COM4506/6506: Testing and Verification in Safety Critical Systems

Dr Ramsay Taylor

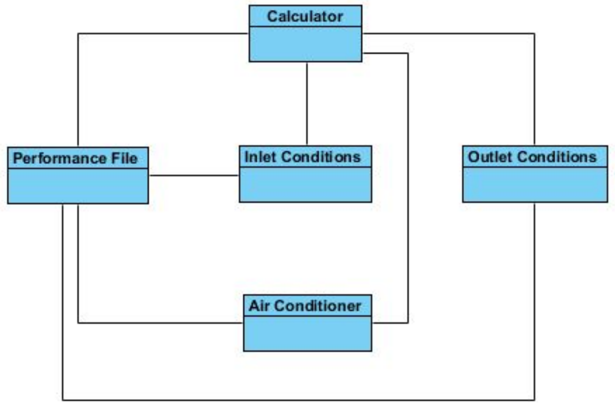
Contents

- Formality
- Process based Specifications
- State based Specifications
- Both?

Formality

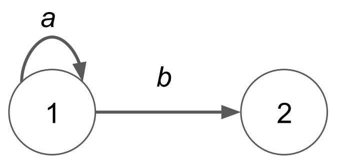
Various *Structured Documents* can be helpful as tools for human processes (e.g. Hazard Analysis).

Their *flexibility* can be a strength and a weakness...



Not just algebra

Finite State Machines have *Formal Semantics*



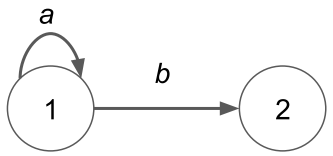
==

	1	2
1	a	b
2	0	0

Not just algebra [ok, this is mostly algebra, isn’t it?]

Formal Semantics allow Formal Reasoning

$$\begin{bmatrix} a & b \\ 0 & 0 \end{bmatrix}^2 == \begin{bmatrix} a.a + a.0 & a.b + b.0 \\ 0.a + 0.0 & 0.b + 0.0 \end{bmatrix} == \begin{bmatrix} a.a + a & a.b + b \\ 0 & 0 \end{bmatrix}$$

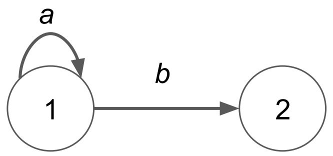


Not just algebra [ok, this is mostly algebra, isn’t it?]

Formal Reasoning allows *Formal Verification* (of **properties, not systems**!)

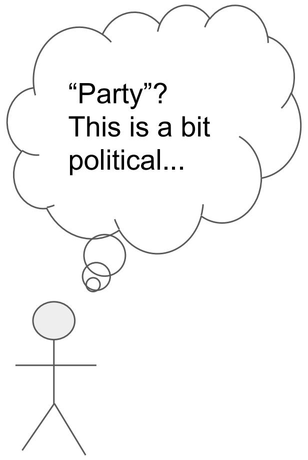
“Can we ever do *b* then *a*?

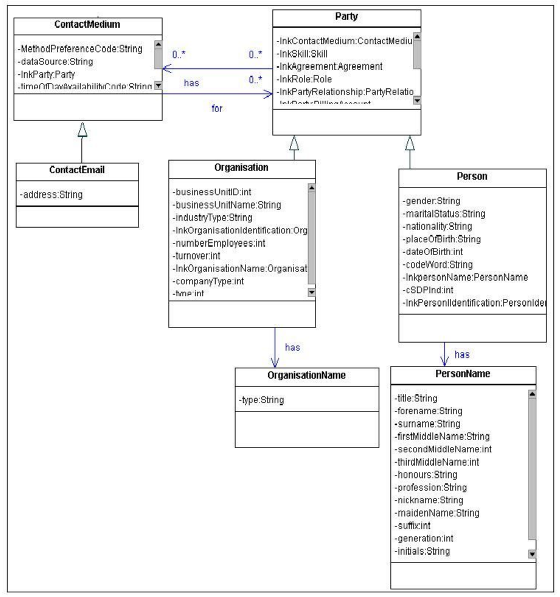
$$\begin{bmatrix} a.a + a & a.b + b \\ 0 & 0 \end{bmatrix}$$

