Temporal Logic

[Esoterica]

Michael Fisher
Department of Computer Science, University of Liverpool, UK

[MFisher@liverpool.ac.uk]

An Introduction to Practical Formal Methods Using Temporal Logic

What is Temporal Logic?

Now that we have seen both syntax and semantics for PTL, together with a variety of examples expressed in its language, we can reconsider the more philosophical question

"what is temporal logic?"

While we have given a formal description of the PTL logic, there are a number of alternative formalisations providing different, and interesting, ways to view PTL.

In the following we will briefly examine a few of these, as this is useful in shedding light on the precise nature of (propositional) temporal logic.

Sample PTL Axioms

$$\vdash \quad \Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$$

$$\vdash \quad (\mathbf{start} \Rightarrow \Box A) \Rightarrow \Box A$$

$$\vdash \quad \Box \neg A \Leftrightarrow \neg \Diamond A$$

$$\vdash \quad \bigcirc(A \wedge B) \Leftrightarrow \bigcirc A \wedge \bigcirc B$$

$$\vdash \quad A\mathcal{U}(B \wedge A\mathcal{U}C) \Rightarrow A\mathcal{U}C$$

Key axiom describing how this form of temporal logic works is the *induction axiom*:

$$\vdash \quad \Box(\varphi \Rightarrow \bigcirc\varphi) \;\Rightarrow\; (\varphi \Rightarrow \Box\varphi)$$

1: PTL as First-Order Logic

"PTL corresponds to a specific, decidable (PSPACE-complete) fragment of classical first-order logic (with arithmetic operations)."

As above,

$i \models$	**start**	is represented by	$i = 0$
$i \models$	$\bigcirc p$	is represented by	$p(i+1)$
$i \models$	$\Diamond p$	is represented by	$\exists j.\ (j \geq i) \wedge p(j)$
$i \models$	$\Box p$	is represented by	$\forall j.\ (j \geq i) \Rightarrow p(j)$

Example

Using the above translation,

$$\Box q \;\Rightarrow\; \Diamond q$$

now becomes

$$\forall i. \quad \big(\; (\forall j.\ (j \geq i) \Rightarrow q(j)) \quad \Rightarrow \quad (\exists k.\ (k \geq i) \wedge q(k)) \; \big)$$

Through some (first-order) logical manipulation this can be shown to be true as long as quantification occurs over a non-empty domain.

Since the domain here corresponds to the set of moments in time, then the domain should actually be infinite.

Induction Axiom Again

One way to view PTL logic is as a fragment of classical logic.

So, re-using the translations above:

$$\begin{array}{lcl} i \models \bigcirc p & \rightarrow & p(i+1) \\ i \models \Diamond p & \rightarrow & \exists j.\ (j \geq i) \wedge p(j) \\ i \models \Box p & \rightarrow & \forall j.\ (j \geq i) \Rightarrow p(j) \end{array}$$

The *Induction Axiom* can be translated as

$$[\forall i.\ \varphi(i) \Rightarrow \varphi(i+1)] \;\Rightarrow\; [\varphi(0) \Rightarrow \forall j.\ \varphi(j)]$$

which easily transforms into

$$[\varphi(0) \wedge \forall i.\ \varphi(i) \Rightarrow \varphi(i+1)] \;\Rightarrow\; \forall j.\ \varphi(j)$$

2: PTL Characterises Induction

"PTL captures a simple form of arithmetical induction."

Recall that the key axiom describing how PTL works is the induction axiom: $\vdash \; \Box(\varphi \Rightarrow \bigcirc\varphi) \;\Rightarrow\; (\varphi \Rightarrow \Box\varphi)$

As we saw above, this can be described as

$$[\varphi(0) \wedge \forall i.\ \varphi(i) \Rightarrow \varphi(i+1)] \;\Rightarrow\; \forall j.\ \varphi(j)$$

This should now be familiar as the *arithmetical induction principle*, i.e. if we can

1. show $\varphi(0)$, and
2. show that, for any i, if we already know $\varphi(i)$ then we can establish $\varphi(i+1)$

then $\varphi(i)$ is true for *all* elements, i, of the domain.

