A PROPOSITIONAL DENSE TIME LOGIC

(Based on nested sequences)

Mohsin Ahmed and G. Venkatesh
Department of Computer Science and Engineering,
Indian Institute of Technology, Bombay 76, India.
Email: {mosh,gv}@cse.iitb.ernet.in

Abstract

This paper extends propositional linear time temporal logic (PTL) to propositional dense time logic (PDTL). While a PTL model is a single sequence of states, a PDTL model, called an omega-tree, consists of a nested sequence of states. Two new operators, called *within* and *everywhere* are introduced to access nested sequences. Besides its application in describing activities for Artificial Intelligence, PDTL can be used to represent more naturally procedural abstractions in control flow. PDTL is shown to be decidable by a tableau based method, and a complete axiomatization is given.

PDTL's omega tree models allow a dense mix of events. By imposing a stability condition on the propositions we get a subset of the omega tree models called ordinal trees which are free of dense mix. This logic called Propositional Ordinal Tree Logic (POTL) is also shown to be decidable in exponential time. Ordinal tree models though based on dense points, represent interval based information which maybe refined to any finite level. Hence POTL is a good bridge between point based and interval based temporal logics.

Ordinal trees can be easily embedded as a temporal data structure in a conventional logic programming language and thus provide a framework for temporal logic programming.
Keywords: Temporal logic, dense time, ordinal trees.

1 Introduction

There is frequent need to qualify information with time, especially if we are talking about the real world and the events taking place in it. Consider, for example, the statement: "The ball hit the wall before rolling off into the drain." This describes a sequence of events, the first of which consists of the ball hitting the wall, the last event describes the ball falling into the drain, and the events in between describe the ball rolling. A convenient way of describing this information is to say what happened in each of three consecutive intervals: one for striking, one for rolling and one for falling. The rolling interval can be further refined to describe what occurs within it. While a rolling interval may not be particularly eventful, we could instead consider the example "After hitting the wall, the ball bounced on the ground before falling into the drain." Here, the intervening interval consists of a unknown number of bouncing events.

While there have been a number of studies in representing temporal information within a logical framework [6, 10, 12, 13, 19], many of these deal with a time model which is discrete (i.e. consists of a sequence of states). This does not allow us to refine the time between two states. In particular, we cannot interpolate an unbounded (or infinite) sequence of states between two states. Such a requirement also arises when we wish to compose specifications of systems [12]. This paper does not focus on this issue, but instead looks at the issue of representing dense time information.

We present a temporal logic based on a model of time which allows arbitrary nesting of sequences. To be more precise, our model consists of a sequence of states at the top level. Between any two states of a sequence, another sequence of states can be interpolated. This model can be represented as an infinite binary tree, and is called the *omega tree* model.

Two new operators are introduced into propositional linear time temporal logic (PTL) [10]. The new operators $\oslash$ (read *within*) and $\boxplus$ (read *everywhere*), allow us to refine an interval and describe it in more detail. The dual of $\boxplus$ is $\diamondplus$ (read *somewhere*). This gives us the ability to represent dense time information. A behavior consisting of a sequence of events terminating in a limit point can also be expressed, for example a bouncing ball coming to rest can be represented by:

$$\oslash(\oslash Up \wedge \Box(\oslash Up \leftrightarrow \bigcirc \oslash Down)) \wedge (\bigcirc \boxplus Down) \wedge \boxplus(Down \leftrightarrow \neg Up),$$

where Up and $Down$ are propositions. This formula can be read as: "Within the first interval the ball is initially up, and it alternates between up and down. From the next interval onwards it remain down, and at every time point the ball is either up or down." Such a sequence can be nested in a bigger sequence of events, as in the statement "Every time he hit the ball, the ball bounced and came to rest." This statement can represented by $\Box(Hit \rightarrow \phi)$, where ϕ is the formula given before, and Hit is a new propositional symbol.

We show that the resulting logic, called Propositional Dense Time Logic (PDTL) is decidable in exponential time by a tableau based decision procedure similar to that used for deciding the satisfiability of PTL formulas [13, 19]. A simple axiomatization for PDTL is given and its completeness is shown.

Though PDTL is reducible to the Deterministic Propositional Dynamic Logic (D-PDL) [5] and hence its decidability follows from that of D-PDL [11], PDTL's *within* and *next* operators have a direct temporal interpretation in terms of nested sequences. Moreover the tableau based method is more appealing intuitively, and is a direct extension of the methods used for PTL [13, 19].

PDTL allows us to write a satisfiable sentence: $\boxplus(\diamondplus P \wedge \diamondplus \neg P)$, which describes a dense mix of states that satisfy P and $\neg P$. Such formulas contribute to unwieldy models and rarely occur in practice. To filter out such models, we define *stability* of propositions. Call a proposition p *stable* in an omega tree if there is no dense mixture of p and $\neg p$ in the omega tree. *Ordinal Trees* are omega trees where all the propositions are stable, and its logic is called Propositional Ordinal Tree Logic (POTL).

Ordinal trees are omega trees, which can be represented by finite binary trees with back-arcs allowed in a restricted way so that infinitely-nested sequences do not appear. The logic of formulas valid over ordinal trees (POTL) is shown to be decidable by extending PDTL's tableau based procedure. Here nodes which do not contain any ordinal tree models are also closed. Thus POTL is also decidable in exponential time, as compared to deciding it in non-elementary recursive time by interpreting it in *S2S* of Rabin [15, 16].

An ordinal tree can represent a nested infinite sequences of events such as $a \cdot b^{\omega} \cdot c^{\omega^2} \cdot b$, where a, b and c are some events, and a^k denotes a occurring k times. In [2] we describe an extension to prolog that uses ordinal trees for temporal logic programming. For example, the temporal query: "Was the janitor absent on Wednesday?", can be positively answered from the causal-rule: "Whenever it rained, the janitor was absent the next day." and the temporal-fact: "It rains every Tuesday."

Discrete time temporal logic programming has also been investigated by several researchers[1, 4, 8, 9, 14]. The theory of dense time logics has also been studied by a number of researchers [7, 17, 18]. Alur and Henzinger in [3] show a real time logic to be undecidable. Our method differs from others, in that we provide for the first time a dense time logic (POTL) that is:

- Theoretically interesting, since it is decidable in exponential time by a simple extension of the tableau method for PTL.
- Practically interesting, as it gives rise to a temporal data structure that is useful in a logic programming language[2].

Organization of the paper

Section 2 defines the language and section 3 defines the models and semantics of PDTL, along with some examples. The decidability of PDTL is covered in section 4 and section 5 covers the axiomatization and completeness of PDTL. In section 6 we look at ordinal trees models and their logic POTL, and section 7 gives a tableau based decision procedure for POTL. Finally section 8 concludes the paper.

2 The Language

Propositional Dense Time Logic (PDTL) is an extension of the usual PTL. The language of PDTL contains the usual propositional logic symbols: truth constants $\mathcal{T}$ (true) and $\mathcal{F}$ (false), set of propositions $PROP = \{p_1, \ldots, p_n\}$, logical connectives: $\neg$ (not), $\rightarrow$ (implies), the usual temporal operators: $\Box$ (henceforth), $\bigcirc$ (next), with the following new operators: $\boxplus$ (everywhere) and $\oslash$ (within). The *until* operator of PTL [10] is not considered in this paper.

