Alcino Cunha

SPECIFICATION AND MODELING

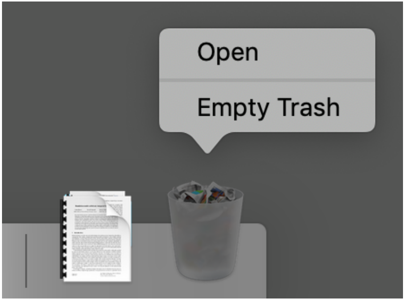
COMPUTATION TREE LOGIC

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

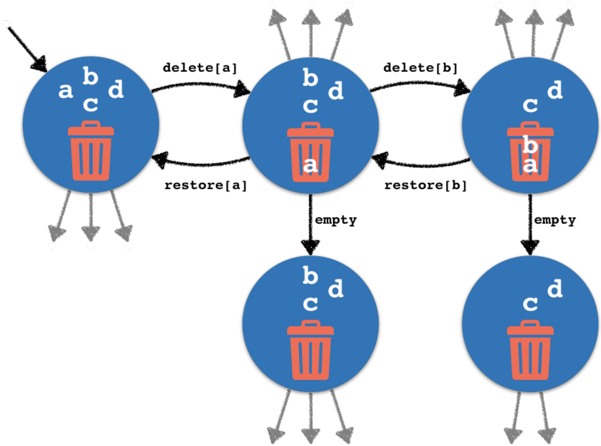
```
var sig File {}
var sig Trash in File {}

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?

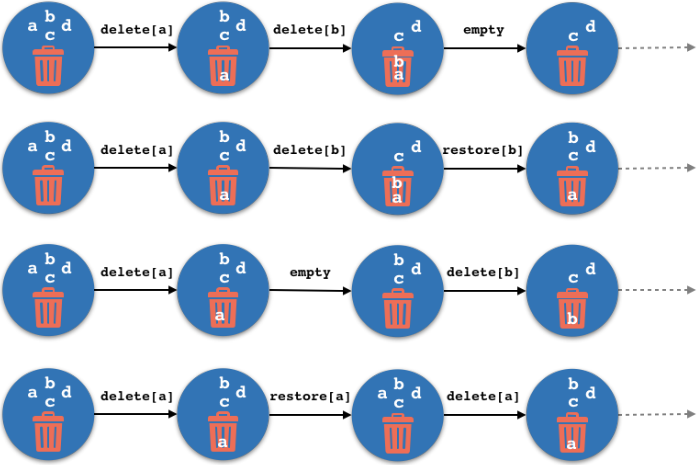
```
assert Inevitable {
    always (all f : File | eventually (f in Trash))
}

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

MODELS OF TIME

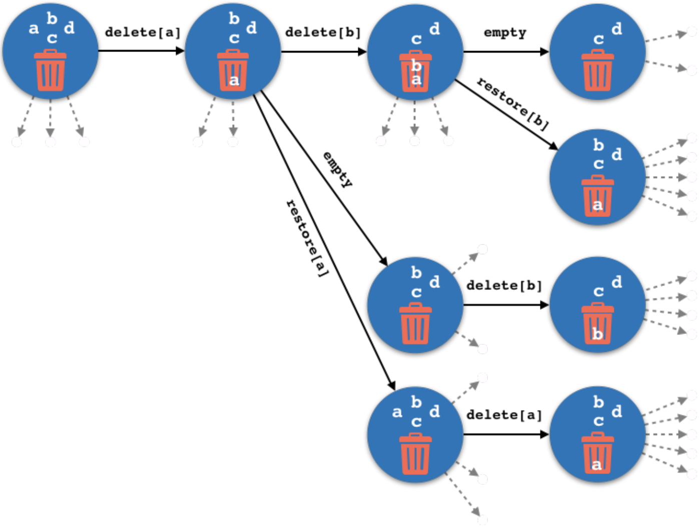
TRASH TRANSITION SYSTEM



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
$\mathrm{G}\phi \quad \Box\phi$	ϕ is always true from now on
$\mathrm{F}\phi \quad \Diamond\phi$	ϕ will eventually be true
$\mathrm{X}\phi \quad \bigcirc\phi$	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

