponding $\theta_{L[i]}$ is itself infinitely often true (as tracked by the associated u'_i).

- The variables $u'_1, \ldots, u'_{|L|}$ are used to ensure that the subset of liveness tests $\theta_{L[1]}, \ldots, \theta_{L[|L|]}$ corresponding to the selected subset of enabling tests $\eta_{L[1]}, \ldots, \eta_{L[|L|]}$ are each infinitely often true. This does not necessarily occur in one state so a more complicated tracking system is required than the one which monitors $\eta_{L[i]}$ for each $i\colon 1 \leq i \leq |L|$.

With this in mind, we now define the state formulas $init_L$ and w_L and the transitional configuration T_L:

$$
\begin{array}{ll}
init_L: & \bigwedge_{1 \leq i \leq |L|} u'_i \\
w_L: & init_L \quad \wedge \quad \bigwedge_{1 \leq i \leq |L|} (r'_i \equiv \eta_{L[i]}) \\
T_L: & (\bigcirc r'_1) \equiv r'_1 \quad \wedge \quad \cdots \quad \wedge \quad (\bigcirc r'_{|L|}) \equiv r'_{|L|} \\
& \quad \wedge \quad (\bigcirc u'_1) \equiv \big(\mathit{if}\ w_L\ \mathit{then}\ (\neg r'_1 \vee \theta_{L[1]})\ \mathit{else}\ (u'_1 \vee \theta_{L[1]})\big) \\
& \quad \wedge \quad \cdots \\
& \quad \wedge \quad (\bigcirc u'_{|L|}) \equiv \big(\mathit{if}\ w_L\ \mathit{then}\ (\neg r'_{|L|} \vee \theta_{L[|L|]})\ \mathit{else}\ (u'_{|L|} \vee \theta_{L[|L|]})\big).
\end{array}
$$

Here is a description of each of these:

- The initial condition $init_L$ sets each u'_i to true to naturally force a resetting at the beginning of the tracking of the selected subset of the liveness tests $\theta_{L[1]}$, $\ldots$, $\theta_{L[|L|]}$. We do not similarly initialize each r'_i. Instead, when we test the Büchi transition configuration for satisfiability, all possible combinations of values for $r'_1, \ldots, r'_{|L|}$ are automatically considered. As long as at least one suitable selection exists, the transition configuration is satisfiable and hence the original infinite-time one is as well.

- The test w_L serves as the new conditional liveness formula and assists in capturing the behaviour of the original conditional liveness formula L. It thereby ensures that there are infinitely many states in which the values of $r'_1, \ldots, r'_{|L|}$ agree in value with the respective values of $\eta_{L[1]}, \ldots, \eta_{L[|L|]}$ and that at these times each u'_i is true, thereby ensuring that each selected $\theta_{L[i]}$ is infinitely often true. It plays the role which the conjunction $u'_1 \wedge u'_2 \wedge u'_3$ does in the simplified example already presented but needs to also deal with conditional nature of the enabling tests $\eta_{L[1]}, \ldots, \eta_{L[|L|]}$.

- The transition formula T_L ensures that the variables $r'_1, \ldots, r'_{|L|}$ do not change over time. It also controls the variables $u'_1, \ldots, u'_{|L|}$. Each u'_i tracks the liveness requirements imposed by the conditional liveness formula L's conjunct $\eta_{L[i]} \supset \diamondsuit \theta_{L[i]}$. Whenever the acceptance test w_L is true, the tracking is reset. By the definition of w_L, this is the same as testing for $u'_1, \ldots, u'_{|L|}$ all simultaneously being true and in addition each r'_i equalling $\eta_{L[i]}$. At this time a kind of reinitialization of $u'_1, \ldots, u'_{|L|}$ is performed in order to start the tracking of another cycle of liveness tests. More precisely, at this time each u'_i is set to true if either r'_i is false or $\theta_{L[i]}$ are true (i.e., $\neg r'_i \vee \theta_{L[i]}$ or equivalently $r'_i \supset \theta_{L[i]}$) and set to false otherwise. At all other times, u'_i only changes from false to true if $\theta_{L[i]}$ becomes true for the first time in the current tracking cycle.

The following valid implication concerning finite, nonempty intervals holds for each $i\colon 1 \leq i \leq |L|$ and illustrates how u'_i tracks the behaviour of $\theta_{L[i]}$. It corresponds to the valid implication (26) in our simplified example.

$$\models \quad \boxdot T_L \ \wedge \ init_L \ \wedge \ \bigcirc \boxdot \neg w_L \quad \supset \quad \boxdot(\mathit{more} \ \supset \ u'_i \leftarrow (r'_i \supset \diamondsuit \theta_{L[i]})).$$

Now w_L itself implies $init_L$ since $init_L$ is itself a conjunct in w_L. Therefore if a finite nonempty interval satisfies $\boxdot T_L$ and has its first state satisfying w_L, but no others, except for possibly the last one, then the final value of u'_i agrees with the value of the implication $r'_i \supset \diamondsuit \theta_{L[i]}$ on the interval:

$$\models \quad \boxdot T_L \ \wedge \ w_L \ \wedge \ \bigcirc \boxdot \neg w_L \quad \supset \quad u'_i \leftarrow (r'_i \supset \diamondsuit \theta_{L[i]}).$$

This PTL formula is analogous to the earlier valid formula (27) in our illustrative example.

The valid formula below is comparable to the earlier valid implication (28):

$$\models \quad \boxdot_{\mathrm{m}} T_L \ \wedge \ init_L \ \wedge \ more \quad \supset \quad \big((\diamond^+ w_L) \ \equiv \ L\big).$$

By using this valid implication and induction over time, we can obtain the validity of the next formula concerning the formulas L, T_L, $init_L$ and w_L. It permits us to replace $\Box \diamond^+ L$ with the structurally simpler formula $\Box \diamond^+ w_L$ and corresponds to the valid implication (29) in our example:

$$\models \quad \boxdot_{\mathrm{m}} T_L \ \wedge \ init_L \quad \supset \quad (\Box \diamond^+ w_L \ \equiv \ \Box \diamond^+ L).$$

The construction of the formulas T_L, $init_L$ and w_L ensures that the original infinite-time transition configuration is satisfiable iff the new Büchi one (24) is and any interval satisfying the Büchi one also satisfies the original one. The Büchi transition configuration is itself satisfiable iff the following PTL formula is satisfiable in finite time:

$$\boxdot_{\mathrm{m}}(T \wedge T_L) \wedge (init \wedge init_L) \wedge \diamond(w_L \wedge \mathit{finite} \wedge more \wedge (\vec{V}' \leftarrow \vec{V}')),$$

where V' is the set of all variables in V together with the $2|L|$ new ones. The proof of this is just an application of Theorem 5.29 where, as already noted, we re-express w_L as the well-formed conditional liveness formula $\neg w_L \ \supset \ \diamondsuit_{\mathrm{tp}} \mathit{false}$.

The behaviour imposed by T_L is only really relevant in the righthand subformula contained within the $\diamond$ operator. Consequently, it suffices to test for satisfiability of the variant formula given below which can omit the formula $init_L$:

$$\boxdot_{\mathrm{m}} T \wedge init \wedge \diamond\big(\boxdot_{\mathrm{m}} T_L \wedge w_L \wedge \mathit{finite} \wedge more \wedge (\vec{V}' \leftarrow \vec{V}')\big). \tag{30}$$

We can similarly omit $init_L$ in the Büchi transition configuration (24) for the same reason.

As mentioned earlier, at most $2|L|$ extra variables are needed. It is sometimes possible that fewer suffice. If any $\eta_{L[i]}$ happens to be the formula $true$ or a formula which $\boxdot_{\mathrm{m}} T$ ensures never changes its value within an interval, then we do not need the corresponding r'_i and can instead simply use $\eta_{L[i]}$ itself. The equivalence $r'_i \equiv \eta_{L[i]}$ associated with r'_i can be omitted from w_L and similarly the equivalence $(\bigcirc r'_i) \equiv r'_i$ can be deleted from T_L. Finally, the implication $r'_i \supset \theta_{L[i]}$ in T_L can be replaced by the implication $\eta_{L[i]} \supset \theta_{L[i]}$. Therefore it is possible that strictly less than $2|L|$ extra variables might suffice. However, at least $|L|$ are necessary since we still need all of the $|L|$ variables $u'_1, \ldots, u'_{|L|}$ to track the behaviour of liveness tests $\theta_{L[1]}, \ldots, \theta_{L[|L|]}$ unless some $\eta_{L[i]}$ is false in which case the behaviour of $\theta_{L[i]}$ is irrelevant. One way to establish this is by symbolically examining the conjunction $\boxdot_{\mathrm{m}} T \wedge init$ to determine whether it forces $\eta_{L[i]}$ to be false.

A further reduction in variables is possible by monitoring the liveness tests $\theta_{L[1]}$, $\ldots$, $\theta_{L[|L|]}$ using a counter ranging over the values 0, 1, $\ldots$, $|L|$, inclusive, instead of the $|L|$ variables u'_1, $\ldots$, $u'_{|L|}$. This counter only requires $\lceil \log_2(|L| + 1) \rceil$ variables which serve as bits to represent its values. It can be initialized to any value less than or equal to $|L|$. The behaviour in subsequent states is as follows:

- When the counter equals some value k less than $|L|$, there are two possibilities:
  - If the liveness test $\theta_{L[k+1]}$ is true or r'_{k+1} is false (i.e., the implication $r'_{k+1} \supset \theta_{L[k+1]}$ is true), then counter's value in the next state is $k + 1$.

Axioms:

N1 (K).	$\vdash \circledcirc(X \supset X') \supset \circledcirc X \supset \circledcirc X'$
N2 ($\mathrm{D_c}$).	$\vdash \bigcirc X \supset \circledcirc X$

Inference rules:

NR1.	If X is a tautology, then $\vdash X$
NR2 (MP).	If $\vdash X \supset X'$ and $\vdash X$, then $\vdash X'$
NR3 (RN).	If $\vdash X$, then $\vdash \circledcirc X$

Table 6: Complete axiom system for NL (Modal system $\mathrm{K{+}D_c}$)

– Otherwise the counter's value in the next state remains k.

- Whenever the counter equals $|L|$ it is set to 0 in the next state.

The Büchi acceptance test checks that the counter equals $|L|$ infinitely often. The number of extra variables needed to represent a single state is at most $|L| + \lceil \log_2(|L| + 1) \rceil$ so at most $|V| + |L| + \lceil \log_2(|L| + 1) \rceil$ variables are needed in total. If all of the enabling tests $\eta_{L[1]}, \ldots, \eta_{L[|L|]}$ are vacuously true or formulas which $\boxdot T$ ensures never change value, then even just $\lceil \log_2(|L| + 1) \rceil$ new variables suffice.

Remark 8.4. *One interesting aspect of our framework stems from the observation already mentioned that it is not strictly necessary to make reference to Büchi automata and use the associated Büchi transition configurations in the justification of the second infinite-time decision procedure. This is because the basic insights of the decision procedure can be largely considered within the context of finite time by viewing it as transforming one formula concerning finite time into another. More specifically, as we already noted at the beginning of Subsection 8.1 in regard to the first decision procedure for infinite time, the original infinite-time transition configuration for T is satisfiable iff the next* PTL *formula (i.e., the earlier formula* (17)*) is in finite time:*

$$\boxdot T \wedge \mathit{init} \wedge \Diamond(L \wedge \mathit{finite} \wedge \mathit{more} \wedge (\vec{V} \leftarrow \vec{V})).$$

We then employ the formulas T_L and w_L to obtain the formula (30) *which is satisfiable iff the previous one is. Any model of* (30) *can be used for the original formula* (17) *as well. The methods previously cited from the literature as being suitable for BDD-based reachability analysis of Büchi automata can presumably be adapted for use here.*

9 Axiom system for NL

In preparation for the proof of axiomatic completeness for PTL, we now consider an axiom system for NL. The axiomatic completeness of NL later plays a major role in the completeness proof for PTL.

Within this section, the variables X, X', X_0 and X'_0 denote NL *formulas.*

Table 6 contains a complete axiom system for NL adapted from the modal logic $\mathrm{K{+}D_c}$. Here $\circledcirc$ (weak next), previously defined in Table 1 to be a derived operator, is instead regarded as a primitive construct. We can consider $\bigcirc X$ to be an abbreviation for $\neg \circledcirc \neg X$. Hughes and Cresswell [46, Problem 6.8 on p. 123 with solution on p. 379] briefly discuss how to show deductive completeness of the logic $\mathrm{K{+}D_c}$.

Table 7 contains a complete axiom system for NL in which $\bigcirc$, rather than $\circledcirc$, is the primitive operator. Consequently, $\circledcirc$ is derived in the manner already shown in Table 1.

