Temporal Specification

[Case Studies]

Michael Fisher

Department of Computer Science, University of Liverpool, UK

[MFisher@liverpool.ac.uk]
Think of Mother as a simple resource controller.

- She has a bag of sweets.
- Each child wants a sweet, and can ask for one.
- Mother will only give out one sweet at a time.

**Mother:**

\[
\begin{align*}
\text{asked}(\text{jill}) & \implies \Diamond \text{give}(\text{jill}) \\
\land \text{asked}(\text{jack}) & \implies \Diamond \text{give}(\text{jack}) \\
\land \text{true} & \implies (\neg \text{give}(\text{jill}) \vee \neg \text{give}(\text{jack}))
\end{align*}
\]

Intuitive meanings of these formulae:

1. if a child asked for a sweet then it will eventually receive one;
2. only one child can receive a sweet at any moment.
Jill is a child who requests a sweet as often as possible:

\[
\text{JILL: } \quad \begin{array}{c}
\square \\
\land \\
\text{start} & \Rightarrow & \text{req(jill)} \\
\land & \text{req(jill)} & \Rightarrow & \Diamond \text{req(jill)}
\end{array}
\]

Now, to implement communication, we define the formula

\[
\text{Comms(mother, jill)} = \quad \square (\text{req(jill)} \Rightarrow \Diamond \text{asked(jill)}) \\
& \quad \square (\text{give(jill)} \Rightarrow \Diamond \text{got(jill)})
\]
‘Jack’

Jack will have similar behaviour to Jill, but will make a request every other moment in time.

\[
\text{JACK: } \begin{bmatrix}
\text{start} & \Rightarrow & \text{req(jack)} \\
\wedge & \text{req(jack)} & \Rightarrow & \Diamond \text{wait} \\
\wedge & \text{wait} & \Rightarrow & \Diamond \text{req(jack)}
\end{bmatrix}
\]

Again, to implement communication, we define the formula

\[
\text{Comms(mother, jack)} = \begin{bmatrix}
\Box (\text{req(jack)} \Rightarrow \Diamond \text{asked(jack)}) \\
\Box (\text{give(jack)} \Rightarrow \Diamond \text{got(jack)})
\end{bmatrix}
\]
Communication Structure

Parent

asked_jill
give_jill
asked_jack
give_jack

Jill

req_jill
got_jill

Jack

req_jack
got_jack
Properties of the System

Let us consider the specifications of Mother, Jack and Jill, together with the communications formulae.

1. Jill will request infinitely often — $\square req(jill)$.

2. Jack asked infinitely often — $\square \diamond asked(jack)$.

3. Mother has a *liveness* constraint saying that, if a child has asked, some allocation will be given.

4. Mother has a *safety* constraint saying that allocations to *jack* and *jill* must never happen at the same time.

Can prove many properties of system, e.g. $\square \diamond got(jack)$.
Imagine we now change the behaviour of Jack so that this element only asks for a sweet when it sees Jill being given one.

\[
\text{JACK: } \square \left[ \text{jealous} \Rightarrow \Diamond \text{req(jack)} \right]
\]

Now, to implement this, we must have a form of broadcast communication.
Extension: Broadcast

So, \(\text{give}(jill)\) in Mother’s specification is linked to more than one other proposition, in more than one other element, for example

\[
\text{Comms}(\text{mother, } [jill, \text{ jack}]) = \begin{cases} 
\Box (\text{req}(jill) \Rightarrow \Diamond \text{ asked}(jill)) \\
\Box (\text{give}(jill) \Rightarrow \Diamond \text{ got}(jill)) \\
\Box (\text{give}(jill) \Rightarrow \Diamond \text{ jealous})
\end{cases}
\]
New Communication Structure

Parent

- asked_jill
- give_jill
- asked_jack
- give_jack

Jill

- req_jill
- got_jill

Jack

- jealous
- req_jack
- got_jack
Many Varieties of Communication

We have seen different communication models:

\[ send \Rightarrow receive \]
\[ send \Rightarrow \bigcirc receive \]

\[ \ldots \ldots \ldots \]

\[ send \Rightarrow \Diamond receive \]

as well as multicast/broadcast:

\[ send \Rightarrow \Diamond receive_1 \]
\[ send \Rightarrow \Diamond receive_2 \]
\[ send \Rightarrow \Diamond receive_3 \]

But, notice the difference between the above and

\[ send \Rightarrow \Diamond (receive_1 \land receive_2 \land receive_3) \]