Temporal Logic

[Language, Intuition and Possibilities]

Michael Fisher

Department of Computer Science, University of Liverpool, UK

[MFisher@liverpool.ac.uk]
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.
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*. 
We use a simple temporal logic (PTL) where the accessibility relation characterises a discrete, linear order isomorphic to the Natural Numbers, \( \mathbb{N} \).

The typical temporal operators used are:

- \( \bigcirc \varphi \): \( \varphi \) is true in the *next* moment in time
- \( \square \varphi \): \( \varphi \) is true in *all* future moments
- \( \Diamond \varphi \): \( \varphi \) is true in *some* future moment
- \( \varphi U \psi \): \( \varphi \) is true *up until* some future moment when \( \psi \) is true
- *start*: only true at the *beginning of time*
PTL Examples

\[ \begin{align*}
  \text{monday} & \Rightarrow \Diamond \text{tuesday} \\
  \text{start} & \Rightarrow \text{\textdagger} \text{finish} \\
  \text{july} & \Rightarrow \text{\textdagger} \left( \text{december} \land \text{winter} \right) \\
  \text{send}(\text{msg, rcvr}) & \Rightarrow \text{\textdagger} \text{receive}(\text{msg, rcvr}) \\
  \Box \left( \neg \text{passport} \lor \neg \text{ticket} \right) & \Rightarrow \Box \neg \text{board\_flight} \\
  \text{sunset} & \Rightarrow \Box \left( \text{night} \lor \text{dawn} \right) \\
  \text{born} & \Rightarrow \Diamond \Box \text{old} \\
  \text{monday} & \Rightarrow \text{sad} \lor \text{sad\_saturday}
\end{align*} \]
Looking Back

- was true in the previous moment in time
- was true in all past moments
- was true in some past moment
- has been true since some past moment when was true
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*

- **A** – ‘on all future paths starting here’
- **E** – ‘on some future path starting here’

**Examples:**  \[ A □ \text{safe} \quad E ∃ \text{active} \quad A □ \text{terminate} \]

**Varieties:**

<table>
<thead>
<tr>
<th>Logic</th>
<th>Typical Formula</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTL</td>
<td>A □ (E □ p ∧ E □ q)</td>
<td>lacks expressiveness</td>
</tr>
<tr>
<td>CTL*</td>
<td>A □ □ EAp</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** — $\bigcirc$, $\lozenge$, $\Box$

**interval future** — $\bigcirc$, $\lozenge$

**past** — $\bullet$, $\lozenge$, $\blacksquare$, $\triangleleft$, $\triangleright$

**fixpoints** — $\mu$, $\nu$

**path quantifiers** — $\mathbf{A}$, $\mathbf{E}$

**quantified propositional** — $\exists p \ldots \bigcirc p$

**full first-order** — $\forall x \ldots \bigcirc p(x)$

etc....
**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
Safety:

“something bad will not happen”

Typical examples:

\[ \neg (\text{reactor}_\text{temp} > 1000) \]
\[ \neg (\text{one}_\text{way} \land \Diamond \text{other}_\text{way}) \]
\[ \neg ((x = 0) \land \Diamond \Diamond \Diamond \Diamond (y = z/x)) \]

and so on.....

Usually: \[ \Square \neg .... \]
Liveness:

“something good will happen”

Typical examples:

◊ rich
◊ terminate
◊ \((x > 5)\)

and so on.....

Usually: ◊ ....
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:

$$\forall p \in \text{processes}. \quad \Box \Diamond \text{ready}(p) \Rightarrow \Box \Diamond \text{run}(p)$$

There are many forms of fairness, e.g:

$$\Box \Diamond \text{attempt} \Rightarrow \Box \Diamond \text{succeed}$$

$$\Box \Diamond \text{attempt} \Rightarrow \Diamond \text{succeed}$$

$$\Box \text{attempt} \Rightarrow \Box \Diamond \text{succeed}$$

$$\Box \text{attempt} \Rightarrow \Diamond \text{succeed}$$
Example

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

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

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

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

\[ \text{[LIVENESS]} \]

\[ \text{[SAFETY]} \]

\[ \text{[FAIRNESS]} \]

\underline{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.
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.