Model Checking Computation Tree Logic (CTL)

Didier Buchs
University of Geneva
from CTL Model Checking, Paul Jackson , University of Edinburgh .

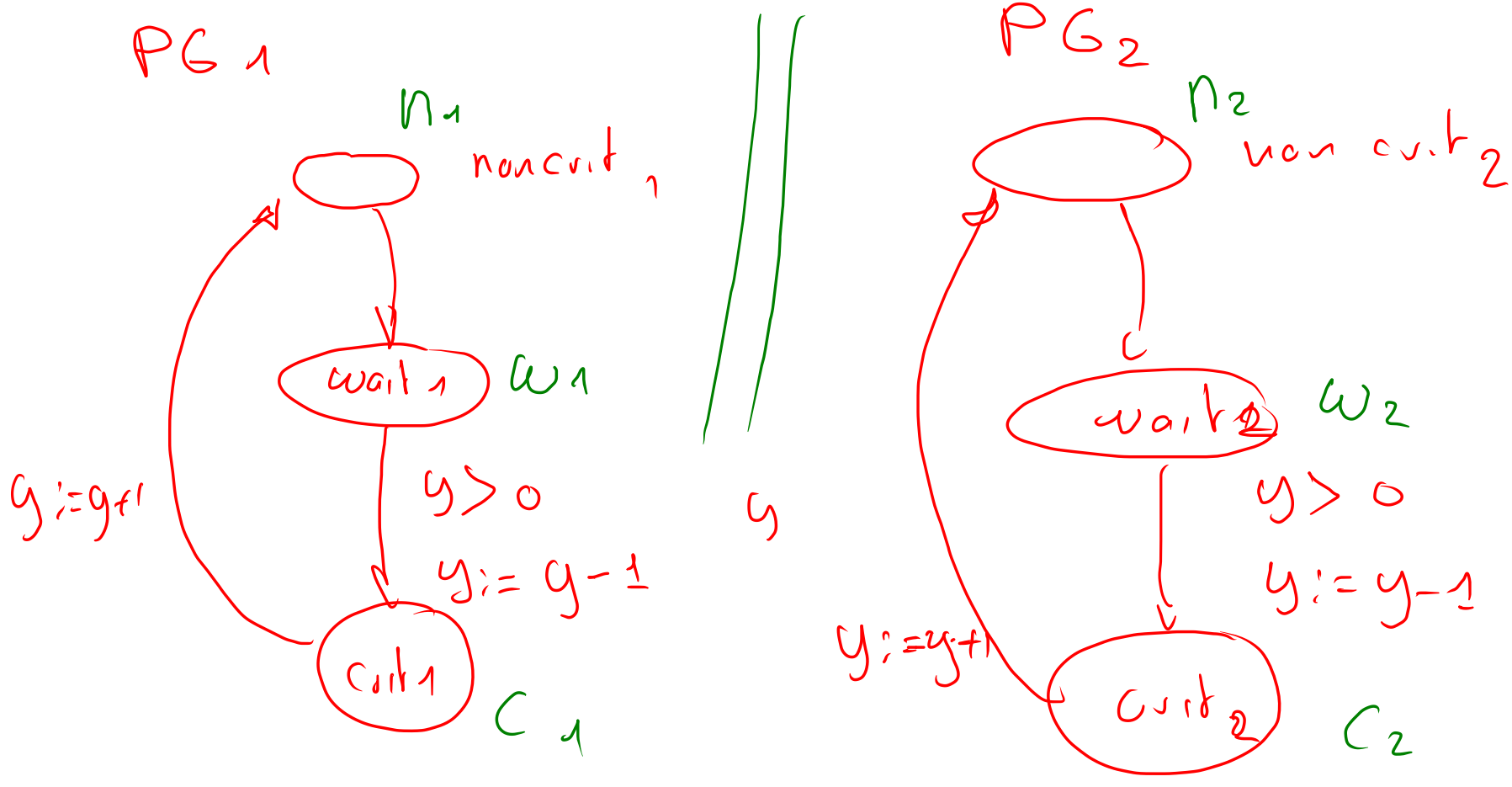
Programme for the upcoming lectures

- Introducing CTL
- Basic Algorithms for CTL
- Basic Decision Diagrams

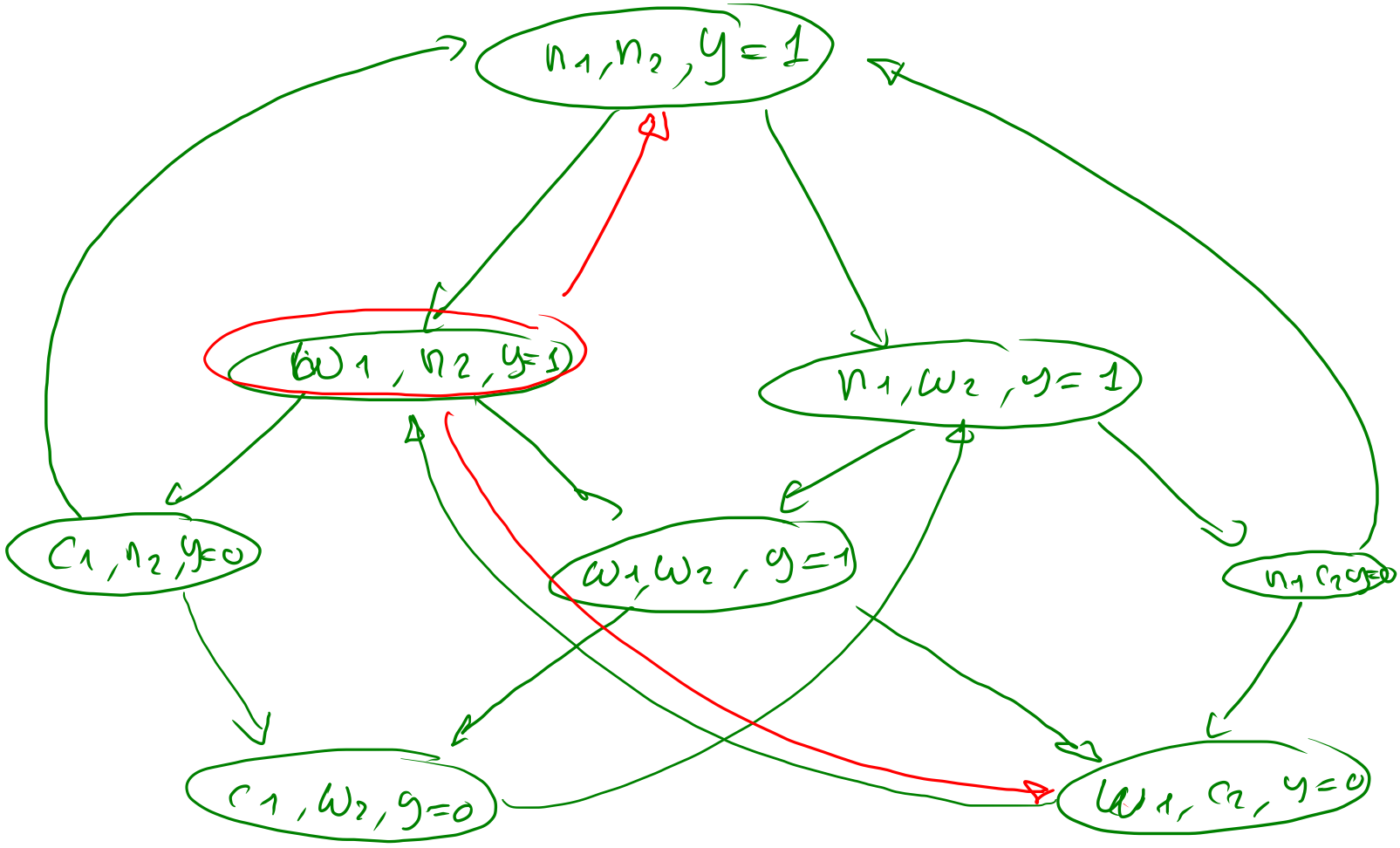
The denotation of a CTL formula

- Before, we defined a satisfaction relation $M, s \models \phi$.
- CTL model checking algorithms usually fix $M = \langle S, \rightarrow, AP, \nu \rangle$ and ϕ and compute $[\![\phi]\!]_M = \{s \in S | M, s \models \phi\}$
- $[\![\phi]\!]_M$ read as "the denotation of ϕ in model M".
- The relationship between satisfaction and denotation is $M, s \models \phi$ iff $s \in [\![\phi]\!]_M$
- Often M is implicit and we write $[\![\phi]\!]$ rather than $[\![\phi]\!]_M$

Semaphore par l'exclusion mutuelle



(Sémantique d'interleavings)



CTL satisfaction for multiple initial states

ours = Model checking CTL + BDD

- In NuSMV, "CTLSPEC ϕ" asks whether ϕ is satisfied in the given model which has a set of initial states
- The NuSMV definition of CTL satisfaction with a set of initial states S_0 is : $M, S_0 \models \phi$ iff $\forall s \in S_0, M, s \models \phi$
- We then have $M, S_0 \models \phi$ iff $S_0 \subseteq [\![\phi]\!]_M$