Define the language of PDTL to be a set L_{PDTL} of formulas generated by L:

$$L ::= \mathcal{F} | p_1 | \ldots | p_n | \neg L | L \rightarrow L | \Box L | \bigcirc L | \boxplus L | \oslash L$$

The new syntactic definitions are: $\Diamond\phi \overset{\text{def}}{=} \neg\Box\neg\phi$, $\diamondplus\phi \overset{\text{def}}{=} \neg\boxplus\neg\phi$.

Notation: $\mathcal{N}$ is the set of natural numbers. $\mathcal{N}^+$ is the set $\{(k_0, \ldots, k_n) | \, k_i \in \mathcal{N}, \, 0 \leq i \leq n\}$, and $\mathcal{N}^* = \mathcal{N}^+ \cup \{()\}$. An element of $\mathcal{N}^*$ will be denoted by $\overline{k} = (k_0, \ldots, k_n)$. $\overline{0}_i$ will denote an i-tuple of zeroes. The concatenation of $\overline{k}$ with i will be denoted by $\overline{k} \cdot i = (k_0, \ldots, k_n, i)$. The concatenation of $\overline{k}$ with $\overline{m} = (m_0, \ldots, m_j)$, will be $\overline{k} \cdot \overline{m} = (k_0, \ldots, k_n, m_0, \ldots, m_j)$.

3 Semantics

While the models of PTL are based on the natural numbers, the models of PDTL are based on nested sequences of natural numbers. The language and the semantics of PTL are extended so that we can talk about the truth of formulas in such a model.

A state is a subset of *PROP*. A ω-sequence of states (the usual model for PTL [10, 13]) is a sequence of states indexed by elements of $\mathcal{N}$. The model for PDTL is an infinitely nested sequence of states indexed by $\mathcal{N}^+$.

Definition 1 (Omega Tree) *An* omega-tree *model for PDTL is an infinite binary tree $T = (\mathcal{N}^+, 0, w, x, s)$, rooted at (0), with two successor functions: $w : \mathcal{N}^+ \mapsto \mathcal{N}^+$*

(within) *and* $x : \mathcal{N}^+ \mapsto \mathcal{N}^+$ (next)*, The valuation function* $s : \mathcal{N}^+ \mapsto 2^{PROP}$*, maps its nodes* ($\mathcal{N}^+$) *to subsets of PROP.*

The w-child of $(\overline{k})$ is $(\overline{k} \cdot 0)$ and the x-child of $(\overline{k} \cdot i)$ is $(\overline{k} \cdot i + 1)$, see figure 1. The omega tree rooted at $(k_0, \ldots, k_n)$ will be denoted by $T((k_0, \ldots, k_n))$.

$$\begin{tikzcd}
& & & \boxed{s((0))} \arrow[dll, "w"'] \arrow[drr, "x"] & & & \\
& \boxed{s((0,0))} \arrow[dl, "w"'] \arrow[dr, "x"] & & & & \boxed{s((1))} \arrow[dl, "w"'] \arrow[dr, "x"] & \\
\boxed{s((0,0,0))} & & \boxed{s((0,1))} & & \boxed{s((1,0))} & & \boxed{s((2))} \\
\cdots & & \cdots & & \cdots & & \cdots
\end{tikzcd}$$

Figure 1: Omega-tree

The sequence of states $\ll s((0)), s((1)), \ldots \gg$ occurs as in a PTL model, but here the sequence of states $\ll s((0,0)), s((0,1)), \ldots \gg$ is a nested sequence between the states $s((0))$ and $s((1))$. This sequence is accessible from $s((0))$ by the $\oslash$ operator. The states can be temporally ordered as follows:

$$\begin{array}{ll}
s(\overline{k}) =_t s(\overline{k} \cdot 0) & \overline{k} \in \mathcal{N}^+ \\
s(\overline{k} \cdot i) \leq_t s(\overline{k} \cdot i \cdot \overline{m}) <_t s(\overline{k} \cdot i + 1) & \overline{k} \in \mathcal{N}^*,\ \overline{m} \in \mathcal{N}^+,\ i \geq 0
\end{array}$$

Since we identify the time point $s(\overline{k})$ with $s(\overline{k} \cdot 0)$, $s(\overline{k}) = s(\overline{k} \cdot 0)$. Truth of formulas in T is defined as follows:

$$\begin{array}{lll}
T \models \phi & \overset{\text{def}}{=} & T((0)) \models \phi \\
(T(\overline{k}) \models p),\ p \in PROP & \text{iff} & p \in s(\overline{k}) \\
T(\overline{k}) \not\models \mathcal{F} & & \\
T(\overline{k}) \models \neg\phi & \text{iff} & \text{not } (T(\overline{k}) \models \phi) \\
T(\overline{k}) \models \phi \rightarrow \psi & \text{iff} & \text{not } (T(\overline{k}) \models \phi) \text{ or } (T(\overline{k}) \models \psi) \\
T((k_1, \ldots, k_n)) \models \bigcirc\phi & \text{iff} & T((k_1, \ldots, k_{(n-1)}, k_n + 1)) \models \phi \\
T((k_1, \ldots, k_n)) \models \Box\phi & \text{iff} & \forall j \geq 0,\ T((k_1, \ldots, k_{(n-1)}, k_n + j)) \models \phi \\
T((k_1, \ldots, k_n)) \models \Diamond\phi & \text{iff} & \exists j \geq 0,\ T((k_1, \ldots, k_{(n-1)}, k_n + j)) \models \phi
\end{array}$$

$\oslash\phi$ asserts that ϕ holds in the w-child of the current state, and $\boxplus\phi$ asserts that ϕ holds everywhere below the current state in the omega-tree model.

$$\begin{array}{lll}
T(\overline{k}) \models \oslash\phi & \text{iff} & T(\overline{k} \cdot 0) \models \phi \\
T((k_1, \ldots, k_n)) \models \boxplus\phi & \text{iff} & \forall j \geq 0,\ \forall \overline{m} \in \mathcal{N}^*,\ T((k_1, \ldots, k_{(n-1)}, k_n + j) \cdot \overline{m}) \models \phi \\
T((k_1, \ldots, k_n)) \models \diamondplus\phi & \text{iff} & \exists j \geq 0,\ \exists \overline{m} \in \mathcal{N}^*,\ T((k_1, \ldots, k_{(n-1)}, k_n + j) \cdot \overline{m}) \models \phi
\end{array}$$

Examples: In the examples below, we assume that time is divided into days at the top level so that the intervals $[s_0, s_1), [s_1, s_2), \ldots$ represent days.