Axioms:

N1′ (N$\Diamond$).	$\vdash \neg \bigcirc \mathit{false}$
N2′ (C$\Diamond$).	$\vdash \bigcirc(X \vee X') \supset \bigcirc X \vee \bigcirc X'$
N3′ ($\mathrm{D_c}$).	$\vdash \bigcirc X \supset \textcircled{w} X$

Inference rules:

NR1′.	If X is a tautology, then $\vdash X$
NR2′ (MP).	If $\vdash X \supset X'$ and $\vdash X$, then $\vdash X'$
NR3′ (RM$\Diamond$).	If $\vdash X \supset X'$, then $\vdash \bigcirc X \supset \bigcirc X'$

Table 7: Alternative complete axiom system for NL based on $\bigcirc$

The axiom system is essentially one of several M-based axiomatizations of normal systems of modal logic covered by Chellas [17] with the addition of the axiom $\mathrm{D_c}$. This second axiom system appears preferable for our purposes since our definition of PTL also takes $\bigcirc$ to be primitive. We therefore use this axiom system here although the methods employed can be easily adapted to the first NL axiom system.

Definition 9.1 (Theoremhood and consistency for NL)**.** *If some* NL *formula X is deducible from the axiom system, we call it an* NL theorem *and denote this theoremhood as $\vdash_{\mathrm{NL}} X$. We define X to be* NL-consistent *if $\neg X$ is* not *an* NL *theorem, i.e., $\nvdash_{\mathrm{NL}} \neg X$.*

Below are some representative lemmas about satisfiability and consistency of NL formulas. They are subsequently used in the completeness proof for the NL axiom system in Table 7.

Lemma 9.2. *For any state formula w and* NL *formula X, if w is satisfiable, then the* NL *conjunction $w \wedge \neg \bigcirc X$ is satisfied by some one-state interval.*

Lemma 9.3. *For any state formula w and* NL *formula X, if both w and X are satisfiable, then so is the formula $w \wedge \bigcirc X$.*
In such as case, if X itself is satisfied by an interval having at most n states, then $w \wedge \bigcirc X$ is satisfied by an interval having at most $n + 1$ states,

Lemma 9.4. *For any* NL *formula X, if $\bigcirc X$ is* NL*-consistent, then so X.*

For any NL formulas X and X', the following are deducible as NL theorems and shortly used to combine two (possibly negated) $\bigcirc$-formulas into one:

$$\vdash_{\mathrm{NL}} \bigcirc(X \wedge X') \equiv \bigcirc X \wedge \bigcirc X' \tag{31}$$

$$\vdash_{\mathrm{NL}} \bigcirc(X \wedge \neg X') \equiv \bigcirc X \wedge \neg \bigcirc X' \tag{32}$$

$$\vdash_{\mathrm{NL}} \neg \bigcirc(X \vee X') \equiv \neg \bigcirc X \wedge \neg \bigcirc X' \ . \tag{33}$$

Axiomatic completeness is usually defined to mean that every valid formula is deducible as a theorem. However, we will make use of the following variant way of expressing completeness:

Lemma 9.5 (Alternative notion of completeness)**.** *A logic's axiom system is complete iff each consistent formula is satisfiable.*

Theorem 9.6 (Completeness of alternative NL axiom system)**.** *The* NL *axiom system in Table 7 is complete.*

Axioms:	Inference rules:
T1. $\vdash \Box(X \supset Y) \supset \Box X \supset \Box Y$	R1. If X is a tautology, then $\vdash X$
T2. $\vdash \bigcirc X \supset \textcircled{w} X$	R2. If $\vdash X \supset Y$ and $\vdash X$, then $\vdash Y$
T3. $\vdash \bigcirc(X \supset Y) \supset \bigcirc X \supset \bigcirc Y$	R3. If $\vdash X$, then $\vdash \Box X$
T4. $\vdash \Box X \supset X \wedge \textcircled{w} \Box X$	
T5. $\vdash \Box(X \supset \textcircled{w} X) \supset X \supset \Box X$	

Table 8: Modified version of Pnueli's complete PTL axiom system DX

Proof. The proof involves the kind of consistency-based reasoning also found in later sections when we hierarchically establish axiomatic completeness for PTL. Using Lemma 9.5, we show that any NL formula X_0 which is NL-consistent (i.e., $\nvdash_{\mathrm{NL}} \neg X_0$) has a satisfying finite interval. Let n be the next-height of X_0, i.e., the maximum nesting of $\bigcirc$s in X_0.

We now do induction on n to show that X_0 is satisfied by some interval with at most $n+1$ states. If $n = 0$, then X_0 is in PROP and hence satisfied by some one-state interval since X_0's consistency ensures that $\neg X_0$ cannot be a tautology (see Inference Rule $\mathrm{NR1}'$). For $n > 0$, we regard the temporal constructs in X_0 which are not nested in other temporal constructs as being primitive. Then conventional propositional reasoning yields a deducibly equivalent formula in disjunctive normal form. As least one disjunct is consistent. Equivalences (31)–(33) are invoked to obtain from such a disjunct a deducibly equivalent NL formula Y with the same next-height and having the form $w \wedge \bigcirc X$ or $w \wedge \neg \bigcirc X$. Induction on n and Lemmas 9.2–9.4 then together ensure that Y is satisfiable. Hence X_0 is as well. □

10 Axiomatic completeness for transition configurations

We now describe a PTL axiom system and then prove axiomatic completeness for transition configurations. Later Sections 11 and 12 hierarchically extend axiomatic completeness to all of PTL.

The PTL axiom system used here is shown in Table 8 and is adapted from another similar PTL axiom system DX proposed by Pnueli [78]. Gabbay et al. [33] showed that DX is complete. Pnueli's original system uses strict versions of $\Diamond$ and $\Box$ (which we respectively denote as $\Diamond^+$ and $\Box^+$ in the earlier Table 1) which do not examine the current state. In addition, Pnueli's system only deals with infinite time. Gabbay et al. [33] also include a variant system called $\mathrm{D^0X}$ based on the conventional $\Diamond$ and $\Box$ operators which examine the current state. The version presented here does this as well and furthermore permits both finite and infinite time.

Definition 10.1 (Theoremhood and consistency for PTL)**.** *If the* PTL *formula X is deducible from the axiom system, we call it a* PTL theorem *and denote this theoremhood as $\vdash X$. We define X to be* consistent *if $\neg X$ is* not *a theorem, i.e., $\nvdash \neg X$.*

In the course of proving completeness for PTL we make use of a definition of completeness for sets of formulas such as sets of transitions configurations:

Definition 10.2 (Completeness for a set of formulas)**.** *An axiom system is said to be complete for a set of formulas if every consistent formula in the set is also satisfiable.*

Lemma	Summary
10.4	If $\dashv \boxdot_{\mathrm{m}} T \wedge \alpha \wedge \bigcirc \beta$, then $\Rightarrow T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$
10.6	If $\dashv \boxdot_{\mathrm{m}} T \wedge \alpha \wedge \Diamond \beta$, then $\Rightarrow (\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta$
10.7	If $\dashv \boxdot_{\mathrm{m}} T \wedge \alpha \wedge \Diamond^+ \beta$, then $\Rightarrow (\$\, T)^* \wedge \alpha \wedge \bigcirc \mathit{sfin}\ \beta$

Table 9: Summary of some basic lemmas for consistency and satisfiability

Now the Alternative Notion of Completeness (Lemma 9.5) can also be readily adapted to sets of formulas. Indeed, our goal in the rest of this section is to show that any consistent transition configuration is also satisfiable.

The next lemma permits us to utilize within PTL the axiomatic completeness of the NL proof system:

Theorem 10.3 (Completeness for NL in PTL)**.** *The* PTL *axiom system is complete for the set of* NL *formulas.*

Proof. Theorem 9.6 establishes the completeness of the alternative NL axiom system in Table 7. We then show that any NL theorem is also a PTL theorem. This can be done by demonstrating that all axioms and inferences rules in the NL axiom system are derivable from PTL ones. □

10.1 Some basic lemmas for completeness

In this subsection, we deal with another part of the completeness proof. We utilize ways to go from certain specific kinds of consistent formulas involving reachability to intervals in order to later construct models for consistent transition configurations in Subsection 10.2. Table 9 summarizes the basic lemmas proved here. Within the table, we use the notation $\Rightarrow X$ already introduced in Definition 3.1 to denote that the formula X is satisfiable and $\dashv X$ to denote that X is consistent.

Lemma 10.4. *For any V-atoms α and β, if the formula $\boxdot_{\mathrm{m}} T \wedge \alpha \wedge \bigcirc \beta$ is consistent, then the formula $T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$ is satisfiable.*

Proof. From the consistency of the formula $\boxdot_{\mathrm{m}} T \wedge \alpha \wedge \bigcirc \beta$ and simple temporal reasoning, we obtain the consistency of the NL^1_V formula $T \wedge \alpha \wedge \bigcirc \beta$. Theorem 10.3 concerning axiomatic completeness for NL formulas in the PTL axiom system then ensures that this is satisfiable. Clearly any interval satisfying it has at least two states. Hence by the earlier Lemma 8.1 the formula $\mathit{skip} \wedge T \wedge \alpha \wedge \bigcirc \beta$ is also satisfiable. Consequently, simple temporal reasoning yields that the semantically equivalent formula $T \wedge \alpha \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$ is satisfiable as well. □

For any V-atom α, within the next two lemmas we let S_α denote the subset of Atoms_V containing exactly every V-atom γ for which the following formula, which concerns reachability from α, is satisfiable:

$$(\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \gamma.$$

Here is a more formal definition of S_α:

$$S_\alpha \stackrel{\mathrm{def}}{=} \{\, \gamma \in \mathit{Atoms}_V : \Rightarrow (\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \gamma \,\}.$$

Lemma 10.5. *For any V-atom α, the following formula is a* PTL *theorem:*

$$\vdash \quad \boxed{\mathrm{m}}\, T \wedge \alpha \;\supset\; \Box \bigvee_{\gamma \in S_\alpha} \gamma. \tag{34}$$

Proof. The following formulas are valid and in NL^1. Hence, they are theorems by the completeness of the PTL axiom system for NL^1 formulas (Theorem 10.3):

$$\vdash \quad \alpha \;\supset\; \bigvee_{\gamma \in S_\alpha} \gamma \qquad\qquad \vdash \quad \mathit{more} \wedge T \wedge \bigvee_{\gamma \in S_\alpha} \gamma \quad \supset \quad \bigcirc \bigvee_{\gamma \in S_\alpha} \gamma.$$

From these and simple temporal reasoning we can readily deduce our goal (34). □

Lemma 10.6. *For any V-atoms α and β, if the formula $\boxed{\mathrm{m}}\, T \wedge \alpha \wedge \Diamond \beta$ is consistent, then the formula $(\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta$ is satisfiable.*

Proof. Suppose on the contrary that $(\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \beta$ is unsatisfiable. Now α is in the set S_α, whereas β is not. Hence, the following formula concerning β not being in S_α is valid and thus a propositional tautology:

$$\vdash \quad \bigvee_{\gamma \in S_\alpha} \gamma \quad \supset \quad \neg\beta. \tag{35}$$

Furthermore, the previous Lemma 10.5 ensures that the next implication is a PTL theorem:

$$\vdash \quad \boxed{\mathrm{m}}\, T \wedge \alpha \;\supset\; \Box \bigvee_{\gamma \in S_\alpha} \gamma. \tag{36}$$

The two implications (35) and (36) together with some simple temporal reasoning let us deduce that α can never reach β:

$$\vdash \quad \boxed{\mathrm{m}}\, T \wedge \alpha \;\supset\; \Box\neg\beta.$$

From this and the general equivalence $\vdash \Box\neg\beta \equiv \neg\Diamond\beta$ we can deduce the following PTL theorem:

$$\vdash \quad \boxed{\mathrm{m}}\, T \wedge \alpha \;\supset\; \neg\Diamond\beta.$$

Therefore, the formula $\boxed{\mathrm{m}}\, T \wedge \alpha \wedge \Diamond\beta$ is inconsistent. This contradicts the lemma's assumption. □

Lemma 10.7. *For any V-atoms α and β, if the formula $\boxed{\mathrm{m}}\, T \wedge \alpha \wedge \Diamond^+ \beta$ is consistent, then the formula $(\$\, T)^* \wedge \alpha \wedge \bigcirc \mathit{sfin}\ \beta$ is satisfiable.*

Proof. From the consistency of the formula $\boxed{\mathrm{m}}\, T \wedge \alpha \wedge \Diamond^+ \beta$, we readily deduce for some *V*-atom γ the consistency of the two PTL$_V$ formulas below:

$$\boxed{\mathrm{m}}\, T \wedge \alpha \wedge \Diamond\gamma \qquad \boxed{\mathrm{m}}\, T \wedge \gamma \wedge \bigcirc\beta.$$

The consistency of the first formula $\boxed{\mathrm{m}}\, T \wedge \alpha \wedge \Diamond\gamma$ and Lemma 10.6 yield that the formula $(\$\, T)^* \wedge \alpha \wedge \mathit{sfin}\ \gamma$ is satisfiable. Lemma 10.4 and the second formula $\boxed{\mathrm{m}}\, T \wedge \gamma \wedge \bigcirc\beta$ then guarantee that the formula $T \wedge \gamma \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta$ is satisfiable. Lemma 5.24 then yields that the next formula is satisfiable:

$$\big((\$\, T)^* \wedge \alpha \wedge \mathit{finite}\big); (T \wedge \gamma \wedge \mathit{skip} \wedge \mathit{sfin}\ \beta).$$

From this and some further simple interval-based reasoning we can establish our goal, namely, that the formula $(\$\, T)^* \wedge \alpha \wedge \bigcirc \mathit{sfin}\ \beta$ is satisfiable. □

10.2 Completeness for transition configurations

We now apply the material presented in the previous Subsection 10.1 to ultimately establish completeness for finite- and infinite-time transition configurations. Here is a summary of the completeness theorems for them:

Type of transition configuration	Where proved
Finite-time	Theorem 10.8
Infinite-time	Theorem 10.9

The remaining two kinds of transition configurations are subordinate to these. For the sake of brevity, we do not consider them here.

