Temporal Specification

[Communication]

Michael Fisher

Department of Computer Science, University of Liverpool, UK

[MFisher@liverpool.ac.uk]

An Introduction to Practical Formal Methods Using Temporal Logic
Recall that we earlier had

\[
Spec_1 = a \land \mathcal{O} b \land \mathcal{O} \mathcal{O} c \land \mathcal{O} \mathcal{O} \mathcal{O} \mathcal{O} (a \lor b)
\]

\[
Spec_2 = x \land \mathcal{O} \mathcal{O} \mathcal{O} y \land \mathcal{O} \mathcal{O} \mathcal{O} \mathcal{O} z
\]

and that the synchronous, concurrent combination of the two elements \(E_1\) and \(E_2\) was simply \(Spec_1 \land Spec_2\), i.e.

\[
(a \land x) \land \mathcal{O} b \land \mathcal{O} \mathcal{O} (c \land y) \land \mathcal{O} \mathcal{O} \mathcal{O} \mathcal{O} (z \land (a \lor b))
\]

Notice: \textit{no} interaction between \(\{a, b, c\}\) and \(\{x, y, z\}\).

We now want to allow one element to affect another.

Specifically, we will introduce \textit{communication}.
Here, communication is achieved between two elements by ensuring that certain variables are *shared* between the elements.

Thus, each element can read from, and write to, these variables.

Consequently, for simple communication, one element just writes to a shared variable, then the other element reads the value of that variable.
Typical constraints on shared variables are that

1. both *read* and *write* operations for these variables are atomic, i.e., each take one execution step and are uninterruptable.

2. at most one element is allowed to write to a variable at any one time.

For example: $\square \neg (\text{write}(E_1) \land \text{write}(E_2))$
Message Passing

Here, communication is achieved by one element sending a message to another element.

Depending on the properties of the communication medium, this message might arrive

- very quickly,
- at some (indeterminate) moment in the future, or
- possibly never (if errors can occur).

Typically, send and receive predicates are used to represent the sending and receiving of messages.

Usually, with this form of message-passing, the message arrives at its destination and is recorded immediately.
Channels (1)

This form of communication essentially combines aspects of both message passing and shared variables.

The variety we consider has two elements:

1. messages can only be passed down specified channels; and

2. message send (write to channel) and receive (read from channel) actions are typically synchronised.

\[ E_1 \xrightarrow{\text{CHANNEL}} E_2 \]
Channels (2)

Channels are uni-directional structures that exist between two processes.

In general only one item of information (message) at a time is allowed within a channel and communication can only take place through existing channels.

For example if a channel exists between $E_1$ and $E_2$ then $E_1$ can send a message to $E_2$ down this channel.

However, if there is no channel from $E_1$ to $E_3$, then communication to $E_3$ is impossible.

Contrast this with both message passing where a message can be addressed to any other element, and shared variables where just one value is stored.
Channels (3)

**Synchronised Communication** where one element can attempt to send a message via a channel, but may have to wait until the element at the other end of the channel is ready to receive it.

Similarly, a element can try to receive a message from a channel, but may have to wait until the other element sends a message.

**Typical Syntax:** when \( ch \) is the name of channel between \( E_1 \) and \( E_2 \), then the message send \( ch!out\_msg \) in \( E_1 \) is synchronised with the receive \( ch?in\_msg \) in \( E_2 \).
Our Assumptions

Throughout most of our description of temporal specification, we will be dealing with systems that exhibit

- true concurrency,
- synchronous concurrency, and
- general message-passing.

Note, however, that we will be mentioning some of the problems of asynchrony and shared variables later.

Also, when we examine model checking, we will be using a model that involves channel-based communication.
Imagine a specification, $Spec_1$, for a element $E_1$, and another specification, $Spec_2$, for element $E_2$.

Now we wish communication to occur between elements.

One way to achieve this is to identify those aspects in each system (hence, in each specification) that correspond to information coming into a element, or going out of it.

For example, a specification, $Spec_i$ contains:

- a set of propositions used within it, $Props(Spec_i)$,
- a set of propositions corresponding to *incoming* information, $In(Spec_i)$, and
- a set of propositions corresponding to *outgoing* information, $Out(Spec_i)$.
Linking Specifications (2)

The idea here is that

1. the propositions in \( \text{In}(Spec_i) \) become true when an item of information is received, while

2. making the propositions in \( \text{Out}(Spec) \) true has the side-effect of passing information out of the element.

In general, when we link elements, we also link their specifications.

Thus, the specification of the whole system would be

\[
Spec_1 \land Spec_2 \land \text{Comms}(E_1, E_2)
\]
A typical example (using message-passing) is:

\[ Spec_1 \land Spec_2 \land Comms(E_1, E_2) \]

Here \( Comms \) links messages sent by each element to those received by the other.