'It is always hot': $\boxplus Hot$. 'There will be rain': $\diamondplus Rain$. 'The rain will last over a time interval': $\diamondplus \oslash \boxplus Rain$. 'It rains daily': $\Box \oslash \diamondplus Rain$. This does not mean that it rains the whole day, but that on everyday there is rain. In fact it is consistent with the next example. 'It never rains for a full day': $\Box \oslash \diamondplus \neg Rain$. 'Everyday the rains are followed by a flood': $\Box \oslash \diamondplus (\oslash \diamondplus \boxplus Rain \wedge \bigcirc \diamondplus \boxplus Flood)$. 'If he takes a walk everyday, then someday he will get wet in the rain': $\Box \oslash \diamondplus \mathit{Walk} \rightarrow \Diamond \oslash \diamondplus(\mathit{Walk} \wedge \mathit{Rain})$. Note that this is not a theorem. Consider in contrast the next example. 'Even though it rains daily, it is possible for him to take a walk everyday and not get wet in the rain': $\Box \oslash (\diamondplus \mathit{Walk} \wedge \diamondplus \mathit{Rain} \wedge \boxplus(\mathit{Walk} \rightarrow \neg \mathit{Rain}))$.

PDTL offers us an interesting way of modeling qualitative temporal information 'event p occurred *many* times' in terms of $\oslash \Box p$, which deliberately identifies 'many occurrences' with 'infinitely many occurrences in finite time' [2]. We could also use this logic to reason more clearly about program executions. To represent that a property ϕ holds after one statement is executed we write $\bigcirc \phi$. If the statement is an abstraction of further statements (e.g. a procedure call), then the properties during the call can be written as $\oslash \bigcirc^i \psi$.

4 Satisfiability and Tableaux

Given a formula ϕ, we give a tableau method [13, 19] of deciding whether it is satisfiable or not. A tableau is a rooted directed graph, containing two kinds of nodes. A *state-Node* is a node containing only *literals* (propositions and their negations) and formulas of types: $\bigcirc \phi$ or $\oslash \phi$. A state-node η has two subnodes: the *within-subnode* $[\eta]_w$ and the *next-subnode* $[\eta]_x$. A empty-node is considered to be a state node. All other nodes are *logical-nodes*. A logical-node η has two subnodes $[\eta]_l$ and $[\eta]_r$.

4.1 Tableaux Building

We build a tableau for ϕ starting from the root node $\{\phi\}$. A leaf-node is expanded according to the type of formulas in it. On expansion, the leaf nodes generate subnodes to which it is linked by outgoing arcs. If a new subnode has the same formulas as an existing node μ, then the new sub-node is not created, instead η is linked to μ.

In a logical-leaf-node η, pick any expandable formula and call it the *principal formula* of η (denoted by $Pr(\eta)$). This formula will be expanded to generate subnodes according to table 1. Other formulas in η, called the side formulas are then copied unchanged to the subnodes. A state-leaf-node generates two subnodes:

	$Pr(\eta) \in \eta$	$[\eta]_l$, $([\eta]_r = \mathcal{F})$
1	$\Box A$	$A, \bigcirc \Box A$
2	$\Diamond A$	$A \vee \bigcirc \Diamond A$
3	$\boxplus A$	$A, \bigcirc \boxplus A, \oslash \boxplus A$
4	$\diamondplus A$	$A \vee (\bigcirc \diamondplus A \vee \oslash \diamondplus A)$
5	$\neg \bigcirc A$	$\bigcirc \neg A$
6	$\neg \oslash A$	$\oslash \neg A$
7	$\neg \Box A$	$\Diamond \neg A$
8	$\neg \Diamond A$	$\Box \neg A$
9	$\neg \boxplus A$	$\diamondplus \neg A$
10	$\neg \diamondplus A$	$\boxplus \neg A$

	$Pr(\eta) \in n$	$[\eta]_l$	$[\eta]_r$
11	$A \rightarrow B$	$\neg A$	B
12	$A \wedge B$	A, B	$\mathcal{F}$
13	$A \vee B$	A	B
14	$A \leftrightarrow B$	$\neg A, \neg B$	A, B
15	$\neg(A \rightarrow B)$	$A, \neg B$	$\mathcal{F}$
16	$\neg(A \wedge B)$	$\neg A$	$\neg B$
17	$\neg(A \vee B)$	$\neg A, \neg B$	$\mathcal{F}$
18	$\neg(A \leftrightarrow B)$	$\neg A, B$	$A, \neg B$
19	$\neg\neg A$	A	$\mathcal{F}$

Table 1: Expansion table for logical nodes

$[\eta]_w = \{\phi \mid \oslash \phi \in \eta\} \cup \{\mathit{literal} \in \eta\}$ and $[\eta]_x = \{\phi \mid \bigcirc \phi \in \eta\}$.

Let S be the set of formulas that appear in the tableau of $\{\phi\}$. We have $|S| < (4\,|\phi|)$, so number of nodes $< 2^{|S|} < 2^{4|\phi|}$ and the tableau expansion eventually halts.

4.2 Closing Nodes

A node is *closed* when we can show it is unsatisfiable, that is we can prove the negation of the node. A node which is not closed is called *open*. An empty node is considered open. If a node contains both A and $\neg A$ (or $\mathcal{F}$), then it is obviously unsatisfiable and therefore closed. However it is more difficult to give conditions for closing a node containing an unsatisfiable formula of the form $\Diamond A$ or $\blacklozenge A$.

If $\Diamond A$ is unsatisfiable at a node, it will be carried over only to every x-descendant of that node, that is, it will be indefinitely postponed. However if $\blacklozenge A$ is unsatisfiable at a node, it will be carried over separately to both its x and w descendant. Thus, we should close a node containing $\Diamond A$, if $\neg A$ holds at every node reachable from it by logical and x arcs. Similarly we close a node containing $\blacklozenge A$, if $\neg A$ holds at every reachable node.

Define R, Rx and Rxw to be the set of open descendants of a node as follows:

$R(\eta)$	$=$	$\{\eta\} \cup Rx([\eta]_l) \cup Rx([\eta]_r)$	if η is a logical node
$R(\eta)$	$=$	$\{\eta\}$	otherwise
$Rx(\eta)$	$=$	$\{[\eta]_l, [\eta]_r\} \cup Rx([\eta]_l) \cup Rx([\eta]_r)$	if η is a logical node
$Rx(\eta)$	$=$	$\{[\eta]_x\} \cup Rx([\eta]_x)$	if η is a state node
$Rwx(\eta)$	$=$	$\{[\eta]_l, [\eta]_r\} \cup Rwx([\eta]_l) \cup Rwx([\eta]_r)$	if η is a logical node
$Rwx(\eta)$	$=$	$\{[\eta]_x, [\eta]_w\} \cup Rwx([\eta]_w) \cup Rwx([\eta]_x)$	if η is a state node
x-leaf-scc(η)	iff	$\forall\mu(\mu \in Rx(\eta) \rightarrow \eta \in Rx(\mu))$	
wx-leaf-scc(η)	iff	$\forall\mu(\mu \in Rwx(\eta) \rightarrow \eta \in Rwx(\mu))$	

$R(\eta)$ is the set of logical descendants of η, that is, nodes reachable from η through logical arcs. $Rx(\eta)$ is the set of nodes reachable from η by a path of *logical* and *next* arcs; this set is a maximal x-strongly connected component when x-leaf-scc(η) is true. Similarly $Rwx(\eta)$ is the set of nodes reachable from η by any path, and this set is a wx-strongly connected component when wx-leaf-scc(η) is true.