Theorem 10.8. *Completeness holds for any finite-time transition configuration* $\Box\, T \wedge init \wedge finite$.

Proof. From the consistency of the finite-time transition configuration $\Box\, T \wedge init \wedge finite$ and simple temporal reasoning we can demonstrate that for some V-atoms α and β, the next formula is consistent:

$$\boxdot\, T \wedge \alpha \wedge init \wedge sfin\,(T \wedge \beta).$$

From this and further simple temporal reasoning it is readily follows that the following formulas are all consistent:

$$\alpha \wedge init \qquad \boxdot\, T \wedge \alpha \wedge \Diamond\, \beta \qquad T \wedge \beta \wedge empty.$$

The first of these is itself satisfiable since any consistent formula in PROP is satisfiable. The second one and Lemma 10.6 yields that the PITL formula $(\$\, T)^* \wedge \alpha \wedge sfin\ \beta$ is satisfiable. The third formula $T \wedge \beta \wedge empty$ is in NL[1] and hence by Theorem 10.3 satisfiable. Hence the following formulas are all satisfiable:

$$\alpha \wedge init \qquad (\$\, T)^* \wedge \alpha \wedge sfin\ \beta \qquad T \wedge \beta \wedge empty.$$

This and Theorem 7.1 then yield the satisfiability of the finite-time transition configuration $\Box\, T \wedge init \wedge finite$. □

Theorem 10.9. *Completeness holds for any infinite-time transition configuration* $\Box\, T \wedge init \wedge \Box \Diamond^+ L$.

Proof. From the consistency of the infinite-time transition configuration $\Box\, T \wedge init \wedge \Box \Diamond^+ L$ and simple temporal reasoning we can demonstrate that for some V-atoms α and β, the next formula is consistent:

$$\boxdot\, T \wedge \alpha \wedge init \wedge \Box \Diamond^+ (\beta \wedge L). \tag{37}$$

Lemma 6.4 ensures that the formulas $\beta \wedge L$ and $\beta \wedge En_{L,\beta}$ are semantically equivalent. The proof of this only requires simple propositional reasoning not involving the temporal operators in L. Hence the next equivalence is readily deducible as a PTL theorem using substitution into a propositional tautology (see Definition 3.3 and PTL inference rule R1 in Table 8):

$$\vdash \quad \beta \wedge L \;\equiv\; \beta \wedge En_{L,\beta}. \tag{38}$$

From the consistency of formula (37) and the deducibility of formula (38), we can show the consistency of the next formula:

$$\boxdot T \wedge \alpha \wedge \mathit{init} \wedge \Box \Diamond^{+}(\beta \wedge En_{L,\beta}).$$

This and simple temporal reasoning then together yield the consistency of the following formulas involving some additional V-atoms $\gamma_1, \ldots, \gamma_{|En_{L,\beta}|}$ (not necessarily distinct):

$$\begin{array}{lllll} \alpha \wedge \mathit{init} & \boxdot T \wedge \alpha \wedge \Diamond \beta & \boxdot T \wedge \beta \wedge \Diamond^{+} \beta & & \\ \text{for each } \gamma_i\text{:} & \boxdot T \wedge \beta \wedge \Diamond \gamma_i & \gamma_i \wedge \theta(En_{L,\beta}, i) & \boxdot T \wedge \gamma_i \wedge \Diamond \beta. \end{array}$$

The consistency of the propositional formulas $\alpha \wedge \mathit{init}$ and $\gamma_i \wedge \theta(En_{L,\beta}, i)$ for each V-atom γ_i ensures they are satisfiable. Lemma 10.6 is then applied to the remaining consistent formulas, except for $\boxdot T \wedge \beta \wedge \Diamond^{+} \beta$ which requires Lemma 10.7. The combined result is that the following formulas are all satisfiable:

$$\begin{array}{lllll} \alpha \wedge \mathit{init} & (\$\, T)^{*} \wedge \alpha \wedge \mathit{sfin}\ \beta & (\$\, T)^{*} \wedge \beta \wedge \bigcirc \mathit{sfin}\ \beta & & \\ \text{for each } \gamma_i\text{:} & (\$\, T)^{*} \wedge \beta \wedge \mathit{sfin}\ \gamma_i & \gamma_i \wedge \theta(En_{L,\beta}, i) & (\$\, T)^{*} \wedge \gamma_i \wedge \mathit{sfin}\ \beta. \end{array}$$

Hence by Theorem 7.4, the original consistent infinite-time transition configuration is indeed satisfiable. □

11 Invariants and related formulas

We will shortly introduce the concepts of invariants and invariant configurations which together act as a natural middle level between transition configurations and full PTL and involve the use of auxiliary variables. These variables provide a way to reduce the nesting of temporal operators within other temporal operators and thereby simplify further analysis. Satisfiability, existence of small models, decidability and axiomatic completeness for invariant configurations can be readily related to the analysis of transition configurations. Furthermore, it is not hard to reduce arbitrary PTL formulas to invariant configurations by utilising such auxiliary variables.

The analysis of invariant configurations and arbitrary PTL formulas does not require any further interval-based reasoning or PITL.

Definition 11.1 (Invariants and dependencies)**.** *An* invariant *is any finite conjunction of zero or more equivalences in which each equivalence's left side is a distinct propositional variable and each equivalence's right side is one of the following:*

- *Some* PTL *formula of the form* $\Diamond w$*, for some state formula* w*.*
- *Some* NL[1] *formula.*

The variables occurring on the left sides of equivalences are called dependent variables *and any other variables are called* independent variables*. The right sides are called* dependent formulas *and each equivalence is itself called a* dependency*. Hence for a given invariant* I*, it follows that* $|I|$ *denotes the number of dependencies in* I*. Also, for any* $k : 1 \leq k \leq |I|$*,* $I[k]$ *denote the* k*-th dependency in* I*. Each dependency containing* $\Diamond$ *is referred to as a* $\Diamond$-dependency.

Below is a sample invariant referred to as I_1:

$$I_1: \quad r_1 \equiv \Diamond(p \wedge \neg q) \quad \wedge \quad r_2 \equiv (r_1 \wedge \bigcirc r_2). \tag{39}$$

Here $|I_1|$ equals 2, the first dependency $I_1[1]$ is the equivalence $r_1 \equiv \Diamond(p \wedge \neg q)$ and the second dependency $I_1[2]$ is the equivalence $r_2 \equiv (r_1 \wedge \bigcirc r_2)$. A dependent variable can be referenced in any dependent formula including the one associated with it. The dependent variable r_2 in the invariant I_1 illustrates this.

Note that an invariant is not necessarily satisfiable as in $r_1 \equiv \neg r_1$. Also note that dependencies of the two forms $r \equiv w$ and $r \equiv \bigcirc w$, for some propositional variable r and state formula w, are both subsumed by the second case in Definition 11.1. If desired, a more restrictive definition of invariants limited to dependencies of the form w, $\bigcirc w$ and $\Diamond w$ is possible.

We can view an invariant I as being any conjunction of the form $\bigwedge_{k:1\leq k\leq |I|}(u_k \equiv \phi_k)$ so that u_k is the k-th dependent variable and ϕ_k is the k-th dependent formula in I. Observe that for any $k : 1 \leq k \leq |I|$, the conjunct $I[k]$ has the form $u_k \equiv \phi_k$ and I itself can be expressed as $\bigwedge_{k:1\leq k\leq |I|} I[k]$.

Starting with an invariant I, we analyse certain low-level formulas referred to here as *invariant configurations*.

Definition 11.2 (Invariant configurations)**.** *An* invariant configuration *is a formula of the form* $\Box\, I \ \wedge \ X$ *where the* PTL *formula* X *is in one of three categories shown below:*

Type of invariant configuration	*Syntax of* X
Basic	w
Finite-time	$w \wedge \mathit{finite}$
Infinite-time	$w \wedge \mathit{inf}$

Here w *is a state formula.*

For example, the conjunction $\Box\, I_1 \ \wedge \ r_2$ is a basic invariant configuration which is true for intervals which are infinite, have r_1 and r_2 always true and p and $\neg q$ both always eventually true.

Let us now introduce some simple notation needed for reasoning about liveness and $\Diamond$-dependencies. This will be used in the definition of an invariant's associated conditional liveness formula.

Definition 11.3 (Liveness tests of an invariant)**.** *For any invariant* I *and any* $k : 1 \leq k \leq |I|$*, if the dependency* $I[k]$ *is a* $\Diamond$*-dependency, define the* liveness test $\theta_{I[k]}$ *to be the state formula which is the operand of the* $\Diamond$*-construct in* $I[k]$.

For instance, the sample invariant I_1 in (39) contains exactly one $\Diamond$-dependency $I_1[1]$. Therefore I_1 has a single liveness test $\theta_{I_1[1]}$ which denotes the formula $p \wedge \neg q$. Observe that for any invariant I, if the dependency $I[k]$ is a $\Diamond$-dependency, then it has the form $u_k \equiv \Diamond\, w$, where u_k is I's k-th dependent variable, and therefore $I[k]$ can also be expressed as $u_k \equiv \Diamond\, \theta_{I[k]}$.

The next definition of a restricted kind of invariant helps to simplify the notation used in the reduction of invariant configurations to transition configurations:

Definition 11.4 (Ordered invariants)**.** *An invariant is said to be* ordered *if all of its* $\Diamond$*-dependencies precede any others.*

The sample invariant I_1 in (39) is itself such an ordered invariant. It is not hard to rearrange an arbitrary invariant's dependencies to obtain a semantically equivalent ordered invariant. In the rest of this section, we will without loss of generality limit our attention to ordered invariants and invariant configurations based on them.

We now associate with an ordered invariant I a transition formula T_I and a conditional liveness formula L_I. They serve to expeditiously reduce invariant configurations to transition configurations previously analysed in earlier sections. Definition 11.5 below describes T_I. The subsequent Definition 11.6 describes the form of L_I.

Definition 11.5 (Transition formula for an ordered invariant)**.** *For an ordered invariant I, the associated transition formula T_I is an* NL^1 *formula which captures I's transitional behaviour between pairs of adjacent states. It is obtained from I by replacing each $\Diamond$-dependency with another dependency not containing $\Diamond$ and leaving the remaining $\Diamond$-free dependencies unchanged. More precisely, each dependency in I of the form $r \equiv \Diamond w$, for some propositional variable r and state formula w, is replaced by the $\Diamond$-free equivalence $r \equiv (w \vee \bigcirc r)$.*

Observe that the transition formula T_I is in NL^1 and is also a well-formed invariant. Also, for any $k : 1 \leq k \leq |I|$, if the dependency $I[k]$ does not contain $\Diamond$, then it and T_I's corresponding dependency $T_I[k]$ are identical.

