Temporal Logic

[Language, Intuition and Possibilities]

Michael Fisher

Department of Computer Science, University of Liverpool, UK

[MFisher@liverpool.ac.uk]

An Introduction to Practical Formal Methods Using Temporal Logic

Towards Temporal Logic

In classical logic, formulae are evaluated within a single fixed world. For example, a proposition such as “it is Monday” is either true or false.

Propositions are then combined using constructs such as ‘∧’, ‘¬’, etc.

In temporal logics, evaluation takes place within a set of worlds. Thus, “it is Monday” may be satisfied in some worlds, but not in others.

The accessibility relation between worlds then describes a particular model of time.

The set of classical operators is extended with various temporal operators which navigate by this relation.

Expression

Commonly, propositional, discrete, linear temporal logic extends the descriptive power of propositional logic in order to be able to

- describe sequences (hence: linear)
- of distinct (hence: discrete) worlds,
- with each world being similar to a classical (propositional) model.

So, we can equivalently describe the basis of our model of time in terms of a sequence of worlds, a sequence of states, or a sequence of propositional models.

Each state in the sequence is taken as modelling a different moment in time; hence the name temporal logic.

PTL “in a nutshell”

We use a simple temporal logic (PTL) where the accessibility relation characterises a discrete, linear order isomorphic to the Natural Numbers, \( \mathbb{N} \).

The set of classical operators is extended with various temporal operators which navigate by this relation.

φ is true in the next moment in time
φ is true in all future moments
φ is true in some future moment
φ is true up until some future moment when ψ is true
start only true at the beginning of time

The typical temporal operators used are

- \( \bigcirc \phi \)
- \( \bigboxdot \phi \)
- \( \bigtriangledown \phi \)
- \( \phi U \psi \)
PTL Examples

\[ \text{monday} \Rightarrow \bigcirc \text{tuesday} \]
\[ \text{start} \Rightarrow \lozenge \text{finish} \]
\[ \text{july} \Rightarrow \lozenge (\text{december} \land \text{winter}) \]
\[ \text{send}(\text{msg}, \text{rcvr}) \Rightarrow \lozenge \text{receive}(\text{msg}, \text{rcvr}) \]
\[ \Box ((\neg \text{passport} \lor \neg \text{ticket}) \Rightarrow \bigcirc \neg \text{board\_flight}) \]
\[ \text{sunrise} \Rightarrow \bigcirc (\text{night} \lor \text{dawn}) \]
\[ \text{born} \Rightarrow \lozenge \Box \text{old} \]
\[ \text{monday} \Rightarrow \text{sad} \Rightarrow \text{Usaturday} \]

Branching Models

Alternative is to use a branching model of time.

Branching Temporal Logic

CTL Syntax: each temporal operator is now prefixed by one of the following path operators

\[ \text{A} \rightarrow \text{‘on all future paths starting here’} \]
\[ \text{E} \rightarrow \text{‘on some future path starting here’} \]

Examples:
\[ \text{A} \Box \text{safe} \]
\[ \text{E} \bigcirc \text{active} \]
\[ \text{A} \lozenge \text{terminate} \]

Varieties:

<table>
<thead>
<tr>
<th>Logic</th>
<th>Typical Formula</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTL</td>
<td>( \text{A} \Box (\text{E} \bigcirc p \land \text{E} \bigcirc q) )</td>
<td>lacks expressiveness</td>
</tr>
<tr>
<td>CTL*</td>
<td>( \text{A} \Box \text{EA}_p )</td>
<td>complex</td>
</tr>
</tbody>
</table>

There are many other varieties also.
Varieties of Temporal Model

Many different models of time are used, all constraining the accessibility relation \((R)\) in some way, for example:

- **linear** — each moment has at most one future moment
- **branching** — a moment may have several future moments
- **discrete** — if \(R(w_1, w_2)\) then there is no \(w_3\) such that \(R(w_1, w_3)\) and \(R(w_3, w_2)\)
- **dense** — if \(R(w_1, w_2)\) then there is always a \(w_3\) such that \(R(w_1, w_3)\) and \(R(w_3, w_2)\)
- **finite past** — there exists a \(w_1\) such that we can find no \(w_2\) such that \(R(w_2, w_1)\)

Varieties of Temporal Operators

The vast range of different models lead to a large range of operators that can be seen in temporal languages:

- **standard discrete linear future** — \(\Box, \Diamond, \square\)
- **interval future** — \(U, W\)
- **past** — \(\blacklozenge, \blacklozenge, \blacklozenge, S, Z\)
- **fixpoints** — \(\mu, \nu\)
- **path quantifiers** — \(A, E\)
- **quantified propositional** — \(\exists p, \Box p\)
- **full first-order** — \(\forall x, \Box p(x)\)
- etc.