Closing Algorithm:

```
repeat
    1. Construct $Rx(\eta)$ and $Rwx(\eta)$ for each open node $\eta$.
    2. Close any node $\eta$ that satisfies any one of the conditions:
        2.1. $\psi, \neg\psi \in \eta$, (or $\mathcal{F} \in \eta$).
        2.2. All the logical children of $\eta$ are closed.
        2.3. Either $[n]_x$ or $[n]_w$ is closed.
        2.4. $\Diamond\psi \in \eta$ and $\psi \notin \bigcup Rx(\eta) \cup \eta$
        2.5. $\blacklozenge\psi \in \eta$ and $\psi \notin \bigcup Rwx(\eta) \cup \eta$
    until no more nodes can be closed.
```

Theorem 1 *$\{\phi\}$ is open iff it is satisfiable.*

Proof: ($\Leftarrow$) Let $\{\phi\}$ be closed, we show that it is unsatisfiable. Suppose $\{\phi\}$ is closed, later we show $\vdash \neg\phi$ in an axiom system Ax_{PDTL}. By soundness of the

Ax_{PDTL}, we have $\models \neg\phi$. Therefore ϕ is unsatisfiable. ∎

($\Rightarrow$) For the purpose of building models we need only the open state nodes. The tableau τ is shortened to a ***state node tableau*** Δ by removing the logical nodes as follows:

- The nodes of Δ are the open state nodes of τ.
- For $\mu_1, \mu_2 \in \Delta$, if $\mu_2 \in R([\mu_1]_x)$ is in τ, then an x arc is drawn from μ_1 to μ_2, similarly if $\mu_2 \in R([\mu_1]_w)$ then a w arc is drawn from μ_1 to μ_2. Note that a node in Δ may have many x and w children.

A omega tree model for the formulas in a node η can be extracted from the set of open state nodes in Δ. Define the mapping $m : \mathcal{N}^+ \to \Delta$ as follows:

```
Let $m((0)) := \eta, \quad \eta \in R(\{\phi\})$
If $m(\overline{k} \cdot k') = \mu$ then
    If not $x$-leaf-scc($\mu$) then
        Let $m(\overline{k} \cdot k' + 1)$ be any $x$-child of $\mu$, and $m(\overline{k} \cdot k' \cdot 0)$ be any $w$-child of $\mu$.
    If $x$-leaf-scc($\mu$) and not $wx$-leaf-scc($\mu$) then
        By the leaf condition there is a $x$-path $\ll \nu_0, \ldots, \nu_k \gg$ passing
        through every node in $Rx(\mu)$, such that $\mu = \nu_0 = \nu_k$, and
        $\nu_{j+1}$ is a x-child of $\nu_j$, $\quad 0 \leq j < k$.
        For $j := 0$ to $k$ do
            Let $m(\overline{k} \cdot k' + j) := \nu_j$
            and let $m(\overline{k} \cdot k' + j \cdot 0)$ be any $w$-child of $\nu_j$
    If $wx$-leaf-scc($\mu$) then
        By the leaf condition there is a $wx$-path $\ll \nu_0, \ldots, \nu_k \gg$ passing
        through every node in $Rwx(\mu)$, such that $\mu = \nu_0 = \nu_k$,
        and $\nu_{j+1}$ is a x-child or a w-child of $\nu_j$, for $0 \leq j < k$.
        Let $\overline{l} \cdot l' := \overline{k} \cdot k'$
        For $j := 0$ to $k - 1$ do
            if $\nu_{j+1}$ is a x-child of $\nu_j$ then
                Let $m(\overline{l} \cdot l' + 1) := \nu_{j+1}$ and let $m(\overline{l} \cdot l' \cdot 0)$ be any $w$-child of $\nu_j$.
                Let $\overline{l} \cdot l' := \overline{l} \cdot (l' + 1)$
            otherwise let
                $m(\overline{l} \cdot l' \cdot 0) := \nu_{j+1}$ and let $m(\overline{l} \cdot l + 1)$ be any $x$-child of $\nu_j$.
                Let $\overline{l} \cdot l' := \overline{l} \cdot l' \cdot 0$
```

The states of the omega tree can now be defined:

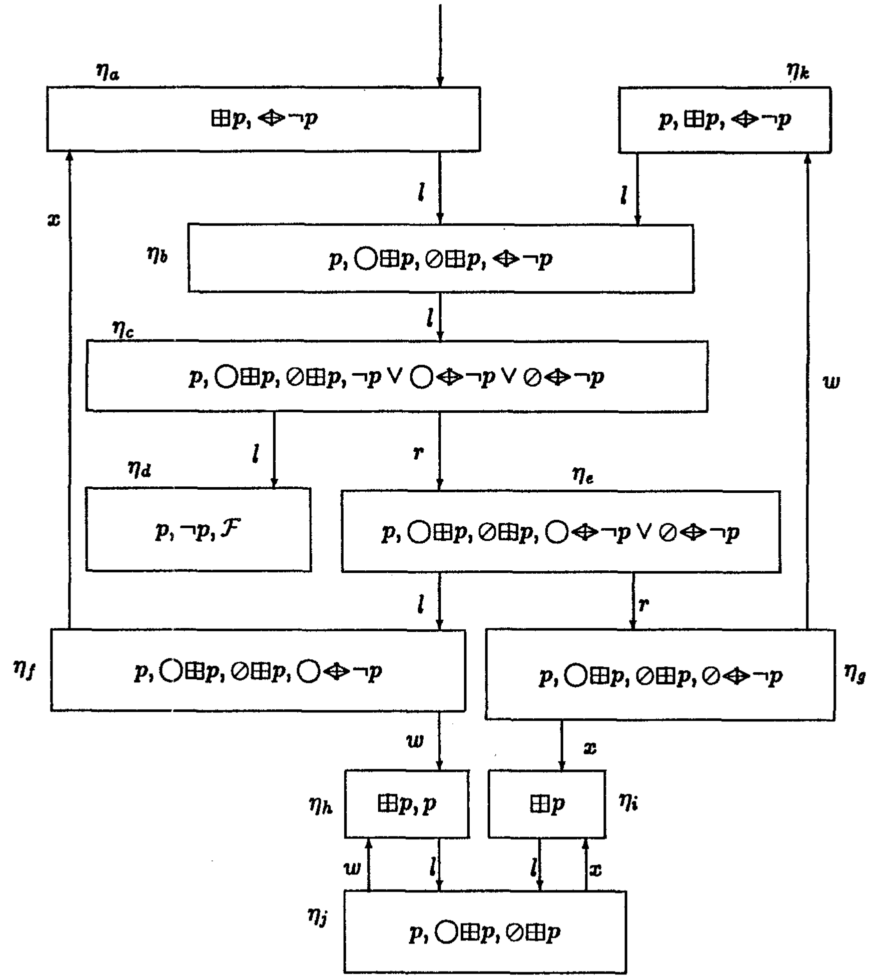
$$s(\overline{k}) \stackrel{\text{def}}{=} \bigcup_{i \geq 0} (m(\overline{k} \cdot \overline{0}_i) \cap PROP), \quad \overline{k} \in \mathcal{N}^+$$