Here is the transition formula T_{I_1} associated with the sample invariant I_1 in (39):

$$T_{I_1}: \quad r_1 \equiv ((p \wedge \neg q) \vee \bigcirc r_1) \quad \wedge \quad r_2 \equiv (r_1 \wedge \bigcirc r_2).$$

Given an ordered invariant I, we now associate a specific conditional liveness formula L_I with it:

Definition 11.6 (Conditional liveness formula of an ordered invariant)**.** *For any ordered invariant I having exactly n $\Diamond$-dependencies for some n, let the conditional liveness formula L_I be a conjunction of n implications now described. For each $k : 1 \leq k \leq n$, the k-th implication is obtained by simply replacing the outermost equivalence operator in I's k-th $\Diamond$-dependency by the implication operator and using $\diamondsuit$ instead of $\Diamond$. Therefore, for each $k : 1 \leq k \leq n$, the dependency $I[k]$ has the form $u_k \equiv \Diamond \theta_{I[k]}$ and the implication $L_I[k]$ has the form $u_k \supset \diamondsuit \theta_{I[k]}$.*

The definition of I's conditional liveness formula L_I intentionally ignores any NL^1 dependencies in I since T_I already adequately deals with them. As a result, L_I can contain fewer conjuncts than I and T_I. Below is the conditional liveness formula L_{I_1} associated with ordered invariant I_1 in (39):

$$L_{I_1}: \quad r_1 \supset \diamondsuit(p \wedge \neg q).$$

It is not hard to see that, unlike I's transition formula T_I, the conditional liveness formula L_I associated with I is not a well-formed invariant, in part because the main operator in each conjunct of L_I is $\supset$ rather than $\equiv$.

Let us define a convenient notion for measuring how many symbols there are in a formula:

Definition 11.7 (Formula size)**.** *For any formula, the number of symbols in it, excluding parentheses, is called its* formula size*.*

For simplicity when determining formula size, we will regard all conventional propositional operators such as $\wedge$, $\supset$ and $\equiv$ as being primitives. For example, the formula size of each of the two formulas $p \vee ((\bigcirc q) \wedge r)$ and $(p \equiv q) \wedge \bigcirc r$ is 6.

It also seems reasonable to regard the operator $\diamondsuit$ in conditional liveness formulas as a primitive when calculating formula size since the decision procedures for infinite time treat it as such. Alternatively, one can expand it using its definition in Table 1. This requires two extra symbols. Another possibility is to use $\Diamond$ instead, as discussed after Definition 5.1 of conditional liveness formulas.

One reason we ignore parentheses in formula size is that they are not relevant to the kind of internal data representations such as trees and BDDs normally encountered in implementations. In addition, for the sake of readability, many of our sample transition formulas, invariants and other formulas often use spacing instead of parentheses. This makes any consistent counting of bracket symbols more difficult.

The formula size of a conjunction C such as any invariant and conditional liveness formula should not be confused with its size as a conjunction. The later was previously defined in Definition 3.4 to be the number of conjuncts and is denoted as $|C|$. For example, the conjunction $p \wedge q$ has formula size 3, whereas its size $|p \wedge q|$ as a conjunction is 2.

The formula size of an invariant I's associated transition formula T_I and conditional liveness formula L_I are related to the formula size of the invariant but are also affected by any $\Diamond$-dependencies in the invariant. The next lemma makes this more precise:

Lemma 11.8 (Upper bounds on formula size of T_I and L_I)**.** *Suppose that an invariant I has formula size l. Let m be the number of instances of the operator $\Diamond$ in I. Then the formula size of the transition configuration T_I is at most $l + 3m$.*

If we regard the operator $\diamondsuit$ used by the conditional liveness formula L_I as primitive, then the formula size of L_I never exceeds that of I and only depends on the number of $\Diamond$ operators in I and the formula size of their operands. More precisely, L_I has the same formula size as the conjunction of all $\Diamond$-dependencies in I and therefore has formula size no larger than I's.

Proof. In the case of the transition formula T_I, each operand $\Diamond w$ in I is replaced in T_I by the formula $\Diamond(w \vee \bigcirc r)$, for some dependent variable r. This has the three extra symbols $\vee \bigcirc r$ since we ignore bracketing. Within the conditional liveness formula L_I, each $\Diamond$-dependency in I of the form $r \equiv \Diamond w$ is simply replaced by the implication $r \supset \diamondsuit w$ which has the same formula size. □

11.1 Reduction of basic invariant configurations

Starting with an ordered invariant I, let us now consider the relationship between its basic invariant configuration and the associated finite-time and infinite-time invariant configurations. This permits us to focus the remaining analysis on the two later kinds of invariant configurations.

Lemma 11.9. *A basic invariant configuration $\Box\, I \wedge w$ is satisfiable iff at least one of its associated finite-time and infinite-time invariant configurations is satisfiable.*

Proof. This follows from the validity of the formula $finite \vee inf$ and simple propositional reasoning. □

The finite-time and infinite-time invariant configurations for the ordered invariant I each have a corresponding semantically equivalent transition configuration of the same kind now described and shortly proved:

	Invariant configuration	Transition configuration	Where proved
Finite time	$\Box\, I \wedge w \wedge \mathit{finite}$	$\Box\, T_I \wedge w \wedge \mathit{finite}$	Theorem 11.11
Infinite time	$\Box\, I \wedge w \wedge \mathit{inf}$	$\Box\, T_I \wedge w \wedge \Box \Diamond^+ L_I$	Theorem 11.14

Observe that the reductions from the two types of the invariant configurations to the corresponding transition configurations do not introduce any extra variables.

In what follows we will often abstract the behaviour of a $\Diamond$-dependency by using two propositional variables p and q and representing the dependency as the PTL equivalence $p \equiv \Diamond\, q$. This technique is used to establish the next lemma:

Lemma 11.10. *The formulas $\Box\, I$ and $\Box\, T_I$ are semantically equivalent on finite intervals. In other words, the following implication is valid:*

$$\models \quad \mathit{finite} \quad \supset \quad \Box\, I \;\equiv\; \Box\, T_I.$$

Proof. We can represent $\Box\, I$ as the conjunction $\bigwedge_{k:1 \leq k \leq |I|} \Box\, I[k]$ and similarly represent $\Box\, T_I$ as the conjunction $\bigwedge_{k:1 \leq k \leq |I|} \Box\, T_I[k]$. For any $k : 1 \leq k \leq |I|$, if $I[k]$ is in NL^1 then $T_I[k]$ is identical to it and hence $\Box\, I[k]$ and $\Box\, T_I[k]$ are identical. Otherwise, $I[k]$ is a $\Diamond$-dependency. In such a case, the formula $\Box\, I[k]$ can be seen as a substitution instance of the PTL formula $\Box(p \equiv \Diamond\, q)$ containing the two propositional variables p and q. Now $\Box\, T_I[k]$ therefore corresponds to the formula $\Box(p \equiv (q \vee \bigcirc p))$. Simple temporal reasoning can then be used to show that each of these implies the other in any finite interval. □

Let us note that the validity for finite time of the relevant equivalence $\Box(p \equiv \Diamond\, q) \equiv \Box(p \equiv (q \vee \bigcirc p))$ can even be readily checked by a computer implementation of a decision procedure for PTL with finite time.

Theorem 11.11. *The finite-time invariant configuration for I is semantically equivalent to the associated finite-time transition configuration.*

Proof. This readily follows from Lemma 11.10 and propositional reasoning. □

Unfortunately, the equivalence $\Box\, I \equiv \Box\, T_I$ can fail to be valid for infinite time if I contains $\Diamond$-dependencies because T_I does not fully capture the liveness requirements of such dependencies. Lemma 11.13 later on corrects for this problem by showing that in infinite time the two formulas $\Box\, I$ and $\Box\, T_I \wedge \Box \Diamond^+ L_I$ are semantically equivalent. The reason that $\Box\, I \equiv \Box\, T_I$ is not necessarily valid is because when we consider an individual $\Diamond$-dependency, the formulas $\Box(p \equiv \Diamond\, q)$ and $\Box(p \equiv (q \vee \bigcirc p))$ are not semantically equivalent on infinite-time intervals since on such an interval, the first formula can be false and the second one true. An example of this occurs in any infinite interval where p is always true and q is always false. Therefore, if I contains $\Diamond$-dependencies, then $\Box\, I$ can be false on an infinite-time interval even though $\Box\, T_I$ is true on the interval. However, the next lemma holds even for infinite time:

Lemma 11.12. *The* PTL *implication $\Box\, I \;\supset\; \Box\, T_I$ is valid.*

Proof. The NL^1-dependencies in I and T_I are identical. Furthermore, for the $\Diamond$-dependencies we make use of the next valid PTL formula:

$$\models \quad \Box(p \equiv \Diamond\, q) \;\supset\; \Box(p \equiv (q \vee \bigcirc p)).$$ □

We see from Lemma 11.12 that the formula $\Box\, I \supset \Box\, T_I$ is valid for both finite and infinite time. However if I contains $\Diamond$-dependencies, then the converse implication $\Box\, T_I \supset \Box\, I$ is not necessarily valid for infinite time because the implication $\Box(p \equiv (q \vee \bigcirc p)) \supset \Box(p \equiv \Diamond\, q)$ fails to be valid. We now discuss the principles which successfully correct for this. First of all, the following weakened implication concerning an individual $\Diamond$-dependency is valid:

$$\models \quad \Box(p \equiv (q \vee \bigcirc p)) \quad \supset \quad \Box(\Diamond\, q \supset p).$$

Here we use the formula $\Diamond\, q \supset p$ instead of the stronger equivalence $p \equiv \Diamond\, q$. The following equivalence then strengthens the effect of $\Box(p \equiv (q \vee \bigcirc p))$ by adding the formula $\Box(p \supset \Diamond\, q)$:

$$\models \quad \Box(p \equiv \Diamond\, q) \quad \equiv \quad \Box(p \equiv (q \vee \bigcirc p)) \wedge \Box(p \supset \Diamond\, q).$$

In fact, we can even replace the conjunct $\Box(p \supset \Diamond\, q)$ by the weaker formula $\Box\, \Diamond(p \supset \Diamond\, q)$ which adds a $\Diamond$:

$$\models \quad \Box(p \equiv \Diamond\, q) \quad \equiv \quad \Box(p \equiv (q \vee \bigcirc p)) \wedge \Box\, \Diamond(p \supset \Diamond\, q).$$

All three valid formulas only contain the propositional variables p and q and can consequently be readily checked for infinite-time validity by any computer implementation of a decision procedure for PTL with infinite time.

Now suppose the ordered invariant I has m $\Diamond$-dependencies and hence $m = |L_I|$. If we have m pairs of propositional variables $p_1, q_1, \ldots, p_m, q_m$ (corresponding to I's $\Diamond$-dependencies) then the following generalization of the previous valid equivalence is itself valid:

$$\models \quad \Box \bigwedge_{1 \le k \le m} (p_k \equiv \Diamond\, q_k)$$
$$\equiv \quad \Box \bigwedge_{1 \le k \le m} (p_k \equiv (q_k \vee \bigcirc p_k)) \wedge \Box\, \Diamond \bigwedge_{1 \le k \le m} (p_k \supset \Diamond\, q_k).$$

The left side of the equivalence corresponds to the invariant $I[1 : m]$. Similarly, the first conjunct on the right side corresponds to $T_I[1 : m]$ and the second one to L_I, except for the use of $\Diamond$ instead of $\diamondsuit$.

Now within infinite time, $\Box\, \Diamond$ and $\Box\, \Diamond^+$ have the same behaviour and in addition $\Diamond$ and $\diamondsuit$ act identically. We use this to obtain the next lemma which expresses I in terms of T_I and L_I:

Lemma 11.13. *The formula* $inf \supset \big(\Box\, I \equiv (\Box\, T_I \wedge \Box\, \Diamond^+ L_I)\big)$ *is valid.*

Theorem 11.14. *An infinite-time invariant configuration* $\Box\, I \wedge w \wedge inf$ *for the ordered invariant* I *is semantically equivalent to the associated infinite-time transition configuration* $\Box\, T_I \wedge w \wedge \Box\, \Diamond^+ L_I$.

Proof. This readily follows from Lemma 11.13 and simple temporal reasoning. □

11.2 Bounded models for basic invariant configurations

The theorem given below gives the small model property for basic invariant configurations:

Type of invariant config.	Max. # of variables to represent a state
Finite-time	$\|V\|$
Infinite-time (see §8.1)	$2\|V\| + n$
Infinite-time (see §8.2)	$\|V\| + 2n$
Infinite-time (also in §8.2)	$\|V\| + n + \lceil \log_2(n+1) \rceil$,

where n is the number of $\Diamond$ operators in the invariant.