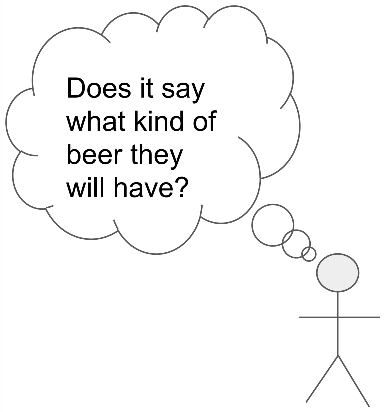


(“Model Checking” is a more involved and exhaustive version of this sort of thing.)

Abstraction

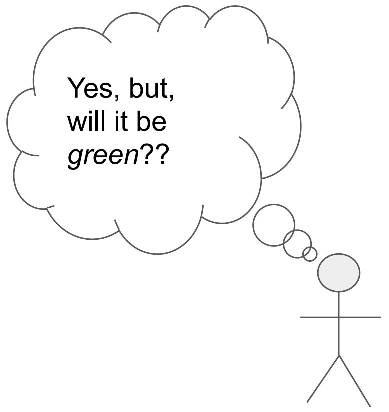






Abstraction

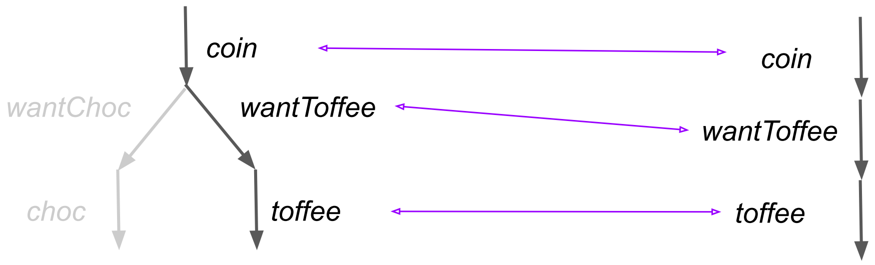
$$\begin{array}{|l} \hline mov_{LIT\#SRC,REG\#TGT} \\ \Delta\ System \\ \hline registers' = registers \oplus TGT \mapsto SRC \\ memory' = memory \\ \hline \end{array}$$



Process based Formal Specifications

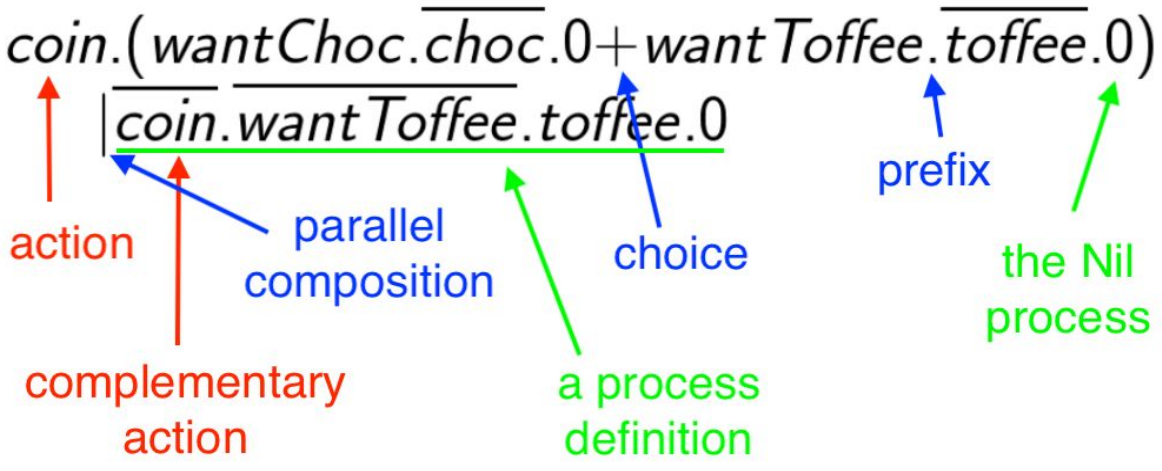
CSP and CCS are *Process Algebras*, and they deal with *Communicating Systems*