Denotational semantics for CTL

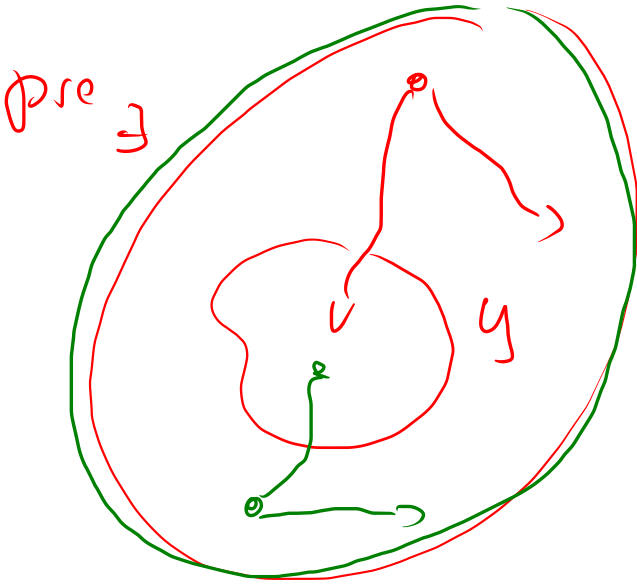
Instead of defining $[\![\phi]\!]$ in terms of $\models \phi$,we can define it directly–recursively on the structure of ϕ , ψ and $p \in AP$

$$
\begin{array}{rcl}
[\![t]\!] & = & S \\
[\![f]\!] & = & \varnothing \\
[\![p]\!] & = & \nu(p) \\
[\![\neg\phi]\!] & = & S - [\![\phi]\!] \\
[\![\phi \wedge \psi]\!] & = & [\![\phi]\!] \cap [\![\psi]\!] \\
[\![\phi \vee \psi]\!] & = & [\![\phi]\!] \cup [\![\psi]\!]
\end{array}
$$

Since $[\![\phi]\!]$ is always a finite set, these are computable.

Denotational semantics for CTL : temporal operators

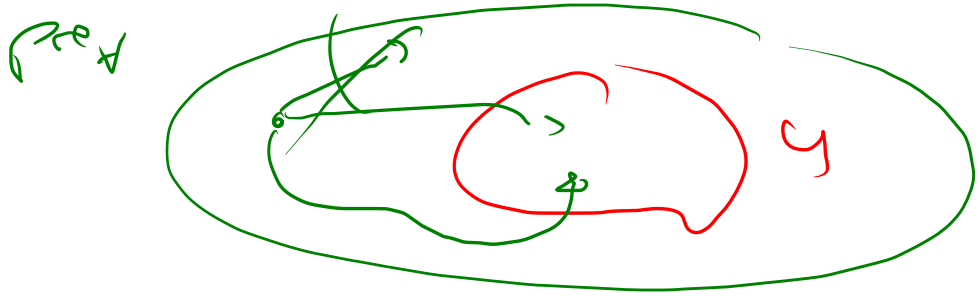
$$[\![\mathrm{EX}\,\phi]\!] = pre_{\exists}([\![\phi]\!])$$
$$[\![\mathrm{AX}\,\phi]\!] = pre_{\forall}([\![\phi]\!])$$



where

$pre_{\exists}(Y) = \{s \in S | \exists s' \in S, s \to s' \wedge s' \in Y\}$

$pre_{\forall}(Y) = \{s \in S | \forall s' \in S, s \to s' \Rightarrow s' \in Y\}$



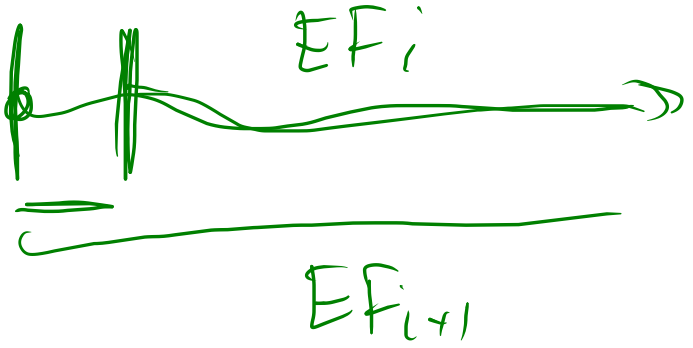
These are computable if we have the whole transition system, but what about the rest E.g.