It is easy to see that $T \models \phi$ by induction on the structure of ϕ. The important part is to verify that if $T(\overline{k} \cdot k') \models \Diamond\psi$ (respectively $T(\overline{k} \cdot k') \models \Leftrightarrow\psi$) then $T(\overline{k} \cdot k' + i) \models \psi$ for some $i \in \mathcal{N}$ (respectively $T(\overline{k} \cdot k' + i \cdot \overline{j}) \models \psi$ for some $i \in \mathcal{N}$ and $\overline{j} \in \mathcal{N}^*$). This can be shown using the fact $T(\overline{k} \cdot k')$ was constructed using every node of the x-leaf-scc (wx-leaf-scc respectively) [13]. ∎

Corollary 2 *PDTL is decidable in exponential time.*

Proof: The required strongly connected components can be constructed in time $O(|E| + |\tau|)$, where $|E|$ is the number of edges in τ. Therefore the whole tableau procedure takes $O(2^{c|\phi|})$ for some constant c. ∎

Example 1 The tableau for the unsatisfiable formula $\boxplus p \wedge \Diamond\neg p$ is given in figure 2. We close η_a because $\Diamond\neg p \in \eta_a$ and $\neg p \notin \bigcup Rwx(\eta_a) \cup \eta_a$.



η_a: $\boxplus p, \Diamond\neg p$
η_k: $p, \boxplus p, \Diamond\neg p$
η_b: $p, \bigcirc\boxplus p, \oslash\boxplus p, \Diamond\neg p$
η_c: $p, \bigcirc\boxplus p, \oslash\boxplus p, \neg p \vee \bigcirc\Diamond\neg p \vee \oslash\Diamond\neg p$
η_d: $p, \neg p, \mathcal{F}$
η_e: $p, \bigcirc\boxplus p, \oslash\boxplus p, \bigcirc\Diamond\neg p \vee \oslash\Diamond\neg p$
η_f: $p, \bigcirc\boxplus p, \oslash\boxplus p, \bigcirc\Diamond\neg p$
η_g: $p, \bigcirc\boxplus p, \oslash\boxplus p, \oslash\Diamond\neg p$
η_h: $\boxplus p, p$
η_i: $\boxplus p$
η_j: $p, \bigcirc\boxplus p, \oslash\boxplus p$
l
r
w
x

Figure 2: Tableau for $\{\boxplus p, \Diamond\neg p\}$

5 Completeness

Axioms (Ax_{PDTL}):

$Ax1$	$\Box(A \to B) \to (\Box A \to \Box B)$	$Ax2$	$\boxplus(A \to B) \to (\boxplus A \to \boxplus B)$
$Ax3$	$\bigcirc(A \to B) \to (\bigcirc A \to \bigcirc B)$	$Ax4$	$\oslash(A \to B) \to (\oslash A \to \oslash B)$
$Ax5$	$\bigcirc\neg A \leftrightarrow \neg \bigcirc A$	$Ax6$	$\oslash\neg A \leftrightarrow \neg \oslash A$
$Ax7$	$\Box A \to (A \wedge \bigcirc\Box A)$	$Ax8$	$\boxplus A \to (A \wedge \oslash\boxplus A \wedge \bigcirc\boxplus A)$
$Ax9$	$(A \wedge \Box(A \to \bigcirc A)) \to \Box A$	$Ax10$	$(A \wedge \boxplus(A \to (\bigcirc A \wedge \oslash A))) \to \boxplus A$
		$Ax11$	$p \leftrightarrow \oslash p, \quad \forall p \in PROP$

Rules of Inference:

$$PL: \frac{\vdash_{PL} A}{\vdash A}, \quad MP: \frac{\vdash A, A \to B}{\vdash B}, \quad RN: \frac{\vdash A}{\vdash \Box A}, \quad RE: \frac{\vdash A}{\vdash \boxplus A}$$

The rule PL allows all Propositional Logic tautologies to be theorems of PDTL.

Proposition 3 (Soundness) *The axioms are sound, and the rules of inference preserve soundness.*

Some Theorems of PDTL:

$T1$	$\vdash$	$\Box(A \wedge B) \leftrightarrow (\Box A \wedge \Box B)$	$T2$	$\vdash$	$\bigcirc(A \wedge B) \leftrightarrow (\bigcirc A \wedge \bigcirc B)$
$T3$	$\vdash$	$\boxplus(A \wedge B) \leftrightarrow (\boxplus A \wedge \boxplus B)$	$T4$	$\vdash$	$\oslash(A \wedge B) \leftrightarrow (\oslash A \wedge \oslash B)$
$T5$	$\vdash$	$\Box A \to \bigcirc A$	$T6$	$\vdash$	$\boxplus A \to (\oslash A \wedge \bigcirc A)$

Example 2 $\vdash \boxplus A \to \Box A$

Proof:

$\vdash (\boxplus A \wedge A) \to (\bigcirc\boxplus A \wedge \bigcirc A)$	1	$Ax8, T6$
$\vdash (\boxplus A \wedge A) \to \bigcirc(\boxplus A \wedge A)$	2	$T2$
$\vdash \Box((\boxplus A \wedge A) \to \bigcirc(\boxplus A \wedge A))$	3	$RN, 2$
$\vdash (\boxplus A \wedge A) \to \Box(\boxplus A \wedge A)$	4	$Ax9, 3$
$\vdash \Box(\boxplus A \wedge A) \to (\Box\boxplus A \wedge \Box A)$	5	$T1$
$\vdash (\boxplus A \wedge A) \to \Box A$	6	$PL, 4, 5$
$\vdash \boxplus A \to A$	7	$Ax8$
$\vdash \boxplus A \to \Box A$	8	$PL, 6, 7$ ∎

Theorem 4 *If $\{\phi_1, \ldots, \phi_m\}$ is closed, then $\neg(\phi_1 \wedge \cdots \wedge \phi_m)$ is provable from the axioms.*

Proof: We prove this in the order in which the nodes are closed, so that at each step the theorem holds for nodes closed at a earlier stage. We use N to denote the conjunction of formulas in a node η. Consider the ways in which a node η is closed:

1. Node η contains both ϕ and $\neg\phi$ (or $\mathcal{F}$), then it is unsatisfiable, and $\vdash_{PL} \neg N$.
2. Logical node η is closed, because all its children are closed. Let L and R denote the conjunction of formulas in $[\eta]_l$ and $[\eta]_r$ respectively. From the expansion table for a logical node we get $\vdash N \to (L \vee R)$. Rules 1-10 in the expansion table follow from the axioms and definitions for PDTL, and rules 11-19 follow from PL. Assuming $\vdash (\neg L, \neg R)$, we get $\vdash \neg N$.