$$coin.(wantChoc.\overline{choc}.0+wantToffee.\overline{toffee}.0)$$
$$|\overline{coin}.\overline{wantToffee}.toffee.0$$



Process based Formal Specifications

Every symbol is *formally defined*



$$coin.(wantChoc.\overline{choc}.0+wantToffee.\overline{toffee}.0)$$
$$|\overline{coin}.\overline{wantToffee}.toffee.0$$

Process based Formal Specifications

Process Algebras *abstract* the *actions*.

This is *good* if we want to work out whether things will happen *in some sequence* or if things will *deadlock*

This is *bad* if we care about the details of the operations.

We can only verify properties over things that exist in the model.

State Specifications

Z, B, and bits of some others are *State Specifications*.

They define bits of system *State*

And then *operations* over this state.

$$BIT == \{0,1\}$$
$$INT32 == \{0..2^{32}\}$$
$$REGNAMES == \{eax, ebx, ecx, edx, esp, ebp\}$$

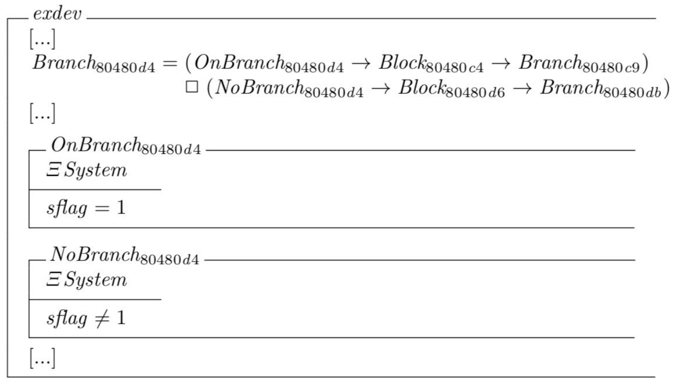
System

$$\begin{array}{|l}
memory : INT32 \nrightarrow INT32 \\
registers : REGNAMES \rightarrow INT32 \\
ioports : INT32 \nrightarrow INT32 \\
zf, cf, sf : BIT
\end{array}$$

$mov_{LIT\#SRC,REG\#TGT}$

$$\begin{array}{|l}
\Delta\ System \\
\hline
registers' = registers \oplus TGT \mapsto SRC \\
memory' = memory
\end{array}$$

Can I do both Process and State specifications



exdev

[...]

$$\begin{aligned}Branch_{80480d4} = {} & (OnBranch_{80480d4} \rightarrow Block_{80480c4} \rightarrow Branch_{80480c9}) \\ & \Box\ (NoBranch_{80480d4} \rightarrow Block_{80480d6} \rightarrow Branch_{80480db})\end{aligned}$$

[...]

OnBranch$_{80480d4}$

$\Xi System$

$sflag = 1$

NoBranch$_{80480d4}$

$\Xi System$

$sflag \neq 1$

[...]

The *predicates* in Z, for example, limit the sequence of possible operations, so you can (ab)use that.

CSP-OZ, Circus, and Event-B are languages that explicitly combine process and state languages.

There are plenty more languages out there!

Summary

- Formal Methods reduce ambiguity
- They require abstraction, so don’t remove all ambiguity!
- Formal Models allow Formal Reasoning, and so Formal Verification
- There are different formal languages for different problems