3: PTL as a Multi-Modal Logic

"PTL can be seen as a multi-modal logic, comprising two modalities, [1] and [∗], which interact closely."

In our syntax, '[1]' is usually represented as '$\bigcirc$', while '[∗]' is usually represented as '$\Box$'. So, the induction axiom in PTL

$$\vdash \;\; \Box(\varphi \Rightarrow \bigcirc\varphi) \;\Rightarrow\; (\varphi \Rightarrow \Box\varphi)$$

can now be viewed as the *interaction* axiom

$$\vdash \;\; [*](\varphi \Rightarrow [1]\varphi) \;\Rightarrow\; (\varphi \Rightarrow [*]\varphi)$$

in a modal logic with two modalities.

There are here two distinct accessibility relations, but [∗] represents the reflexive transitive closure of [1].

4: PTL Describes Sequences

"PTL can be thought of as a logic over sequences."

As mentioned earlier, the models for PTL are infinite sequences.

So, a sequence-based semantics can be given for PTL:

$s_i, s_{i+1}, \ldots \models \bigcirc p$	if, and only if, $s_{i+1}, \ldots \models p$
$s_i, s_{i+1}, \ldots \models \lozenge p$	if, and only if, there exists a $j \geq i$ such that $s_j, \ldots \models p$
$s_i, s_{i+1}, \ldots \models \square p$	if, and only if, for all $j \geq i$ then $s_j, \ldots \models p$

In this way, PTL can be seen as a logic for describing such infinite sequences.

5: PTL Describes ω-automata

"PTL can be seen as a syntactic characterisation of certain finite-state automata over infinite words."

Models of PTL can be seen as strings accepted by a class of finite automata — Büchi Automata.

Correspondingly, temporal formulae might be used to describe certain automata. Intuitively:

- formulae such as $\square(p \Rightarrow \bigcirc q)$ give constraints on possible transitions between automaton states;
- formulae such as $\square\lozenge r$ give constraints on *accepting states* within an automaton, i.e. states that must be visited infinitely often; and
- formulae such as $s \Rightarrow \square t$ describe global invariants within an automaton.

Normal Form for PTL

PTL formulae can become quite complex and difficult to understand.

It is often useful to replace one complex formula by several simpler formulae. We will use a particular *normal form* where PTL formulae are represented by

$$\square \bigwedge_{i=1}^{n} R_i$$

where each of the R_i, termed a *rule*, is an implication in the style

formula about current behaviour
$\Rightarrow$
formula about current and future behaviour

Separated Normal Form

Separated Normal Form (SNF) additionally restricts each R_i to be of one of the following forms

$\textbf{start} \Rightarrow \bigvee_{b=1}^{r} l_b$	(an *initial* rule)
$\bigwedge_{a=1}^{g} k_a \Rightarrow \bigcirc \bigvee_{b=1}^{r} l_b$	(a *step* rule)
$\bigwedge_{a=1}^{g} k_a \Rightarrow \lozenge l$	(a *sometime* rule)

Note, here, that each k_a, l_b, or l is simply a literal.

Examples

The example formulae below correspond to the three different types of SNF rules.

INITIAL:	**start**	$\Rightarrow$	$losing \lor hopeful$
STEP:	$(losing \land \neg hopeful)$	$\Rightarrow$	$\bigcirc losing$
SOMETIME:	$hopeful$	$\Rightarrow$	$\Diamond \neg losing$

Any PTL formula can be transformed into a set of SNF rules that have (essentially) equivalent behaviours at the expense of only a polynomial increase in both the size of the representation and the number of atomic propositions used within the representation.

Büchi Automata and PTL

Models for PTL are essentially infinite, discrete, linear sequences, with an identified start state.

Thus, each temporal formula corresponds to a set of models on which that formula is satisfied.

Now, look again at these models. See them as *strings*.

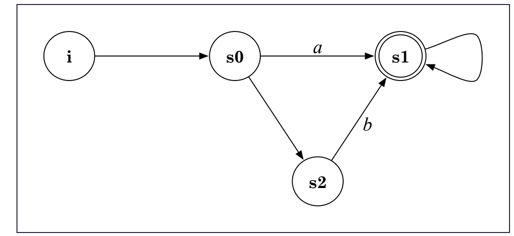
Viewing models as strings allows us to utilise the large amount of previous work on *finite automata*.