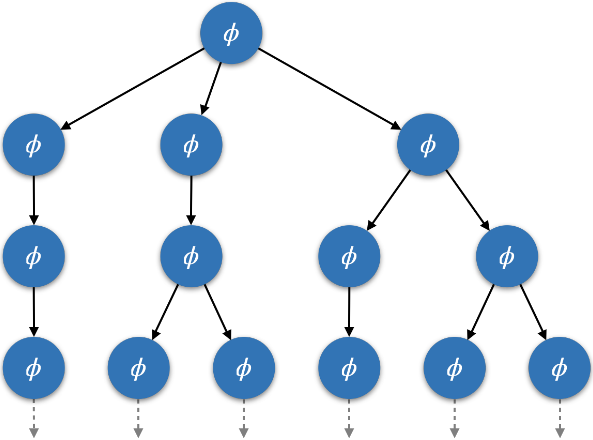
- A path quantifier must always be followed by a temporal operator
- In practice we have ten temporal connectives

SYNTAX

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

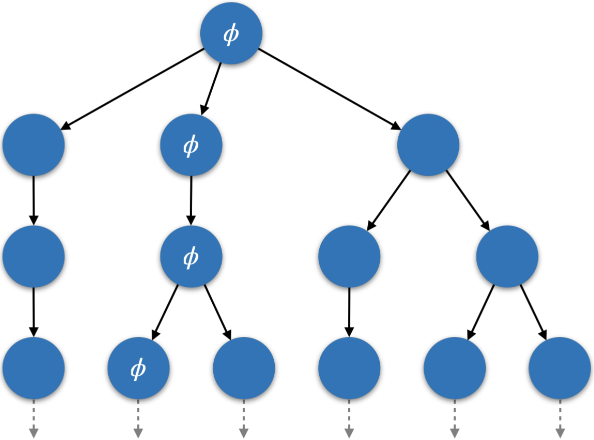
SEMANTICS BY EXAMPLE

$\mathrm{AG}\,\phi$



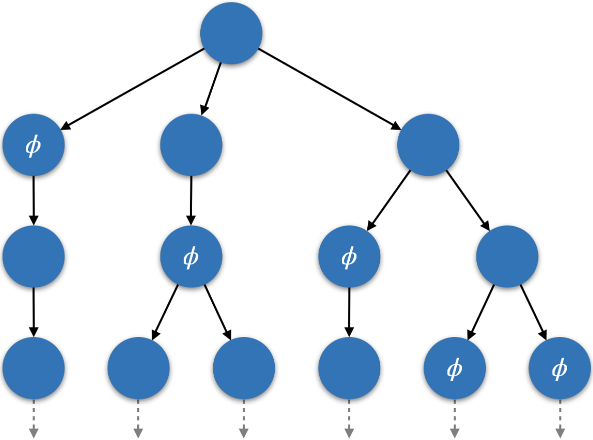
SEMANTICS BY EXAMPLE

$\mathrm{EG}\, \phi$



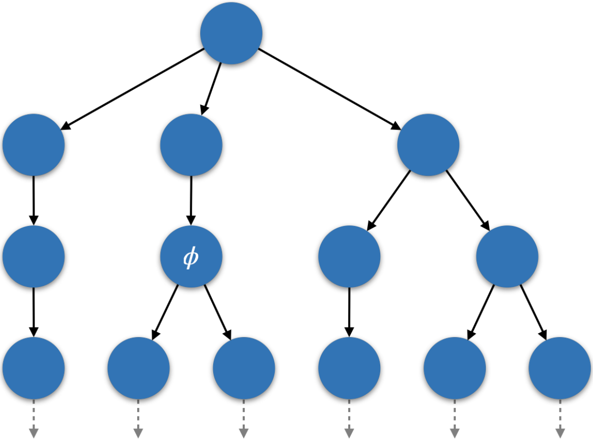
SEMANTICS BY EXAMPLE

$\text{AF}\,\phi$



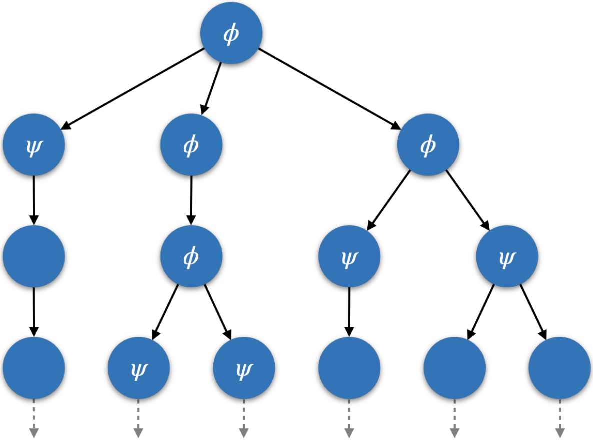
SEMANTICS BY EXAMPLE

$\mathrm{EF}\,\phi$



SEMANTICS BY EXAMPLE

ϕ AU ψ



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

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