Working with PTL (1)

**Deduction**
Various proof methods have been developed, e.g.
- tableaux
- non-clausal resolution
- sequent systems
- translation methods (to FOL)
- translation methods (to MSOL)
- clausal resolution
- interactive theorem provers

Working with PTL (2)

**Model Checking**
Very popular technique for checking whether a temporal formula is satisfied in a particular (finite) structure
- the finite structure is often derived from program or hardware descriptions.

**Automata**
Temporal logics have a close relationship to \(\omega\) automata, i.e. finite state automata over infinite objects
- consequently automata-theoretic methods are often employed.
Temporal Logic Gets Everywhere

- specification and verification of (dynamic) programs
- specification and verification of distributed and concurrent programs
- representation of tense in natural language
- characterising temporal database queries
- verification of finite state models derived from ‘real’ systems (model checking)
- agent theory
- direct execution
- real-time analysis
- temporal data mining
- exploring the limits of decidability

Intuition: Safety Properties

Safety:
“something bad will not happen”

Typical examples:
- \( \neg (\text{reactor}_\text{temp} > 1000) \)
- \( \neg (\text{one}_\text{way} \land \text{other}_\text{way}) \)
- \( \neg ((x = 0) \land \text{other}(y = z/x)) \)
and so on....

Usually: \( \neg \...... \)

Intuition: Liveness Properties

Liveness:
“something good will happen”

Typical examples:
- \( \diamond \text{rich} \)
- \( \diamond \text{terminate} \)
- \( \diamond (x > 5) \)
and so on.....

Usually: \( \diamond \...... \)

Intuition: Fairness Properties

Fairness (strong):
“if we attempt/request infinitely often, then we will be successful/allocated infinitely often”

Often only really useful when scheduling processes, responding to messages, etc. Typical example:

\[ \forall p \in \text{processes. } \square \diamond \text{ready}(p) \Rightarrow \square \diamond \text{run}(p) \]

There are many forms of fairness, e.g:

- \( \square \diamond \text{attempt} \Rightarrow \square \diamond \text{succeed} \)
- \( \square \diamond \text{attempt} \Rightarrow \diamond \text{succeed} \)
- \( \diamond \text{attempt} \Rightarrow \neg \square \diamond \text{succeed} \)
- \( \diamond \text{attempt} \Rightarrow \diamond \text{succeed} \)
Example

We might describe the properties we require of a simple resource allocation system using temporal logic:

\[ \Diamond (\exists x. \text{allocate}(x)) \quad \text{[LIVENESS]} \]

\[ \land \quad \Box \neg (\text{allocate}(a) \land \text{allocate}(b)) \quad \text{[SAFETY]} \]

\[ \land \forall y. \Box \Diamond \text{request}(y) \Rightarrow \Box \Diamond \text{allocate}(y) \quad \text{[FAIRNESS]} \]

Aside: here, each process/agent requests a resource using 'request' with its name as argument.

System allocates to process/agent \( p \) by \( \text{allocate}(p) \).

Using Temporal Formulae (1)

We can usefully specify, using temporal formulae, systems with

- dynamic,
- concurrent,
- distributed,
- .....etc.....

aspects.

For example, from the resource allocator specification earlier, we might:

1. given a program intended to implement a resource allocator, we might prove that the program’s specification implies some temporal requirements.

Using Temporal Formulae (2)

But that isn’t all:

2. given a program that implements such a resource allocation system, we might model check a finite representation of the program to see if the implementation actually satisfies the required specification;

3. we might be able to synthesize a program directly from the above temporal logic requirement; or

4. we might be able to directly execute the above temporal specification to provide an implementation.

What Next?

We can now cover a range of topics concerning temporal logics, such as:

- syntax and semantics of temporal logic;
- simple temporal specifications;
- temporal reasoning, using a resolution procedure;
- formal verification, using model checking; and
- executable temporal specifications.

The first one is essential; the second is very useful; while the rest are independent of each other.