Alcino Cunha

SPECIFICATION AND MODELING

COMPUTATION TREE LOGIC

Universidade do Minho & INESC TEC

2019/20

TRASH

TRASH

Design a trash component such that:

- It is always the case that any existing file can end up in the trash

TRASH BEHAVIOUR

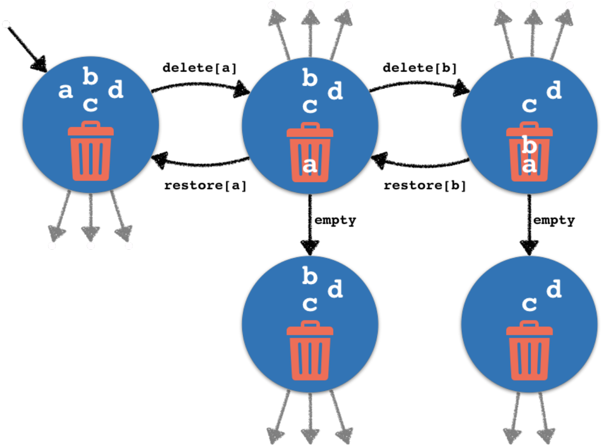
```
var sig File {}
var sig Trash in File {}

pred delete[f : File] { ... }
pred restore[f : File] { ... }
pred empty { ... }
pred do_nothing { ... }

fact {
  no Trash
  always (
    (some f: File | delete[f] or restore[f]) or empty or do_nothing
  )
}
```

HOW TO EXPRESS POSSIBILITY IN LTL?

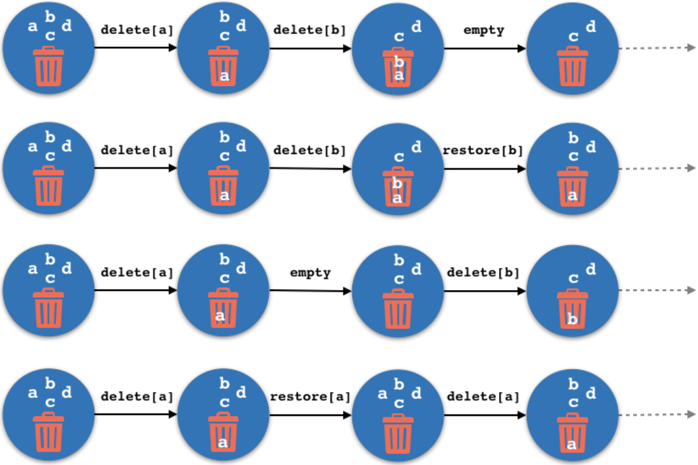
```
assert Inevitable {
    always (all f : File | eventually (f in Trash))
}

assert Possible {
    always (all f : File | ????? (f in Trash))
}
```

MODELS OF TIME

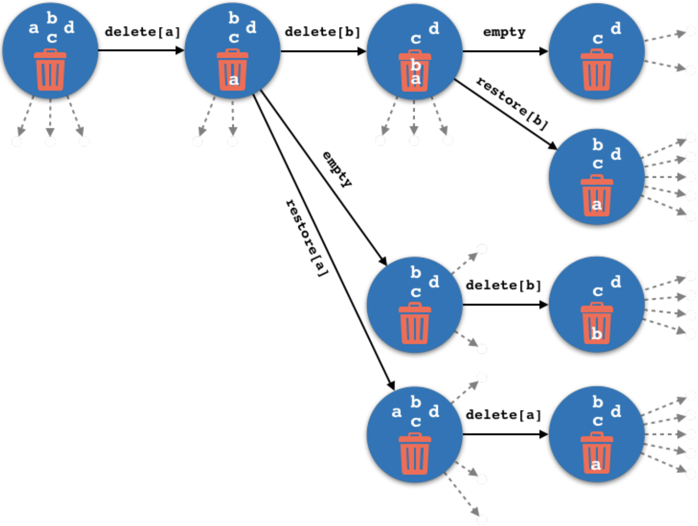
TRASH TRANSITION SYSTEM



LINEAR MODEL OF TIME



BRANCHING MODEL OF TIME



LINEAR TEMPORAL LOGIC VS COMPUTATION TREE LOGIC

- The transition system is abstracted by a set of infinite *traces*
  - This is known as a *linear model of time*
  - Forgets the choices available at each state
  - It is the semantic model for the *Linear Temporal Logic* (LTL)

VS

- The transition system is abstracted by a set of infinite *computation trees*
  - This is known as a *branching model of time*
  - Keeps the choices available at each state
  - It is the semantic model for the *Computation Tree Logic* (CTL)

COMPUTATION TREE LOGIC

TEMPORAL OPERATORS

Operator		Meaning
$\mathrm{G}\,\phi$	$\square\,\phi$	ϕ is always true from now on
$\mathrm{F}\,\phi$	$\diamond\,\phi$	ϕ will eventually be true
$\mathrm{X}\,\phi$	$\bigcirc\,\phi$	ϕ will be true in the next state
$\phi\ \mathrm{U}\ \psi$		ψ will eventually be true and ϕ is true until then
$\phi\ \mathrm{R}\ \psi$		ψ can only be false after ϕ is true