Table 10: Variables used by decision procedures for invariants

Theorem 11.15. *Suppose V is a finite set of variables and the variables in the ordered invariant I and the state formula w are all elements of V. Then the basic invariant configuration $\Box\, I \wedge w$ is satisfiable iff it is satisfied by some some finite interval with interval length less than $|Atoms_V|$ or by an infinite, ultimately periodic one consisting of an initial segment with interval length less than $|Atoms_V|$ fused with a remaining infinite periodic part with a period having interval length at most $(|L_I|+1)\,|Atoms_V|$.*

Proof. Suppose $\Box\, I \wedge w$ is satisfiable. We will consider the two cases of finite and infinite intervals separately:

Case for finite intervals: Theorem 11.11 ensures that the finite-time invariant configuration $\Box\, I \wedge w \wedge \mathit{finite}$ and its associated finite-time transition configuration $\Box\, T_I \wedge w \wedge \mathit{finite}$ are semantically equivalent. The construction of T_I ensures that any variable occurring in it is a member of the set V. Lemma 6.2 therefore establishes that if the conjunction $\Box\, T_I \wedge w \wedge \mathit{finite}$ is satisfiable, then a satisfying interval exists having less interval length than $|Atoms_V|$. This interval consequently also satisfies the basic invariant configuration $\Box\, I \wedge w$.

Case for infinite intervals: Theorem 11.14 ensures that the infinite-time invariant configuration $\Box\, I \wedge w \wedge \mathit{inf}$ and its associated infinite-time transition configuration $\Box\, T_I \wedge w \wedge \Box \Diamond^+ L_I$ are semantically equivalent. From Lemma 6.9 we have that this second formula is satisfied by an infinite interval consisting of an initial segment having interval length less than $|Atoms_V|$ fused with a periodic interval with period having interval length at most $(|L_I| + 1)\,|Atoms_V|$. The overall ultimately periodic interval therefore also satisfies the formula $\Box\, I \wedge w$. □

11.3 Decision procedures for invariant configurations

The decision procedures for transition configurations presented earlier in Section 8 can also be applied to invariant configurations by means of the previously described reductions from invariant configurations to transition configurations. The earlier Lemma 11.8 gives upper bounds on the formula size for an invariant I's associated transition formula T_I and conditional liveness formula L_I. Furthermore, the information in the previous Table 5 about the maximum number of variables required by the various BDD-based decision procedures to symbolically represent a single state for a transition configuration can be adapted to invariant configurations. Table 10 shows four cases. Recall that the decision procedures construct some BDDs which represent states and others which represent binary relations over pairs of states. Therefore, the number of variables required for the second kind of BDDs is double that shown in Table 10.

11.4 Axiomatic completeness for invariant configurations

Theorem 11.16. *Completeness holds for finite- and infinite-time invariant configurations.*

Proof. Suppose we have some invariant I. Assume without loss of generality that I is ordered since otherwise we can trivially rearrange its dependencies to obtain an ordered invariant which is both semantically and deducibly equivalent to I. Subsection 11.1 already described how to construct a semantically equivalent transition configuration from any finite-time or infinite-time invariant configuration associated with I. The various valid formulas mentioned there can be deduced as PTL theorems to establish that each such finite-time and infinite-time invariant configuration is also deducibly equivalent to the associated transition configuration. This and the previously shown axiomatic completeness for finite-time and infinite-time transition configurations respectively proved in Theorems 10.8 and 10.9 ensure that any consistent finite-time or infinite-time invariant configuration associated with I is satisfiable. Hence, we establish our immediate goal of completeness for finite- and infinite-time invariant configurations. □

Theorem 11.17. *Completeness holds for basic invariant configurations.*

Proof. Suppose we have some consistent basic invariant configuration $\Box\, I \wedge w$. Now the disjunction $\mathit{finite} \vee \mathit{inf}$ is easily deduced as a propositional tautology since inf is defined to be $\neg \mathit{finite}$ (see Table 1). It is then straightforward to show using purely propositional reasoning that $\Box\, I \wedge w$ is deducibly equivalent to the disjunction of its associated finite-time or infinite-time invariant configurations:

$$\vdash \quad \Box\, I \wedge w \quad \equiv \quad (\Box\, I \wedge w \wedge \mathit{finite}) \vee (\Box\, I \wedge w \wedge \mathit{inf}).$$

Hence at least one of the latter is also consistent. The previous Theorem 11.16 ensures that any such consistent finite- or infinite-time invariant configuration is satisfiable as well. An interval which satisfies it can also serve as a model for the basic invariant configuration. This demonstrates the desired axiomatic completeness for all basic invariant configurations. □

12 Dealing with arbitrary PTL formulas

So far we have only looked at bounded models and axiomatic completeness for certain kinds of PTL formulas. For an arbitrary PTL formula X, it is straightforward to construct an invariant I with formula size (recall Definition 11.7) which is linearly bounded by X's formula size. The invariant contains a finite number of dependent variables r_1, r_2, ..., $r_{|I|}$ not themselves occurring in X. We can then mimic the semantics of X since it is satisfiable iff the invariant configuration $\Box\, I \wedge r_1$ is satisfiable. In addition, the implication $\Box\, I \supset (r_1 \equiv X)$ is valid.

Assume that the only temporal operators in X are $\bigcirc$ and $\Diamond$ with others such as $\Box$ expressed using them (e.g., $\Box\, p$ becomes $\neg \Diamond \neg p$). The most straightforward way to construct the invariant for X is to first start with the equivalence $r_1 \equiv X$, where the variable r_1 does not occur in X. Now replace each a subformula having $\bigcirc$ or $\Diamond$ as its main operator in this righthand instance of X by a distinct dependent variable. The original equivalence now becomes $r_1 \equiv w$ for some state formula w. We then construct a conjunction of n additional dependencies (recall Definition 11.1), where n

is the number of X's temporal operators. For example, here is a suitable invariant for the PTL formula $(\bigcirc p) \vee \Diamond(\bigcirc q \vee \neg \bigcirc q')$:

$$r_1 \equiv (r_2 \vee r_3) \ \wedge \ r_2 \equiv \bigcirc p \ \wedge \ r_3 \equiv \Diamond(r_4 \vee \neg r_5) \ \wedge \ r_4 \equiv \bigcirc q \ \wedge \ r_5 \equiv \bigcirc q'. \quad (40)$$

It is easy to check by doing induction on X's syntactic structure that X is satisfiable iff the basic invariant configuration $\Box I \ \wedge \ r_1$ is satisfiable. Furthermore, the implication $\Box I \ \supset \ (r_1 \equiv X)$ can be shown to be valid. Consequently, $\Box I \ \wedge \ r_1$ can be used to represent X's behaviour (modulo the dependent variables which act as auxiliary ones). The bounded model for the invariant configuration (see Theorem 11.15) satisfies X as well.

The decision procedure described in Section 8 can be utilized to check the satisfiability of arbitrary PTL formulas by reducing them first to basic invariant configurations and then testing the associated finite-time and infinite-time transition configurations (see Subsection 11.1). As detailed there, we transform the invariant into a transition formula and conditional liveness formula using Definitions 11.5 and 11.3, respectively. No new dependent variables are needed. The resulting formulas have formula size which is linear in that of the invariant and hence of X itself.

Axiomatic completeness for X readily reduces to that for the invariant configuration $\Box I \ \wedge \ r_1$.

Let us now look in more detail at the formula size of an invariant generated for some arbitrary PTL formula X. The invariant contains one new dependent variable for the overall formula X and at most one new dependent variable for each temporal operator in X. Hence, the total number of dependencies and dependent variables in the invariant (including the original one for X itself) is bounded by the formula size of X. In fact we have the following lemma concerning the formula size of invariants generated from PTL formulas:

Lemma 12.1 (Formula size of generated invariants)**.** *Let X be a* PTL *formula, let l be its formula size and let m be the number of temporal operators in X. The formula size of the invariant generated from X is linearly bounded by X's formula size and is in fact less than $l + 4(m + 1)$.*

Proof. The introduction of a dependency for each temporal operator requires at most four additional symbols. The first dependency for the overall formula X requires just one instance of the dependent variable r_1 and a single logical equivalence operator. Any further new dependencies are introduced because of the presence of temporal operators in X. Each such dependency requires within the final invariant two instances of the associated dependent variable, one instance of the equivalence operator $\equiv$ and one logical-and operator $\wedge$ to include the dependency in the final invariant. □

For example, let $\Diamond^n p$ denote n instances of $\Diamond$ followed by the variable p. Here is the form of an invariant for this for some $n \geq 1$:

$$r_1 \equiv \Diamond r_2 \quad \wedge \quad r_2 \equiv \Diamond r_3 \quad \wedge \quad \cdots \quad \wedge \quad r_n \equiv \Diamond p.$$

The original formula $\Diamond^n p$ has formula size $n + 1$ and the invariant has formula size $5n - 1$. This example is a kind of worst case since many PTL formulas do not contain such a high density of temporal operators.

12.1 Two simple improvements

Let us now consider two simple improvements to the transformation of arbitrary PTL formulas into invariants which can yield shorter invariants containing fewer dependent variables. For the first improvement, note that just prior to introducing a new dependent variable for a subformula, we can check whether the same subformula has already been encountered elsewhere in X and previously assigned a dependent variable. If so, this dependent variable can be used and further (redundant) processing of the subformula can be skipped. This technique both reduces the number of dependent variables and the formula size of the final invariant. For example, if X is $(\Diamond p) \wedge \Diamond(q \vee \Diamond p)$, we can obtain the following invariant:

$$r_1 \equiv (r_2 \wedge r_3) \quad \wedge \quad r_2 \equiv \Diamond p \quad \wedge \quad r_3 \equiv \Diamond(q \vee r_2). \tag{41}$$

Here the dependent variable r_2 corresponding to X's subformula $\Diamond p$ occurs twice in dependent formulas on the righthand sides of equivalences.

The second improvement concerns the state formula in an invariant configuration. If the dependent formula associated with the first dependent variable r_1 is a state formula, then we can eliminate r_1 from the invariant and use the state formula in its place as the initial state formula in any associated invariant configuration. For example, the invariant (41) can be shortened to the following one:

$$r_2 \equiv \Diamond p \quad \wedge \quad r_3 \equiv \Diamond(q \vee r_1).$$

The original formula $(\Diamond p) \wedge \Diamond(q \vee \Diamond p)$ can then be represented by the basic invariant configuration now given:

$$\Box\big(r_2 \equiv \Diamond p \ \wedge \ r_3 \equiv \Diamond(q \vee r_1)\big) \quad \wedge \quad (r_2 \wedge r_3).$$

Therefore if the original formula contains m temporal operators, at most m new dependencies and dependent variables are required in the associated invariant rather than $m + 1$.

Table 11 shows the number of variables required to represent a state in the various BDD-based decision procedures. The bounds can be obtained by observing that the reduction to an invariant requires at most m new variables, where m is the number of temporal operators in the original PTL formula. The earlier Table 10 can then be used to calculate the values by replacing each instance of $|V|$ by $|V| + m$. Recall that the decision procedures construct some BDDs which represent states and others which represent binary relations over pairs of states. Therefore, the number of variables required for the second kind of BDDs is double that shown in Table 11.

12.2 A way to obtain even smaller invariants

We now describe another way of construct invariants with fewer dependencies and dependent variables. It makes use of dependencies containing arbitrarily complex NL^1 formulas. For example, if X is the earlier formula $(\bigcirc p) \vee \Diamond(\bigcirc q \vee \neg \bigcirc q')$, we can obtain the following invariant containing only 3 dependencies and with formula size 20:

$$r_1 \equiv (\bigcirc p \ \vee \ r_3) \quad \wedge \quad r_2 \equiv (\bigcirc q \vee \neg \bigcirc q') \quad \wedge \quad r_3 \equiv \Diamond r_2. \tag{42}$$

This is smaller than the earlier invariant (40) which has 5 dependencies and formula size 28.

Type of interval	Max. # of variables to represent a state
Finite-time	$\|V\| + m$
Infinite-time (using §8.1)	$2(\|V\| + m) + n$
Infinite-time (using §8.2)	$\|V\| + m + 2n$
Infinite-time (also in §8.2)	$\|V\| + m + n + \lceil \log_2(n+1) \rceil$,

where m is the number of temporal operators
and n is the number of $\Diamond$ operators in the formula.

Table 11: Variables used by decision procedures for arbitrary formulas

We start with the equivalence $r_1 \equiv X$, where r_1 is a new dependent variable. Now check whether the equivalence $r_1 \equiv X$ is already itself a well-formed dependency. Recall Definition 11.1, concerning invariants and dependencies, which states that for any propositional variable r, the equivalence $r \equiv X$ is a dependency iff X is either of the form $\Diamond w$ for some state formula w or X is in NL^1. We will now give a lemma which states an alternative characterization of an equivalence such as $r_1 \equiv X$ being a dependency. This is done using two conditions. By expressing the requirements for X in terms of these conditions, we can more clearly see how to obtain a invariant from the initial equivalence $r_1 \equiv X$.