3. A state node η is closed because $[\eta]_x$ was closed, let X be the conjunction of formulas in $[\eta]_x$. Assuming $\vdash \neg X$, and applying $RE, T6, Ax5$, we get $\vdash \neg \bigcirc X$. Using $\vdash N \rightarrow \bigcirc X$, we derive $\vdash \neg N$.
4. State node η is closed because $[\eta]_w$ was closed. Let L be the conjunction of literals in η, and W be the conjunction of other formulas in $[\eta]_w$. Assuming $\vdash \neg(L \wedge W)$, and applying $RE, T6, Ax4$ we get $\vdash \neg \oslash (L \wedge W)$. Applying of $Ax11, T4$ on $\vdash N \rightarrow (L \wedge \oslash W)$, we get $\vdash N \rightarrow \oslash(L \wedge W)$. Now we can derive $\vdash \neg N$.
5. A node η was closed because $\Diamond A \in \eta$ has been indefinitely postponed. Let $P = \bigvee_{\eta_i \in Rx(\eta)}(N_i)$. We first show $\vdash P \rightarrow \neg A$.
Let η' be an open node in $Rx(\eta)$ containing $\Diamond A$, it expands as
$$\begin{aligned} \eta' = \{\Diamond A, N'\} &\Rightarrow \quad \{A, N'\} \quad \vee \quad \{\bigcirc\Diamond A, N'\} \\ &\Rightarrow \bigvee_{i=1}^{m} \underbrace{\{A, N_i'\}}_{\alpha_i} \vee \bigvee_{i=1}^{m} \underbrace{\{\bigcirc\Diamond A, N_i'\}}_{\beta_i} \end{aligned}$$
where $N' = \bigvee_{i=1}^{i=m} N_i'$. Since $A \notin \cup Rx(\eta)$, all α nodes must be closed, and $\vdash N' \rightarrow \neg A$. Since this holds for all nodes in $Rx(\eta)$, $\vdash P \rightarrow \neg A$. Using the fact P is defined over a x-leaf-scc, we get $\vdash P \rightarrow \bigcirc P$, applying RN, $\vdash \Box(P \rightarrow \bigcirc P)$. Hence $\vdash N \rightarrow (P \wedge \Box(P \rightarrow \bigcirc P))$, applying the induction axiom $Ax9$ on it we get $\vdash N \rightarrow \Box P$. Now $Ax1$ gets us $\vdash N \rightarrow \Box\neg A$. Now $\vdash N \rightarrow \Diamond A$ leads to $\vdash \neg N$.
6. A node η was closed because $\diamondsuit A \in \eta$ has been indefinitely postponed. This can happen only if $\neg A$ holds in every descendant of η. Let
$$T \;=\; \bigvee_{\eta_i \in Rwx(\eta)}[\wedge(\eta_i - \{\diamondsuit A, \bigcirc\diamondsuit A, \oslash\diamondsuit A\})]$$
It is so chosen because we do not need $\diamondsuit A$ to prove $\neg A$ and we need to show $T \rightarrow (\bigcirc T \wedge \oslash T)$.
Let η' be an open node in $Rwx(\eta)$ containing $\diamondsuit A$, it expands as
$$\begin{aligned} \eta' = \{\diamondsuit A, N'\} &\Rightarrow \quad \{A, N'\} \vee \{\bigcirc\diamondsuit A, N'\} \vee \{\oslash\diamondsuit A, N'\} \\ &\Rightarrow \bigvee_{i=1}^{m} \underbrace{\{A, N_i'\}}_{\alpha_i} \vee \bigvee_{i=1}^{m} \underbrace{\{\bigcirc\diamondsuit A, N_i'\}}_{\beta_i} \vee \bigvee_{i=1}^{m} \underbrace{\{\oslash\diamondsuit A, N_i'\}}_{\gamma_i} \end{aligned}$$
Where $N' = \bigvee_{i=1}^{i=m} N_i'$. Proceeding as before $\vdash N' \rightarrow \neg A$.
Let μ be a beta (respectively gamma) node of η, then $\diamondsuit A$ will not carry over to $[\mu]_w$ (respectively $[\mu]_x$). To prove $\neg A$ in the nodes of the $[\mu]_w$ (respectively $[\mu]_x$) subtree, we can use the nodes of the subtree at a gamma (respectively beta) node $\mu' \in Rwx(\eta)$ with the same formula as μ and where $\diamondsuit A$ is carried over. Hence $\neg A$ is provable in the subtree of μ also.
Applying RE on: $\vdash T \rightarrow (\bigcirc T \wedge \oslash T)$ we get $\vdash \boxplus(T \rightarrow (\bigcirc T \wedge \oslash T))$. Using the induction axiom $Ax10$, we derive $\vdash N \rightarrow \boxplus T$. This along with the assumption $\vdash N \rightarrow \diamondsuit A$, gives $\vdash \neg N$. ∎

Corollary 5 (Completeness) $(\models \phi) \Rightarrow (\vdash \phi)$

6 Ordinal Trees and POTL

We consider a restricted set of omega tree models called ordinal trees and its logic: the Propositional Ordinal Tree Logic (POTL).

Definition 2 (Stability) *A proposition $p \in PROP$ is called* stable *in an omega tree T if $\exists n \geq 0, \forall (k_0, \ldots, k_n) \in \mathcal{N}^+, \forall j \geq 0, \forall \overline{m} \in \mathcal{N}^*$:*

$$p \in s(k_0, \ldots, k_n) \text{ iff } p \in s(k_0, \ldots, k_n + j, \overline{m})$$

Note that if a proposition p is stable in T then it follows:

$$\exists n \geq 0, \forall (k_0, \ldots, k_n) \in \mathcal{N}^+, \quad T((k_0, \ldots, k_n)) \models (\boxplus p \vee \boxplus \neg p)$$

Definition 3 (Recurrence) *A omega tree T is called recurring if:*

$$\forall \overline{k} \cdot k' \in \mathcal{N}^+, \exists k'' \geq k', m \geq 0, \ \forall i \geq 0, \quad T(\overline{k} \cdot (k'' + i)) = T(\overline{k} \cdot (k'' + i + m))$$

Definition 4 (Ordinal Tree) *A ordinal tree T model for POTL is an omega tree where all the propositions are stable in T and T is recurring.*

Ordinal trees are interval based rather than point based models. For example, the formula $\boxplus(\diamondsuit A \wedge \diamondsuit \neg A)$ has an omega tree model but no ordinal tree models. Figure 3 shows an ordinal tree for $PROP = \{p_1, p_2\}$. Ordinal trees are represented

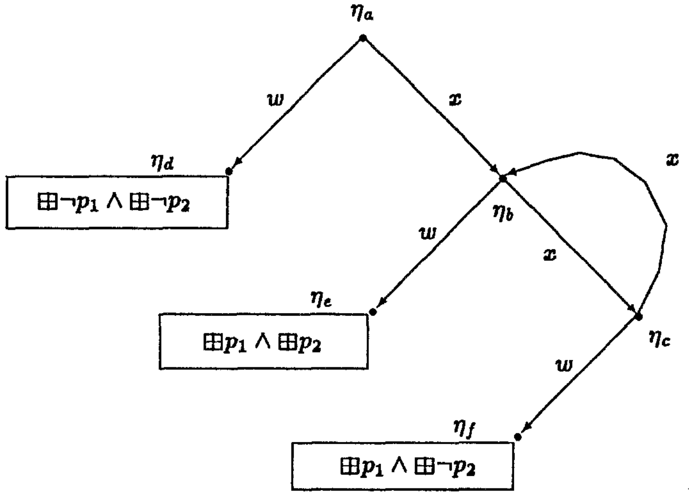

η_a
w
x
η_d
$\boxplus\neg p_1 \wedge \boxplus\neg p_2$
x
η_b
w
x
η_e
$\boxplus p_1 \wedge \boxplus p_2$
η_c
w
η_f
$\boxplus p_1 \wedge \boxplus\neg p_2$