For example, if \( a \in Out(Spec_1) \) and \( x \in In(Spec_2) \) then

\[ a \Rightarrow \Diamond x \]

specifies that, once \( a \) is made true within \( E_1 \), then \( x \) will eventually be made true within \( E_2 \).
Example 1

Assume:

\[
Spec_1 = a \land \bigcirc b \land \bigcirc \bigcirc c \land \bigcirc \bigcirc \bigcirc (a \lor b)
\]

\[
Spec_2 = x \land \bigcirc \bigcirc \bigcirc y \land \bigcirc \bigcirc \bigcirc \bigcirc z
\]

Also, assume that \(a \in \text{Out}(Spec_1)\) and \(x \in \text{In}(Spec_2)\).

Now, if \(\text{Comms}(E_1, E_2) = \square (a \Rightarrow \bigcirc x)\)
this would ensure that \(Spec_1 \land Spec_2 \land \text{Comms}(E_1, E_2) =\)

\[
(a \land x) \land \bigcirc (b \land x) \land \bigcirc \bigcirc (c \land y) \land \bigcirc \bigcirc \bigcirc \bigcirc (z \land ((a \land \bigcirc x) \lor b))
\]

Recall that, since the combination is synchronous, both elements share the same notion of ‘next’.
Example 2

Again:  
\[ Spec_1 = a \land \Box b \land \Box \Box c \land \Box \Box \Box \Box (a \lor b) \]
\[ Spec_2 = x \land \Box \Box y \land \Box \Box \Box \Box z \]

Now, with  
\[ Comms(E_1, E_2) = \Box (a \Rightarrow \Box x) \land \Box (y \Rightarrow \Box \neg a) \]
then \[ Spec_1 \land Spec_2 \land Comms(E_1, E_2) \]
becomes  
\[ (a \land x) \land \Box (b \land x) \land \Box \Box (c \land y) \land \Box \Box \Box \Box (z \land \neg a \land b) \]

Notice, particularly, how the execution within \( E_2 \) (i.e. \( y \) becoming true) now affects the choices within \( E_1 \) (i.e. \( a \) being false and so \( b \) being chosen from \( a \lor b \)).
Example 3

Consider two specifications, for elements $E_3$ and $E_4$, respectively:

$Spec_3$: \[
\begin{array}{c}
\begin{array}{c}
\text{start} \Rightarrow a \\
a \Rightarrow \Box b \\
c \Rightarrow \Box d
\end{array}
\end{array}
\]

$Spec_4$: \[
\begin{array}{c}
\begin{array}{c}
x \Rightarrow \Box y
\end{array}
\end{array}
\]
We might graphically represent $E_3$ on its own as

Here, we know that if we have ‘$a$’ then ‘$b$’ will occur next, and if we have ‘$c$’ then ‘$d$’ will occur next. In addition, we know that ‘$a$’ is true at the start, represented by $\boxed{a}$.

Similarly with element $E_4$:

Notice that, on their own, these elements do very little.
Adding  \( Comms(E_3, E_4) = \Box(b \Rightarrow \Diamond x) \)

We now add a simple \( Comms \) formula:

\[
Comms(E_3, E_4) = \Box(b \Rightarrow \Diamond x)
\]

This ensures that

- once \( b \) is made true in \( Spec_3 \) (at \( i = 1 \)), \( x \) will be made true at the next step after that (\( i = 2 \)), and so
- \( Spec_4 \) ensures that \( y \) will be made true at \( i = 3 \).

Now, if we instead use

\[
Comms(E_3, E_4) = \Box(b \Rightarrow \Diamond x)
\]

the effect on \( x \) will occur at \textit{some} indeterminate moment in the future once \( b \) has occurred.
Adding $Comms(E_3, E_4) = \square (b \Rightarrow \Diamond x)$

Graphically, we might represent this as

Here, time is increasing as we move down.

So: $a$ occurs before $b$ and the message from $b$ to $x$ takes some time (but we are unsure exactly how long, and so the gradient of this line is indeterminate).
Now, with: \( \text{Comms}(E_3, E_4) = \Box(b \Rightarrow \Diamond x) \land \Box(y \Rightarrow \Diamond c) \)

we can infer that \( d \) will eventually become true, i.e. \( \Diamond d \).
Exercise

\[ Spec_5: \begin{array}{c}
\begin{bmatrix}
\text{start} & \Rightarrow & a \\
\land & a & \Rightarrow & \bigcirc b \\
\land & c & \Rightarrow & \bigcirc d
\end{bmatrix}
\end{array} \]

\[ Spec_6: \begin{array}{c}
\begin{bmatrix}
x & \Rightarrow & \bigcirc y
\end{bmatrix}
\end{array} \]

What happens when we have \( Comms(E_5, E_6) \) as

\[ Comms = \square (b \Rightarrow \bigcirc x) \land \square (y \Rightarrow \bigcirc c) \]

And what can we now infer about ‘\( d \)’ in element \( E_5 \)?