Within this approach, let the term *strict* NL^1 *formula* denote an NL^1 formula containing at least one instance of the operator $\bigcirc$. Observe that an NL^1 formula cannot itself contain a strict NL^1 formula within the scope of a $\bigcirc$ operator since this would nest one $\bigcirc$ in another.

Lemma 12.2 (Alternative characterization of dependency)**.** *Let r be some propositional variable and X be a* PTL *formula. The following are equivalent:*

(a) The equivalence $r \equiv X$ is a dependency.

(b) The following two conditions hold for X:

1. *X does not contain a subformula $\Diamond w$, for some state formula w, nested within any operator.*
2. *X does not contain a strict* NL^1 *subformula nested within some temporal operator.*

Proof. $(a) \Rightarrow (b)$*:* From the definition of invariants and dependencies in Definition 11.1 it is not hard to see that if $r \equiv X$ is a dependency, then both conditions are observed. We show this by considering the two possible forms X can have:

- X has the form $\Diamond w$: Clearly Condition 1 holds since X itself is not nested in any temporal operators. Also Condition 2 holds since X contains no $\bigcirc$ at all.
- X is a formula in NL^1: Such an X observes Condition 1 since X does not contains any $\Diamond$ operator. Also, as already noted, an NL^1 formula cannot contain a strict NL^1 subformula within the scope of any $\bigcirc$ operator. Therefore Condition 2 holds.

$(b) \Rightarrow (a)$*:* We first consider the case where X contains a $\Diamond$ and then the case where it does not contain one:

- X contains a $\Diamond$: Condition 1 ensures that this $\Diamond$ must be exactly the outermost operator. Also, by Condition 2, no $\bigcirc$ can occur within the scope of the $\Diamond$ so X has the form $\Diamond\, w$ for some state formula w.

- X does not contain a $\Diamond$: Condition 2 guarantees that any $\bigcirc$ in X does not occur in another $\bigcirc$ so since X has no $\Diamond$ in it then X itself is in NL^1. □

Now let us obtain an invariant from the equivalence $r_1 \equiv X$, where r_1 is a propositional variable not occurring in X. By Lemma 12.2, this equivalence is a dependency iff both alternative Conditions 1 and 2 in (b) hold for X. If either condition is violated by X and hence $r_1 \equiv X$ is not a dependency, we replace some offending temporal subformula, say Y, in X by a new distinct dependent variable, say r_2, to obtain from X a smaller formula X'. Indeed, X can be viewed as a substitution instance of X', that is, $X'^{Y}_{r_2}$. The equivalence associated with X now becomes $r_1 \equiv X'$. We also associate r_2 with the additional equivalence $r_2 \equiv Y$, which is a well-formed dependency since by the two alternative Conditions 1 and 2 in Lemma 12.2 the offending subformula is either of the form $\Diamond\, w$, for some state formula w, or is in NL^1. The process is repeated on X' to check whether it fulfils the alternative Conditions 1 and 2 in Lemma 12.2. If not, this results in more new dependent variables and equivalences, as well as a new formula X'' which used in the next iteration instead of X'. Eventually, we terminate with a finite nonempty set of equivalences which are well-formed dependencies and can be conjoined together to obtain an invariant I with the valid implication $\Box\, I \supset (r_1 \equiv X)$.

As we previously noted, the invariant (42) can be constructed in this way from the formula $(\bigcirc p) \vee \Diamond(\bigcirc q \vee \neg \bigcirc q')$. First observe that the strict NL^1 subformula $\bigcirc q \vee \neg \bigcirc q'$ violates Lemma 12.2's Condition 2 since it occurs within a $\Diamond$ construct. Therefore, we replace it in X by the new dependent variable r_2 to obtain a new overall formula $\bigcirc p \vee \Diamond r_2$. Then observe that the subformula $\Diamond r_2$ violates Lemma 12.2's Condition 1 so replace it by the new dependent variable r_3. The resulting overall formula $\bigcirc p \vee r_3$ is in NL^1 and is therefore suitable for used in the first dependency $r_1 \equiv (\bigcirc p \vee r_3)$. A variation of the technique based on recursive descent is also possible but we omit the details here.

A generalized kind of invariant later described in Subsection 13.3 can further reduce the need for dependent variables by permitting the operand of a $\Diamond$-formula in a $\Diamond$-dependency to be an arbitrary NL^1 formula, rather than just a state formula. More compact list-based representations of invariants are also possible.

13 Some additional features

This section describes a number of extensions to our approach. They include the temporal operator *until* and past-time constructs (both extensively discussed in the literature which is surveyed by Emerson [25], Lichtenstein and Pnueli [57] and other researchers cited elsewhere in our presentation). In addition, the liveness tests found in conditional liveness formulas and invariants can be generalized to be NL^1 formulas, rather than just state formulas. Another feature considered here concerns a subset of PITL called *Fusion Logic* (FL) which includes constructs of the sort found in Propositional Dynamic Logic (PDL) [27, 28, 39–41, 52]. We will look at each of these issues in turn. For the sake of brevity, the presentation is briefer and less formal than in the previous sections.

13.1 The operator *until*

The operator *until* is a binary operator with the syntax $X \,\mathcal{U}\, Y$, where X and Y are PTL formulas. Recall from Section 4 that for any interval σ and natural number k which does not exceed σ's interval length, $\sigma^{k:|\sigma|}$ denotes the suffix subinterval obtained by deleting the first k states from σ. Here is the semantics of *until*:

$\sigma \models X \,\mathcal{U}\, Y$ iff

for some $k \leq |\sigma|$, $\sigma^{k:|\sigma|} \models Y$ and for all $j : 0 \leq j < k, \sigma^{j:|\sigma|} \models X$.

Observe that the operator $\Diamond$ can be expressed in terms of *until* since $\Diamond X$ is semantically equivalent to the formula $true\ until\ X$. Kamp [50] first proposed and extensively studied a version of *until* which ignores the present state (together with a past-time analogue called *since*). He looked at the semantic expressiveness of *until* within different models of time (see also Prior [80], Rescher and Urquhart [83], Emerson [25], Gabbay, Hodkinson and Reynolds [35] and Blackburn, de Rijke and Venema [7]). The influential analyses by Gabbay et al. [33] and Lichtenstein, Pnueli and Zuck [58] of PTL with *until* and a discrete linear model of time are also of particular note. Burgess [15] shows the completeness of an axiom system for PTL with *until* and *since*. However, time is modelled as being linear but not necessarily discrete. Marx, Mikulas and Reynolds [61] consider the complexity of decidability and axiomatic completeness of PTL with various versions of linear time and include an analysis with *until* and *since*. Like Burgess, they do not restrict time to being discrete.

We can alter the definition of invariants by replacing $\Diamond$-dependencies with dependencies of the form $r \equiv (w \,\mathcal{U}\, w')$, where w and w' are state formulas. If the j-th dependency $I[j]$ of an invariant I is such a dependency (called an *until-dependency*), then the corresponding conjunction $T_I[j]$ in I's transition formula T_I has the form $r \equiv (w' \vee (w \wedge \bigcirc r))$. The associated conjunction $L_I[j]$ in L_I is $r \supset \diamondsuit\, w'$. It is not hard to modify the material in Section 11 to ensure that finite-time and infinite-time invariant configurations remain semantically equivalent to the associated transition configurations.

Alternatively, we can transform an invariant with *until* in it to one without it. Each dependency in I of the form $u_k \equiv (w\ until\ w')$ is replaced by the dependency $u_k \equiv \big(u'_k \wedge (w' \vee (w \wedge \bigcirc u_k))\big)$, where u'_k is a new dependent variable with the associated dependency $u'_k \equiv \Diamond w'$. This approach is more hierarchical than the first one but increases the number of dependencies used.

13.2 Past time

Prior [80], Rescher and Urquhart [83] and others originally studied temporal logics having models of time with both a future and a past and including past-time analogues of $\Diamond$ and $\Box$. Time in such frameworks is not necessarily discrete or even linear. Prior [80] credits Scott with the first versions of $\bigcirc$ and a past-time analogue (referred to by Prior as *tomorrow* and *yesterday*, respectively) for use with discrete models of time. Gabbay et al. [33] strongly argue that PTL without past time is sufficiently expressive for many purposes within the context of computer science. Somewhat later, Lichtenstein, Pnueli and Zuck [58] argue the case for past time even in computer science applications of temporal logic. Lichtenstein and Pnueli [57] subsequently describe this reevaluation as "A major reversal of our view about the essentiality of the past operators…". More recently, Laroussinie, Markey and Schnoebelen [56] have formally shown that PTL with past time can be exponentially more succinct that PTL without it.

Let us now consider PTL with a linear, discrete model of time having a bounded past. The syntax is modified to include the two additional primitive operators $\ominus X$ (read *previous* X) and $\diamondminus X$ (read *once* X). The set of PTL formulas including past-time constructs is denoted as PTL$^-$. The semantics of a PTL formula X is now expressed as $(\sigma, k) \models X$ where k is any natural number not exceeding $|\sigma|$. For example, the semantics of $\ominus$ and $\diamondminus$ are as follows:

$$(\sigma, k) \models \ominus X \quad \text{iff} \quad k > 0 \text{ and } (\sigma, k-1) \models X$$
$$(\sigma, k) \models \diamondminus X \quad \text{iff} \quad \text{for some } j : 0 \leq j \leq k, \ (\sigma, j) \models X.$$

We define the operator $\boxminus X$ (read *so-far* X) as $\neg \diamondminus \neg X$ and the operator $\circledcirc X$ (read *weak previous* X) as $\neg \ominus \neg X$. The operator $first$ is defined to be $\neg \ominus true$ and tests for the first state of an interval. The past-time version of $until$ called $since$ can also be included but we omit the details.

A PTL$^-$ formula X is defined to satisfiable iff $(\sigma, k) \models X$ holds for some pair (σ, k) with $k \leq |\sigma|$. The formula X is valid iff $(\sigma, k) \models X$ holds for every pair (σ, k) with $k \leq |\sigma|$. Note that these straightforward definitions of satisfiability and validity correspond to the so-called *floating framework* of PTL with past time. However, Manna and Pnueli propose another interesting approach called the *anchored framework* [60] (also discussed by Lichtenstein and Pnueli in [57]) which they argue is superior. In this framework, satisfiability and validity only examine pairs of the form $(\sigma, 0)$. There exist ways to go between the two conventions but we will not delve into this here and instead simply assume the more traditional floating interpretation.

We now define an analogue of the set of formulas NL:

Definition 13.1 (Previous Logic)**.** *The set of* PTL *formulas in which the only primitive temporal operator is* $\ominus$ *is called here* Previous Logic (PrevL)*. The subset of* PrevL *with no* $\ominus$ *nested in another* $\ominus$ *is denoted as* PrevL^1.

We let the variables Z and Z' denote formulas in PrevL^1. Also, PrevL^1_V denotes the set of all formulas in PrevL^1 only having variables in V.

The following definitions extend the notation of transition configurations to deal with past time:

Definition 13.2 (Past-time transition configurations)**.** *A* past-time transition configuration *is any formula of the form* $\boxminus\Box(T \wedge Z) \wedge X$*, where* T *is in* NL^1_V*,* Z *is in* PrevL^1_V*, and the formula* X *is in* PTL$_V$ *and is in one of the two categories shown below:*

Type of configuration	*Syntax of* X
Finite-time	$w \wedge$ *finite*
Infinite-time	$w \wedge \Box \diamondsuit^+ L$

Here w *is a state formula in* PROP$_V$ *and* L *is a conditional liveness formula in* PTL$_V$.

The formula $\boxminus\Box(T \wedge Z)$ contains both $\boxminus$ and $\Box$ to ensure that both T and Z are true everywhere in the interval.

The analysis of a finite-time or infinite-time past-time transition configurations can be easily reduced to reasoning in PTL without past time. Let us demonstrate this by first examining how to test the satisfiability of a finite-time past-time transition configuration $\boxminus\Box(T \wedge Z) \wedge w \wedge$ *finite*. This involves finding an interval σ and natural number $k \leq |\sigma|$, such that $(\sigma, k) \models \boxminus\Box(T \wedge Z) \wedge w \wedge$ *finite* holds. Note that this

past-time transition configuration is satisfiable iff the following formula, which shifts reasoning back to an interval's starting state, is satisfiable:

$$\diamondminus\big(\Box(T \wedge Z) \wedge \mathit{first} \wedge \Diamond w \wedge \mathit{finite}\big). \tag{43}$$

Here we can dispense with the operator $\boxminus$ since $\boxminus\Box$ and $\Box$ have the same semantics at the starting state.