Figure 3: An ordinal tree

as finite binary trees possibly with back-x-arcs, with the following properties:

1. A back arc from node η' to η is allowed only if $\eta' \in Rx(\eta)$.
2. Each non-leaf node has two outgoing arcs, (x and w-arcs).
3. Each leaf node η is labeled by a formula $\zeta \in \Sigma$, where

$$\Sigma \;=\; \{\boxplus l_1 \wedge \ldots \wedge \boxplus l_n \,|\, l_i = p_i \text{ or } l_i = \neg p_i, 0 < i \leq n\}$$

Note that each leaf node $[\eta]_w$ represents an interval between η and $[\eta]_x$ and hence an ordinal tree represents a nested sequence of intervals. An ordinal tree T can be unrolled to get an omega tree as follows:

- A back arc from η to η' in T is unrolled by copying $T(\eta')$ below η.
- A leaf node η is replaced by the unique PDTL tree for η.

7 Tableau for POTL

The PDTL tableau procedure will be extended to check if the formula ϕ has any ordinal tree models. When we build a PDTL tableau τ for ϕ, a state node η will now have a third child $[\eta]_s$, called its *stable* (s-) child, besides the two usual children $[\eta]_w$ and $[\eta]_x$. If $\eta = \{\bigwedge_i l_i, \bigwedge_j \oslash\phi_j, \bigwedge_k \bigcirc\psi_k\}$ then $[\eta]_s = \{\bigwedge_i \boxplus l_i, \bigwedge_j \phi_j\}$.

Define *Rsxw* descendants of a node as follows:

$$\begin{array}{rcl} Rswx(\eta) & = & \{[\eta]_l, [\eta]_r\} \cup Rswx([\eta]_l) \cup Rswx([\eta]_r), \text{ if } \eta \text{ is a logical node.} \\ Rswx(\eta) & = & \{[\eta]_x, [\eta]_w\} \cup Rswx([\eta]_w) \cup Rswx([\eta]_x) \\ & & \text{if } \eta \text{ is a state node, and } [\eta]_s \text{ is closed} \\ Rswx(\eta) & = & \{[\eta]_x, [\eta]_w, [\eta]_s\} \cup Rswx([\eta]_w) \cup Rswx([\eta]_x) \cup Rswx([\eta]_s) \\ & & \text{if } \eta \text{ is a state node, and } [\eta]_s \text{ is open} \\ swx\text{-leaf-scc}(\eta) & \text{iff} & \forall\mu(\mu \in Rswx(\eta) \rightarrow (\eta \in Rswx(\mu))) \text{ and} \\ & & \forall\mu(\mu \in Rswx(\eta) \wedge \mu \text{ is a state node } \rightarrow ([\mu]_s \text{ is open})) \end{array}$$

$Rswx(\eta)$ is the set of open nodes reachable from η by any path, and this set is a *swx*-strongly connected component (with every state node in it having three children) when swx-leaf-scc(η) is true. The closing algorithm is applied is the same as that for PDTL for the closing conditions (1) to (4), however condition (5) now reads:

Close η if swx-leaf-scc(η) and $\Diamond\!\!\!\diamond\psi \in \eta$ and $\psi \notin \bigcup Rswx(\eta)$.

and the closing of a s-child does not effect its parent state node.

7.1 Marking Nodes

Nodes that have an ordinal tree model will be marked by a sequence of functions $\beta_i, i \geq 0$ defined on τ. A node η will be called *marked* at stage i, if $\beta_i(\eta)$ is defined, that is $\beta_i(\eta) \neq \perp$. The marking will proceed bottom up: first marking nodes in all *swx*-leaf-sccs, and then proceeding upwards marking nodes that can form ordinal trees from the nodes already marked.

The s-children ensure that all *swx*-leaf-scc nodes have ordinal tree models. That is, for any node $\eta \in \tau$: swx-leaf-scc(η) implies that $\bigcup Rswx(\eta)$ does not contain both p and $\neg p$ for all $p \in PROP$, and therefore η has an ordinal tree model that satisfies the formula $\sigma(\eta)$ given below:

$$\sigma(\eta) = \bigwedge \left(\begin{array}{ll} \{\boxplus p : p \in (PROP \cap \bigcup Rswx(\eta))\} & \cup \\ \{\boxplus\neg p : p \in (PROP - \bigcup Rswx(\eta)),\} & \end{array} \right)$$

The marking function β_i for a large enough i, maps a node of the tableau to an ordinal tree model for that node. The range of β_i, $i \geq 0$ is the language $\mathcal{L}$ (for a particular τ) given below:

- For $\eta \in \tau$, if swx-leaf-scc(η) then $\eta \in \mathcal{L}$.
- If $t_1, t_2 \in \mathcal{L}$ then $t_1 \cdot t_2 \in \mathcal{L}$.
- If $t \in \mathcal{L}$ then $t^\omega \in \mathcal{L}$.

Each string in $\mathcal{L}$ represents an ordinal tree model, for example the ordinal tree in figure 3 is represented by $\eta_d \cdot (\eta_e \cdot \eta_f)^\omega$. Initially nodes in *swx*-leaf-sccs are marked: $\beta_0(\eta) = \eta$ if swx-leaf-scc(η), otherwise $\beta_0(\eta) = \perp$. The initial marking β_0 is extended to $\beta_i, 0 < i \leq |\tau|$, such that:

- If $\beta_i(\eta) \neq \perp$ then $\beta_{i+1}(\eta) = \beta_i(\eta)$.
- If $[\eta]_l$ or $[\eta]_r$ is marked, then η is marked in the next stage.
- If $[\eta]_w$ (or $[\eta]_s$) and $[\eta]_x$ are marked, then η is marked in the next stage.

- If every node in an x-leaf-scc has a $[\eta]_w$ (or $[\eta]_s$) child that is marked, then all the nodes in that x-leaf-scc are marked simultaneously in the next stage.