PATH QUANTIFIERS

Operator	Meaning
A ϕ	ϕ is valid in all paths
E ϕ	ϕ is valid in some path

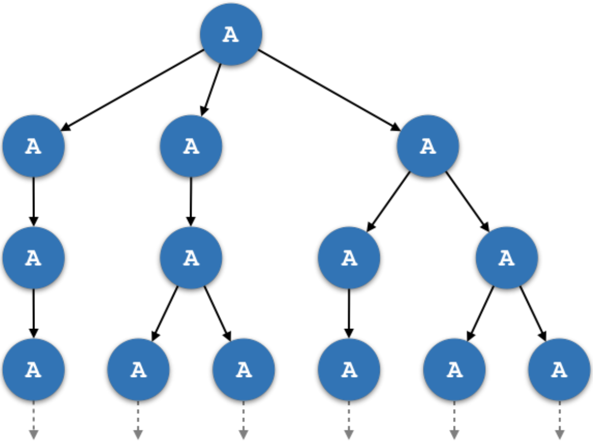
- A path quantifier must always be followed by a temporal operator
- In practice we have ten temporal connectives

SYNTAX

$$
\begin{array}{rcl}
\phi & ::= & \mathsf{AG}\,\phi \\
& | & \mathsf{EG}\,\phi \\
& | & \mathsf{AF}\,\phi \\
& | & \mathsf{EF}\,\phi \\
& | & \mathsf{AX}\,\phi \\
& | & \mathsf{EX}\,\phi \\
& | & \phi\ \mathsf{AU}\ \psi \\
& | & \phi\ \mathsf{EU}\ \psi \\
& | & \phi\ \mathsf{AR}\ \psi \\
& | & \phi\ \mathsf{ER}\ \psi \\
& | & \ldots
\end{array}
$$

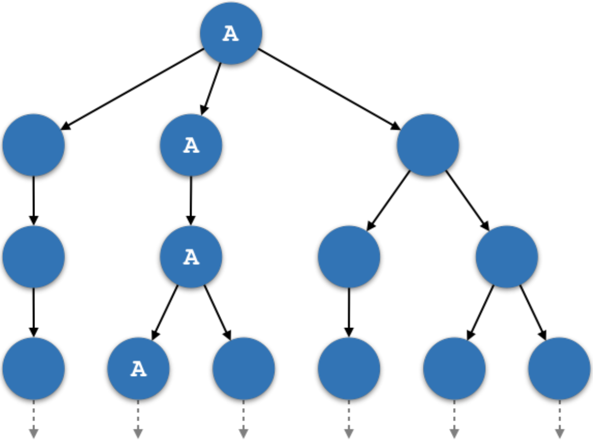
SEMANTICS BY EXAMPLE

AG (**some** A)



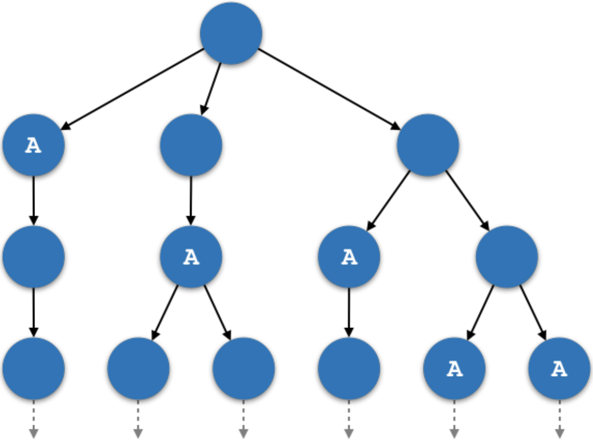
SEMANTICS BY EXAMPLE

`EG (some A)`



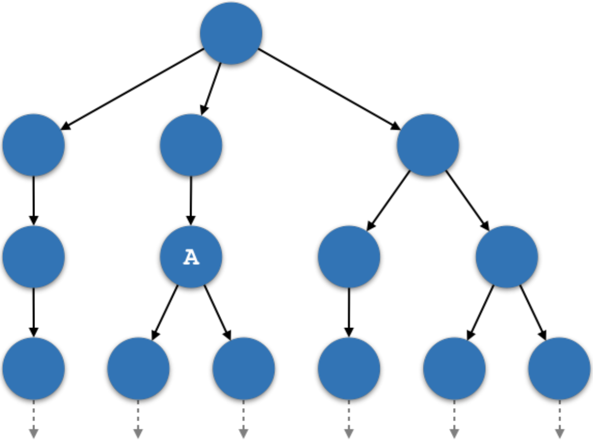
SEMANTICS BY EXAMPLE

AF (**some** A)



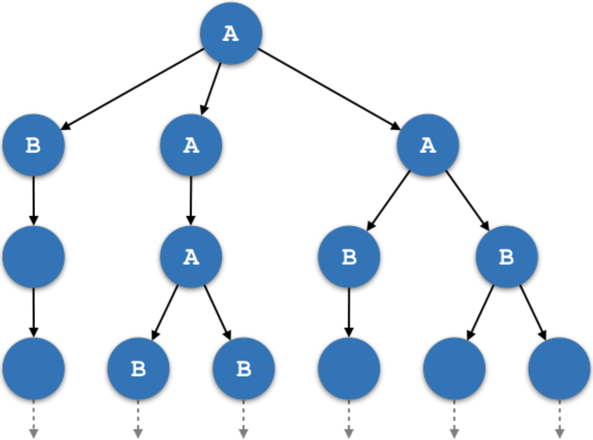
SEMANTICS BY EXAMPLE

`EF (some A)`



SEMANTICS BY EXAMPLE

(**some** A) **AU** (**some** B)



IF ELECTRUM SUPPORTED CTL...

```
assert Possible {
    AG (all f : File | EF (f in Trash))
}
```

EXPRESSIVENESS OF CTL VS LTL

- The expressiveness of LTL and CTL is incomparable
- Some CTL properties cannot be expressed in LTL

$$\mathrm{AG}\,\mathrm{EF}\,\phi$$

- Some LTL properties cannot be expressed in CTL, namely those related to fairness

$$\mathrm{F}\,\mathrm{G}\,\phi \neq \mathrm{AF}\,\mathrm{AG}\,\phi$$