Now for any PTL$^-$ formula X, the formula $\diamondminus X$ is satisfiable iff X is satisfiable. Hence, the formula (43) is satisfiable iff its subformula $\Box(T \wedge Z) \wedge \mathit{first} \wedge \Diamond w \wedge \mathit{finite}$ is satisfiable. Let us now define the NL^1_V formula T' by replacing each $\ominus$ construct in Z by its operand and by taking each state formula in Z which does not occur in $\ominus$ and enclosing it in $\bigcirc$. For example, if Z is the formula $p \vee \ominus(q \wedge r)$, then T' is $(\bigcirc p) \vee (q \wedge r)$. Furthermore, let w' be the state formula in PROP$_V$ obtained from Z by replacing each $\ominus$ construct by false. In our example, w' is $p \vee \mathit{false}$. It can be readily checked that the following formula relating Z and T' is true at any interval's first state: $\Box Z \equiv \boxdot T' \wedge w'$. In other words, the next implication is valid:

$$\models \quad \mathit{first} \quad \supset \quad (\Box Z \equiv \boxdot T' \wedge w').$$

Therefore, the original finite-time past-time transition configuration is satisfiable iff the following formula in PTL without past time is satisfiable:

$$\Box(T \wedge (\mathit{more} \supset T')) \wedge w' \wedge \Diamond w \wedge \mathit{finite}. \tag{44}$$

This is still not a well-formed finite-time transition configuration due to the presence of the formula $\Diamond w$. However, $\Diamond w$ can be reduced by introducing a new propositional variable r as shown in the next formula:

$$\Box\big(T \wedge (\mathit{more} \supset T') \wedge (r \equiv (w \vee \bigcirc r))\big) \wedge w' \wedge r \wedge \mathit{finite}. \tag{45}$$

The reduction of the original past-time transition configuration $\boxminus\Box(T \wedge Z) \wedge w \wedge \mathit{finite}$ to the finite-time transition configuration (45) systematically relates all aspects of the analysis of the past-time transition configuration to the purely future-only reasoning presented earlier. This includes bounded models, decision procedures and axiomatic completeness.

An alternative way to reduce the PTL formula (44) involves interval-based reasoning. We first re-express the formula in PTL as the next semantically equivalent conjunction:

$$\boxdot(T \wedge (\mathit{more} \supset T')) \wedge w' \wedge \Diamond w \wedge \mathit{sfin}\ T. \tag{46}$$

This makes use of the valid PTL equivalence $(\Box X \wedge \mathit{finite}) \equiv (\boxdot X \wedge \mathit{sfin}\ X)$, for any PTL formula X. However, in our case we can omit the subformula $\mathit{more} \supset T'$ in the sfin construct since the operator more ensures that the implication is trivially true in the associated empty interval. Let T'' denote the subformula $T \wedge (\mathit{more} \supset T')$. Theorem 5.4 ensures the semantic equivalence of $\boxdot T''$ and $(\$\, T'')^*$. Now the formula (46) can in turn be itself re-expressed as the following chop-formula:

$$((T'')^* \wedge w' \wedge \mathit{finite});\ ((T'')^* \wedge w \wedge \mathit{sfin}\ T). \tag{47}$$

Let w'' denote a state formula obtained by replacing every $\bigcirc$ construct in T by false. Consequently, w'' is true exactly in states for which $T \wedge \mathit{empty}$ is true. It follows that

we can test for satisfiability of formula (47) by adapting the symbolic methods mentioned in Section 8 to solve for V-atoms α, β and γ for which the following formulas are satisfiable:

$$\alpha \wedge w \quad (\$\, T'')^* \wedge \alpha \wedge \mathit{sfin}\ \beta \quad \beta \wedge w' \quad (\$\, T'')^* \wedge \beta \wedge \mathit{sfin}\ \gamma \quad \gamma \wedge w''.$$

Further details are omitted here.

The treatment for a infinite-time past-time transition configuration is nearly identical to that for a finite-time one since the assumption of a bounded past still applies and avoids the need for a past-time conditional liveness formula. First of all, we replace the subformula finite by $\Box \Diamond^+ L$.

$$\Box(T \wedge T') \wedge w' \wedge \Diamond w \wedge \Box \Diamond^+ L.$$

The use of infinite time ensures we can omit the instance of more found in the finite-time formulas (44) and (45) since T and $\mathit{more} \supset T$ are semantically equivalent on an infinite interval. The formula $\Diamond w$ is itself reduced by introducing a new propositional variable r and conjoining a new implication to L to obtain the well-formed infinite-time transition configuration below:

$$\Box(T \wedge T') \wedge w' \wedge r \wedge \Box \Diamond^+ (L \wedge (r \supset \Diamond\!\!\!\!-\, w)).$$

So far we have only considered finite- and infinite-time transition configurations. Invariants (and hence also invariant configurations) can be extended to support past-time reasoning by adding two new kinds of dependencies. The first has the form $u \equiv Z$, where Z is a formula in PrevL[1], and the second has the form $u \equiv \Diamond\!\!\!\!-\, w$. The use of $\Diamond\!\!\!\!-$ does not involve I's conditional liveness formula L_I due to the assumption of a bounded past. The definitions of invariant configurations remain the same and the reduction of them to past-time transition configurations is straightforward since no dependency contains both future- and past-time temporal constructs. Furthermore, dependencies containing the temporal operator since (a conventional past-time analogue of the operator until) are not much harder to handle than $\Diamond\!\!\!\!-$-dependencies. The reduction of an arbitrary PTL$^-$ formula to an invariant with past time is also straightforward.

13.3 Generalized conditional liveness formulas and invariants

Conditional liveness formulas and invariants require that their liveness tests, which respectively occur as operands of $\Diamond\!\!\!\!-$ and $\Diamond$, must be state formulas (recall Definitions 5.1 and 11.3). We can slightly relax this requirement and permit arbitrary formulas in NL[1]. This makes invariants more succinct since a formula such as $\mathit{sfin}\ w$ can now be expressed using only one dependency such as $u_k \equiv \Diamond(\mathit{empty} \wedge w)$ instead of requiring two. The formula $\Box \Diamond^+ w$ can be expressed with the invariant $u_k \equiv \Diamond(w \wedge \bigcirc u_k)$. The overall analysis of such invariants only differs slightly from that for the basic version of invariants. Invariants with until-dependencies (see Subsection 13.1) can be analogously generalized to permit until-dependencies of the form $u_k \equiv (T\, \mathcal{U}\, T')$, where both T and T' are in NL[1].

Transition configurations containing generalized liveness formulas might be of use as a notation for representing deterministic and nondeterministic ω-automata in temporal logic. However, we need to employ Quantified PTL (QPTL) to existentially quantify over the variables which collectively encode such an automaton's internal state. Further details of this are omitted here.

13.4 Fusion Logic

Regular expressions are a standard notation for representing regular languages. However, within PITL, it is more appropriate to use languages based on fusion (recall Definition 5.20) rather than conventional concatenation. This involves a variation of regular expressions called here *fusion expressions*. We now define a PITL-based representation of them which is in fact a special subset of PITL formulas. This subset will then provide the basis for a generalization of PTL called *Fusion Logic* (FL) which is also itself a subset of PITL. We originally used Fusion Logic in [73] as a kind of intermediate logic when we reduced the problem of showing axiomatic completeness of PITL with finite time to showing axiomatic completeness for PTL. Fusion Logic is closely related to Propositional Dynamic Logic (PDL) [27, 28, 39–41, 52]. A major reason for discussing Fusion Logic here is because it is not hard to extend our decision procedure for PTL with finite time to also handle more expressive interval-oriented FL formulas by simply reducing FL formulas to lower level PTL formulas of the kinds already discussed. This demonstrates another link between PTL and intervals and has practical applications.

Definition 13.3 (Fusion-expression formulas)**.** *The set of* fusion-expression formulas*, denoted* FE*, consists of* PITL *formulas with the syntax given below, where w is a state formula, T is in* NL[1] *and E and F themselves denote* FE *formulas:*

$$w? \qquad E \vee F \qquad \$\, T \qquad E; F \qquad E^*.$$

The syntax of FE *formulas is like that of programs in Propositional Dynamic Logic without rich tests. However* FE *has a semantics based on sequences of states rather than binary relations.*

For any set of variables V, let FE_V *denote the set of* FE *formulas containing only variables in V.*

Unlike letters in conventional regular expressions, any nonmodal formula can be used in $w?$. For example, $false?$ is permitted even though it is unsatisfiable. Consider the following FE formula:

$$\big((\$ \bigcirc p); (q?)\big) \vee (\$ \neg q)^*.$$

This is true on an interval if either the interval has exactly two states and p and q are both true in the second state or it has some arbitrary number of states, say k, with q false in each of the first $k - 1$ states.

Remark 13.4 (Expressing concatenation)**.** *It is important to note that the conventional concatenation of two* FE *formulas E and F can be achieved through the use of the* FE *formula $E; (\$\, true); F$. Here $\$\, true$ is itself an* FE *formula which is an alternative way to express the* PTL *operator $skip$. This temporal operation on E and F is sometimes called "*chomp*", since it is a slight variation of chop. Hence, in the context of temporal logic,* FE *formulas can largely subsume regular expressions although there are slightly different conventions for such things as empty words. We omit the details.*

We now present the sublogic of PITL called here Fusion Logic. In essence, Fusion Logic augments conventional PTL with the fusion-expression formulas already introduced.

Definition 13.5 (Fusion Logic)**.** *Here is the syntax of* FL *where p is any propositional variable, E is any* FE *formula and X and Y are themselves formulas in* FL*:*

$$p \qquad \neg X \qquad X \vee Y \qquad \bigcirc X \qquad \Diamond X \qquad \langle E \rangle X.$$

*We define the new construct $\langle E \rangle X$ (called "*FL-chop*") and its dual $[E]X$ (called "*FL-yields*") using the primitive* PITL *constructs chop and $\neg$:*

$$\langle E \rangle X \stackrel{\text{def}}{\equiv} E;X \qquad [E]X \stackrel{\text{def}}{\equiv} \neg\langle E \rangle \neg X.$$

Within an FL formula, $\bigcirc$, $\Diamond$ and FL-chop are treated as primitive constructs. Unlike PITL, FL limits the left sides of chop to being FE formulas.

In [73], we described an earlier version of FL having $skip$ as a primitive FE formula instead of $\$\,T$. As we noted previously in Remark 13.4, the PTL formula $skip$ can be expressed in FE as $\$\,true$. The two versions of FL can readily be shown to be equally expressive since $\$\,T$ can be replaced with a semantically equivalent disjunction of formulas by using of $?$, $skip$ and chop. For example, the FE formula $\$(p \supset \bigcirc q)$ is semantically equivalent to the FE formula $((\neg p)?; skip) \vee (skip; q?)$. In practice, the version described here is much more natural and succinct.

Henriksen and Thiagarajan [43,44] investigate a formalism related to Wolper's Extended Temporal Logic (ETL) [92,94] and called *Dynamic Linear Time Temporal Logic* which combines PTL and PDL in a linear-time framework with infinite time. It is similar to our Fusion Logic and uses multiple atomic programs instead of the FE operators $?$ and $\$$.

Remark 13.6. *The temporal operators $\bigcirc$ and $\Diamond$ which are primitives in* FL *can actually be expressed as instances of* FL*-chop if finite time is assumed:*

$$\models \bigcirc X \equiv \langle \$\, true \rangle X \qquad \models \Diamond X \equiv \langle (\$\, true)^* \rangle X.$$

In spite of FL being a proper subset of PITL, they have the same expressiveness. We discussed this in [73], where a hierarchical reduction of FL formulas to PTL formulas is also given but is limited to dealing with finite-time intervals. This reduction provides the basis of a decision procedure for FL with finite-time. An alternative hierarchical reduction to transition configurations in PTL is also possible and indeed we have implemented a version of it. Such transition configurations can be tested with the decision procedure for finite time described in Section 8. Like the approach in [73], this reduction could be used for proving the completeness of an axiom system for FL with finite time.

14 Discussion

We conclude with a look at some issues connected with PTL and FL.

As noted earlier, a number of PTL decision procedures are tableau-based algorithms. These include ones described by Wolper [95], Emerson [25] and Lichtenstein and Pnueli [57]. It appears that with some care a tableau-based approach can be hierarchically reduced to our framework. It might therefore be worthwhile to investigate the relationship between the two approaches in more detail.