Theorem 6 *ϕ has an ordinal tree model iff $\beta_{|\tau|}(\{\phi\}) \neq \perp$, where $|\tau|$ is the size of the tableau τ for ϕ.*

Proof: ($\Leftarrow$) The mark of the node at stage i (i.e. $\beta_i(\eta)$) itself represents an ordinal tree model starting at that node.
($\Rightarrow$) Let T be an ordinal tree model satisfying ϕ, without loss of generality we can assume T is minimal, i.e. there are no ordinal tree models with fewer nodes than T satisfying ϕ. T can be embedded in τ, since τ contains all the information regarding all the models of ϕ. For each $\eta \in T$, let (η) be the corresponding node in τ. A leaf node of T will correspond to a node in a leaf node of τ. It is easy to see that for any $\eta \in T$, $(\eta) \in \tau$ will get marked by the marking algorithm by stage i, where i is the length of the largest path (without loops) from η to a *swx*-leaf-scc in T. ∎

Corollary 7 *The problem of testing whether a formula ϕ is satisfiable in POTL is decidable in exponential time.*

Proof: The complexity of the marking algorithm is $O(|E| + |\tau|)$, where $|E|$ is the number of edges in τ, $|\tau| \leq 2^{c|\phi|}$ and $|E| \leq 2^{2c|\phi|}$ for some constant c. ∎

Example 3 (POTL tableau) The formula $\boxplus(p \vee \neg p)$ has an ordinal tree model, the stable leaf nodes of its tableau (figure 4) are η_c and η_f. The tableau has been simplified for exposition. The marking of its nodes is given below:

	η_a	η_b	η_c	η_d	η_e	η_f
β_0	$\perp$	$\perp$	η_c	$\perp$	$\perp$	η_f
β_1	$(\eta_c \cdot \eta_f)^\omega$	$(\eta_f \cdot \eta_c)^\omega$	η_c	$\perp$	$\perp$	η_f
β_2	$(\eta_c \cdot \eta_f)^\omega$	$(\eta_f \cdot \eta_c)^\omega$	η_c	$\eta_c \cdot (\eta_c \cdot \eta_f)^\omega$	$\eta_f \cdot (\eta_f \cdot \eta_c)^\omega$	η_f

Example 4 The formula $\boxplus(p \vee \neg p) \wedge \boxplus\diamondplus p \wedge \boxplus\diamondplus\neg p$ has no ordinal tree models, because it has no *swx*-leaf-scc*s*. In particular the nodes η_c and η_f of figure 4 will be closed because η_c (correspondingly η_f) form a *swx*-leaf-scc which does not satisfy $\neg p$ (respectively p).

8 Conclusion

We have presented a propositional dense time logic which allows us to reason about nested sequences of events. We showed that the logic is decidable in exponential time by a tableau based method and presented a complete axiomatization for it.

We next looked at an interesting subset of models called ordinal trees, where all the propositions are stable. The logic of ordinal trees, POTL was also shown to be decidable in exponential time by extending the tableau based decision procedure used in the first part. Ordinal trees are interval based as they allow only finite level of refinement, and they seem to be a good bridge between point based and interval based temporal logics.

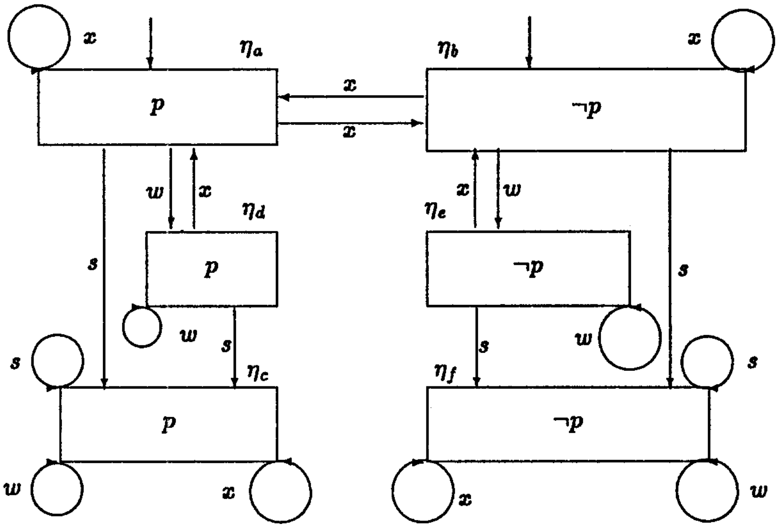


Figure 4: Ordinal tree tableau for $\boxplus(p \vee \neg p)$

Future work: We plan to axiomatize POTL and relate it to D-PDL with *converse*, *S2S* and other dense time logics. We will study if omega trees can be obtained as a limit of ordinal tree models.

We will also be considering logics dealing with finitely nested finite sequences of time points, i.e. we allow only finite subsequences of time points at each level. Each level of time could correspond to some natural periodic events, like 'hours', 'minutes', 'seconds' of a clock. For example, we could specify a day to consists of 24 hours, and make $\oslash \bigcirc^{24} \phi$ invalid or even better let $\oslash \bigcirc^{24} \phi \leftrightarrow \bigcirc \oslash \phi$. Such a representation is particularly useful in modeling synchronous digital circuits and in historical databases.

Acknowledgments: We thank Rohit Parikh, Madhavan Mukund, R. Ramanujam and Kamal Lodaya.

References

[1] M. Abadi and Z. Manna. Temporal logic programming. In *International Conference on Logic Programming San Fransisco, CA*, pages 4–16, 1987.

[2] M. Ahmed and G. Venkatesh. Dense time logic programming. In *Second Symposium on Logical Formalizations of Commonsense Reasoning, Austin, TX*, 1993.

[3] R. Alur and T. Henzinger. Real time logics: Complexity and expressiveness. In *Logic in Computer Science*, pages 390–401, 1990.

[4] M. Baudinet. Temporal logic programming is complete and expressive. In *16th POPL, Austin, TX*, pages 267–279, 1989.

[5] M. Ben-Ari, J. Y. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: Finite models, complexity and completeness. *Journal of Computer and System Sciences*, 25:402–417, 1982.

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

[7] J. P. Burgess. Basic tense logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume II, pages 89–133. D. Reidel, Dordrecht, Holland, 1984.

[8] D. Gabbay. Modal and temporal logic programming. In A. Galton, editor, *Temporal Logics and their applications*, pages 195–273. Academic Press, 1987.

[9] D. Gabbay. Modal and temporal logic programming - ii. In T. Dodd, editor, *Logic Programming*, pages 82–123. Intellect, Oxford, 1991.

[10] D. Gabbay, A. Pnueli, S. Shelah, and S. Stavi. The temporal analysis of fairness. In *7th POPL, Las Vegas, NE*, pages 163–173, 1980.

[11] D. Harel. Propositional dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume II, pages 507–544. D. Reidel, Dordrecht, Holland, 1984.

[12] L. Lamport. Temporal logic of actions. TR 57, Digital, 1990.

[13] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Proc. Conf. Logics of Programs*, pages 196–218. Springer Verlag, 1985. LNCS 193.

[14] B. C. Moszkowski. *Executing Temporal Logic Programs.* Cambridge University Press, Cambridge, 1986.

[15] M. O. Rabin. Decidability of second order theories and automata on infinite trees. *Transactions of AMS*, 141:1–35, July 1969.

[16] W. Thomas. Automata on infinite trees. In J. V. Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 165–186. North Holland, Amsterdam, 1990.

[17] J. F. A. K. van Benthem. *The Logic of Time.* D. Reidel, Dordrecht, Holland, 1983.

[18] J. F. A. K. van Benthem. Time, logic and computation. In J. deBakker, W. deRoever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models of Concurrency.* Springer Verlag, 1989. LNCS 354.

[19] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–93, 1983.