In particular, we can define a finite automaton that accepts *exactly* the strings we are interested in and so we can use finite automata to represent temporal models.

The automata needed are a specific form of automata over infinite strings, often termed 'ω-automata', or Büchi automata. (We will discuss them in detail later.)

Example Automaton (1)

Consider the formula '$\bigcirc(a \lor \bigcirc b)$'. We can generate a corresponding automaton:

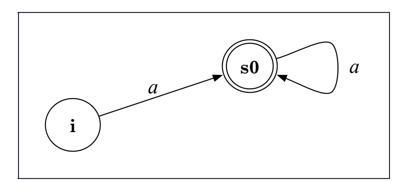


Notation: '*i*' represents an initial state and a double circle represents an *accepting* state (one of which must be visited infinitely often).

An edge with no label accepts *any* value.

Example Automaton (2)

The automaton translation of the formula '$\square a$' has more interesting infinite behaviour:



Thus, now, the "infinite tail" has to always accept an '*a*'.

What are Fixpoints?

In general, fixpoints are solutions to recursive equations, such as $x = f(x)$; in basic algebra, we know that $a = 0$ is a solution of $a = (a * 5)$.

There can often be several solutions to fixpoint equations; $a = 0$ and $a = 1$ are both solutions of $a = (a * a)$.

As we often wish to distinguish solutions, we select using

ν, representing the maximal (greatest) fixpoint, or
μ, representing the minimal (least) fixpoint.

For example, we write down $\nu a.\ (a * a)$ to denote the maximal fixpoint of $a = (a * a)$; similarly with 'μ'.

It is important to note that, in some cases *no* fixpoint exists.

Note: when only one exists, maximal and minimal versions coincide.

What are Temporal Fixpoints?

The μ (least fixpoint) and ν (greatest fixpoint) operators have been transferred to temporal logics.

A fixpoint solution to the formula '$\psi \Leftrightarrow (\varphi \wedge \bigcirc \psi)$' is some formula '$\psi$' that makes the statement true.

Taking $\psi \equiv \Box \varphi$ does make this formula true, since

$$\Box \varphi \Leftrightarrow (\varphi \wedge \bigcirc \Box \varphi)$$

is valid, while $\psi \equiv \bigcirc \varphi$ does not, since

$$\bigcirc \varphi \not\Leftrightarrow (\varphi \wedge \bigcirc \bigcirc \varphi)\,.$$

The fixpoint operators, together with 'next', form the basis of a powerful temporal logic called νTL.

Temporal Fixpoint Examples

$$\Box \varphi \quad \equiv \quad \nu\xi.\ (\varphi \wedge \bigcirc \xi)$$
$$\Diamond \varphi \quad \equiv \quad \mu\xi.\ (\varphi \vee \bigcirc \xi)$$

Here, $\Box \varphi$ is defined as the maximal fixpoint (ξ) of the formula $\xi \Leftrightarrow (\varphi \wedge \bigcirc \xi)$. Thus, the maximal fixpoint above effectively defines $\Box \varphi$ as the 'infinite' formula

$$\varphi \wedge \bigcirc \varphi \wedge \bigcirc \bigcirc \varphi \wedge \bigcirc \bigcirc \bigcirc \varphi \wedge \ldots$$

Note that the *minimal* fixpoint of $\xi \Leftrightarrow (\varphi \wedge \bigcirc \xi)$ is '**false**', since putting **false** in place of 'ξ' is legitimate and **false** is the minimal solution.

N.B: minimality/maximality are defined in relation to '$\Rightarrow$'; so '**false** is the minimal element while '**true**' is the maximal.

A Little Quantification

PTL can be extended with *quantification*.

Full *first-order* quantification is complex, so might allow quantification, but only over Boolean valued variables (specifically, propositions of the language).

Thus, using such a logic, called *Quantified Propositional Temporal Logic* (QPTL), it is possible to write formulae such as

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

where p is a propositional variable.