The BDD-based techniques described in Section 8 can be adapted to check in real time that an executing system is not violating assertions expressed in PTL or FL as

it runs. Whether FL in particular is useful for this in practice is unclear. In addition, it would appear that the reachability analysis employed by us could, in the manner of Bounded Model Checking (BMC) [18], utilize SAT-based techniques for PTL and FL instead of BDDs. However, such a SAT-based approach, unlike the BDD-based one, normally cannot exhaustively test for unsatisfiability because of the lack of a notion corresponding to the convergence of BDDs to the set of all atoms reachable from some starting one. Rather BMC typically works by employing SAT to find at most a single solution not exceeding some predetermined maximum bounded number of states which for practical reasons is generally much less than the worst-case bounds derived from formula syntax. If a solution is not found, this is typically not by itself sufficient to exclude the existence of larger satisfying intervals. Nevertheless, Heljanko, Junttila and Latvala in a recent paper [42] describe a BMC-based complete decision procedure for PTL and its implementation and also mention other related work on this promising topic.

Although our primary application of invariants, transition formulas and conditional liveness formulas has been to temporal logics, we have also used versions of them to analyse Propositional Dynamic Logic (PDL) without the need for Fischer-Ladner closures. Indeed, this was the original motivation for conditional liveness formulas. However, at present the benefits and novelty of utilizing our approach for PDL are less compelling than for PTL.

Acknowledgements

We thank Antonio Cau, Jordan Dimitrov, Rodolfo Gómez, Helge Janicke and an anonymous referee for comments on this work. In the course of discussions, Howard Bowman, Shmuel Katz, Maciej Koutny and Simon Thompson also made helpful suggestions leading to improvements in the presentation of the material. We are especially grateful to Hussein Zedan for his patience and encouragement during the time this research was undertaken.

References

[1] ANSI. Common Lisp: Standard ANSI INCITS 226-1994 (R1999) (formerly ANSI X3.226-1994 (R1999)). `http://www.ansi.org`, 1999.

[2] B. Banieqbal and H. 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, Oct. 1986. revised June 1987.

[3] I. Beer, S. Ben-David, et al. The temporal logic Sugar. In G. Berry, H. Comon, and A. Finkel, editors, *13th Conference on Computer-Aided Verification (CAV01), Paris, France, 18–22 July 2001*, volume 2102 of *LNCS*, pages 363–367, Berlin, 2001. Springer-Verlag.

[4] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. In *Proc. 8th ACM Symp. on Principles of Programming Languages (POPL '81)*, pages 164–176. ACM, Jan. 1981.

[5] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. *Acta Informatica*, 20(3):207–226, 1983.

[6] O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Computer Aided Verification, Proc. 6th Int'l. Workshop*, volume 818 of *LNCS*, pages 142–155, Stanford, California, June 1994. Springer-Verlag.

[7] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Number 53 in Theoretical Tracts in Computer Science. Cambridge University Press, 2001.

[8] R. Bloem, H. N. Gabow, and F. Somenzi. An algorithm for strongly connected component analysis in $n \log n$ symbolic steps. *Formal Methods in System Design*, 28:37–56, 2006.

[9] A. Bolotov, M. Fisher, and C. Dixon. On the relationship between ω-automata and temporal logic normal forms. *Journal of Logic and Computation*, 12(4):561–581, Aug. 2002. Available as `http://www3.oup.co.uk/logcom/hdb/Volume_12/Issue_04/pdf/120561.pdf`.

[10] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8), 1986.

[11] R. E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, Sept. 1992.

[12] 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, California, 1962. Stanford University Press. Reprinted in [13, pp. 425–435].

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

[14] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Inf. and Comp.*, 98(2):142–170, June 1992.

[15] J. P. Burgess. Axioms for tense logic I: "Since" and "until". *Notre Dame Journal of Formal Logic*, 1982.

[16] Cadence Design Systems. `http://www.cadence.com/`.

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

[18] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1), July 2001.

[19] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, Massachusetts, 1999.

[20] CLISP: An ANSI Common Lisp implementation. `http://clisp.cons.org`.

[21] O. Coudert, C. Berthet, and J. C. Madre. Verification of sequential machines using boolean functional vectors. In L. Claesen, editor, *Proc. IFIP International Workshop on Applied Formal Methods for Correct VLSI Design*, pages 111–128, Leuven, Belgium, Nov. 1989.

[22] O. Coudert, C. Berthet, and J. C. Madre. Verification of synchronous sequential machines based on symbolic execution. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems, International Workshop, Grenoble, France, June 12-14, 1989, Proceedings*, volume 407 of *LNCS*, pages 365–373. Springer-Verlag, 1989.

[23] O. Coudert, C. Berthet, and J. C. Madre. A unified framework for the formal verification of sequential circuits. In *Proc. IEEE International Conf. on Computer Aided Design*, pages 126–129, Nov. 1990.

[24] Colorado University Decision Diagram Package (CUDD). Available at `http://vlsi.colorado.edu/~fabio`.

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

[26] E. A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In A. Meyer, editor, *Proc. 1st Ann. IEEE Symp. on Logic in Computer Science (LICS '86)*, pages 267–278. IEEE Computer Society Press, June 1986.

[27] M. J. Fischer and R. E. Ladner. Propositional modal logic of programs (extended abstract). In *Conference Record of the 9th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 286–294, Boulder, Colorado, 2–4 May 1977. ACM.

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

[29] M. Fisher. A normal form for first-order temporal formulae. In D. Kapur, editor, *Automated Deduction - CADE-11, 11th Int'l. Conf. on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings*, volume 607 of *LNCS*, pages 370–384. Springer-Verlag, 1992.

[30] M. Fisher. A normal form for temporal logic and its application in theorem-proving and execution. *Journal of Logic and Computation*, 7(4):429–456, Aug. 1997.

[31] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, Jan. 2001.

[32] T. French. A proof of the completeness of PLTL. Available as `http://www.cs.uwa.edu.au/~tim/papers/pltlcomp.ps`, 2000.

[33] 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.

[34] D. M. Gabbay, M. Finger, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, Volume 2*. Number 40 in Oxford Logic Guides. Oxford University Press, 2000.

[35] D. M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, Volume 1*. Number 28 in Oxford Logic Guides. Clarendon Press, 1994.

[36] R. Goldblatt. *Logics of Time and Computation*, volume 7 of *CSLI Lecture Notes*. CLSI/SRI International, Menlo Park, California, 1987.

[37] 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.

[38] R. H. Hardin, R. P. Kurshan, S. K. Shukla, and M. Y. Vardi. A new heuristic for bad cycle detection using BDDs. *Formal Methods in System Design*, 18(2):131–140, 2001.

[39] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. Reidel Publishing Company, Dordrecht, 1st edition, 1984.

[40] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, Massachusetts, 2000.

[41] D. Harel, D. Kozen, and J. Tiuryn. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 99–217. Kluwer Academic Publishers, Dordrecht, 2nd edition, 2002.

[42] K. Heljanko, T. A. Junttila, and T. Latvala. Incremental and complete bounded model checking for full PLTL. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*, pages 98–111, 2005.

[43] J. G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. Technical Report RS-97-8, BRICS, Department of Computer Science, University of Aarhus, Aarhus, Denmark, Apr. 1997. Available at `http://www.brics.dk/RS/97/8/`.

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

[45] Y. Hollander, M. Morley, and A. Noy. The *e* language: A fresh separation of concerns. In *Technology of Object-Oriented Languages and Systems (Proc. 38th Int'l. TOOLS Conference, TOOLS Europe 2001)*, pages 41–50. IEEE Computer Society Press, Mar. 2001.

[46] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, London, 1996.

[47] IEEE Standards Association. `http://standards.ieee.org/`.

[48] IEEE Standard 1647. Produced by the **e** Functional Verification Language Working Group. `http://www.ieee1647.org/`.

[49] Interval Temporal Logic (ITL) homepage. `http://www.cse.dmu.ac.uk/STRL/ITL/`.

[50] J. A. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

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

[52] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 789–840. Elsevier Science Publishers, Amsterdam, 1990.

[53] F. Kröger. *Temporal Logic of Programs*, volume 8 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1987.

[54] T. Kropf. *Introduction to Formal Hardware Verification*. Springer-Verlag, Heidelberg, Germany, 1999.

[55] M. Lange and C. Stirling. Focus games for satisfiability and completeness of temporal logic. In *Proc. 16th Ann. IEEE Symp. on Logic in Computer Science (LICS 2001)*, pages 357–365, Boston, Mass., USA, June 2001. IEEE Computer Society Press.

[56] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proc. 17th Ann. IEEE Symp. on Logic in Computer Science (LICS 2002)*, pages 383–392, Washington, D.C., USA, 2002. IEEE Computer Society Press.

[57] O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000. Available at `http://www3.oup.co.uk/igpl/Volume_08/Issue_01/#Lichtenstein`.

[58] 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.

[59] Z. Manna and A. Pnueli. Verification of concurrent programs: the temporal framework. In R. S. Boyer and J. S. Moore, editors, *The Correctness Problem in Computer Science*, pages 215–273, New York, 1981. Academic Press.

[60] Z. Manna and A. Pnueli. The anchored version of the temporal framework. In J. W. D. Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency (REX Workshop 1988)*, volume 354 of *LNCS*, pages 201–284. Springer-Verlag, 1989.

[61] M. Marx, S. Mikulas, and M. Reynolds. The mosaic method for temporal logics. In R. Dyckhoff, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2000, St Andrews, Scotland, UK, July 3-7, 2000, Proceedings*, volume 1847 of *LNCS*, pages 324–340. Springer-Verlag, 2000.

[62] K. L. McMillan. *Symbolic model checking*. Kluwer Academic Publishers, Boston, Mass., 1993.

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

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

[65] 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.

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

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

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

[69] B. 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.

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

[71] B. Moszkowski. Compositional reasoning using Interval Temporal Logic and Tempura. In W.-P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference*, volume 1536 of *LNCS*, pages 439–464, Berlin, 1998. Springer-Verlag.

[72] B. Moszkowski. An automata-theoretic completeness proof for Interval Temporal Logic (extended abstract). In U. Montanari, J. Rolim, and E. Welzl, editors, *Proc. 27th Int'l. Colloquium on Automata, Languages and Programming (ICALP 2000)*, volume 1853 of *LNCS*, pages 223–234, Geneva, Switzerland, July 2000. Springer-Verlag.

[73] B. 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 eds.

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

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

[76] The Perl programming language. http://www.perl.org.

[77] PerlDD: Perl extensions to CUDD [24]. Available at http://vlsi.colorado.edu/~fabio.

[78] A. 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.

[79] V. R. Pratt. Process logic. In *Proc. Sixth Ann. ACM Symp. on Principles of Programming Languages*, pages 93–100. ACM, 1979.

[80] A. Prior. *Past, Present and Future*. Oxford University Press, London, 1967.

[81] PSL/Sugar Consortium. http://www.pslsugar.org.

[82] R. Pucella. Logic column 11: The finite and the infinite in temporal logic. *SIGACT News*, 36(1):86–99, 2005. Available at Computing Research Repository (CoRR): http://arxiv.org/abs/cs.LO/0502031.

[83] N. Rescher and A. Urquhart. *Temporal Logic*. Springer-Verlag, New York, 1971.

[84] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, July 1985.

[85] SystemVerilog website. http://www.systemverilog.org.

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

[87] 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.

[88] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In A. Meyer, editor, *Proc. 1st Ann. IEEE Symp. on Logic in Computer Science (LICS '86)*, pages 322–331. IEEE Computer Society Press, June 1986.

[89] M. Y. Vardi and P. L. Wolper. Reasoning about infinite computations. *Inf. and Control*, 115(1):1–37, 15 Nov. 1994.

[90] Verisity Ltd. (acquired by Cadence Design Systems [16] in 2005). http://www.cadence.com/verisity/.

[91] Verisity Ltd. Semantics of temporal ***e***. Revised version of Morley [63]. Available from website of IEEE candidate standard 1647 as http://www.ieee1647.org/downloads/temporale_denotational.pdf, Dec. 2003.

[92] P. Wolper. Temporal logic can be more expressive. In *Proc. 22nd Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 340–348, Nashville, Tennessee, Oct. 1981. IEEE Computer Society Press.

[93] P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In *Lectures on Formal Methods in Performance Analysis (First EEF/Euro Summer School on Trends in Computer Science)*, volume 2090 of *LNCS*, pages 261–277. Springer-Verlag, July 2001.

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

[95] P. L. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 110–111:119–136, 1985.

[96] P. L. Wolper, M. Y. Vardi, and A. P. Sistla. Reasoning about infinite computation paths. In *Proc. 24th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 185–194, Tucson, Arizona, Nov. 1983. IEEE Computer Society Press.

[97] A. Xie and P. A. Beerel. Implicit enumeration of strongly connected components. In *ICCAD '99: Proc. of the 1999 IEEE/ACM Int'l. Conf. on Computer-Aided Design*, pages 37–40, Piscataway, NJ, USA, 1999. IEEE Press.