$[\![\mathrm{EF}\,\phi]\!] = \{s \in S | \exists path\ \pi \text{ s.t. } \pi(0) = s, \exists i\ \pi(i) \models \phi\}$ does not suggest how to compute $[\![\mathrm{EF}\,\phi]\!]$

Considering fixing path length as approximation of $[\![\mathsf{EF}\,\phi]\!]$

Define

$$\begin{aligned} \mathsf{EF}_0\,\phi &= f \\ \mathsf{EF}_{i+1}\,\phi &= \phi \vee \mathsf{EX}\ \mathsf{EF}_i\,\phi \end{aligned}$$



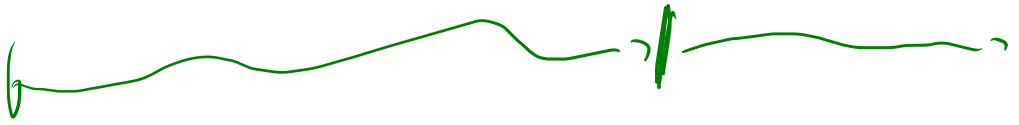
Then have

$$\begin{aligned} \mathsf{EF}_1\,\phi &= \phi \\ \mathsf{EF}_2\,\phi &= \phi \vee \mathsf{EX}\,\phi \\ \mathsf{EF}_3\,\phi &= \phi \vee \mathsf{EX}(\phi \vee \mathsf{EX}\,\phi) \end{aligned}$$

$s \in [\![\mathsf{EF}_i\,\phi]\!]$ if there exists a finite path of length $i-1$ from s and ϕ holds at some point on that path.

Finite Kripke structures

For a given model M, let $n = |S|$,
If there is a path of length $k > n$ on which ϕ holds somewhere, there also will be path of length n.
(Proof : take the k-length path and repeatedly cut out segments between repeated states)
Therefore, $\forall k > n, [\![\mathsf{EF}_k\, \phi]\!] = [\![\mathsf{EF}_n\, \phi]\!]$



Computing $[\![\mathsf{EF}\,\phi]\!]$

By a similar argument

$$[\![\mathsf{EF}\,\phi]\!] = [\![\mathsf{EF}_n\,\phi]\!]$$

$$\begin{array}{rcl} [\![\mathsf{EF}_0\,\phi]\!] & = & \varnothing \\ [\![\mathsf{EF}_{i+1}\,\phi]\!] & = & [\![\phi]\!] \cup \mathit{pre}_\exists([\![\mathsf{EF}_i\,\phi]\!]) \end{array}$$

We have here an effective way of computing $[\![\mathsf{EF}\,\phi]\!]$

Approximation of $[\![\mathsf{EG}\,\phi]\!]$

Define

$$\begin{aligned} \mathsf{EG}_0\,\phi &= t \\ \mathsf{EG}_{i+1}\,\phi &= \phi \wedge \mathsf{EX}\ \mathsf{EG}_i\,\phi \end{aligned}$$

Then have

$$\begin{aligned} \mathsf{EG}_1\,\phi &= \phi \\ \mathsf{EG}_2\,\phi &= \phi \wedge \mathsf{EX}\,\phi \\ \mathsf{EG}_3\,\phi &= \phi \wedge \mathsf{EX}(\phi \wedge \mathsf{EX}\,\phi) \end{aligned}$$

$s \in [\![\mathsf{EG}_i\,\phi]\!]$ if there exists a finite path of length $i-1$ from s and ϕ holds at every point on that path.

As with $[\![\mathsf{EF}\,\phi]\!]$, we have $\forall k > n$, $[\![\mathsf{EG}_k\,\phi]\!] = [\![\mathsf{EG}_n\,\phi]\!] = [\![\mathsf{EG}\,\phi]\!]$ and so we can compute $[\![\mathsf{EG}\,\phi]\!]$. Similarly we can compute the denotation of the other temporal connectives .

Fixed-Point theory

- In general, what's happening here is that we are computing fixed-points.
- A set $X \subseteq S$ is a fixed point of a function $F \subseteq P(S) \to P(S)$ iff $X = F(X)$.
- We have that

$$\begin{aligned} [\![\mathsf{EF}_n\,\phi]\!] &= [\![\mathsf{EF}_{n+1}\,\phi]\!] \\ &= [\![\phi \vee \mathsf{EX}\,\mathsf{EF}_n\,\phi]\!] \\ &= [\![\phi]\!] \cup \mathit{pre}_\exists([\![\mathsf{EF}_n\,\phi]\!]) \end{aligned}$$

  so $[\![\mathsf{EF}_n\,\phi]\!]$ is a fixed point of $F = \lambda Y.[\![\phi]\!] \cup \mathit{pre}_\exists(Y)$
- Also $[\![\mathsf{EF}\,\phi]\!]$ is a fixed-point of F ,since $[\![\mathsf{EF}_n\,\phi]\!] = [\![\mathsf{EF}\,\phi]\!]$.
- More specifically, $[\![\mathsf{EF}_n\,\phi]\!]$ and $[\![\mathsf{EF}\,\phi]\!]$ are the least fixed point of F.

Fix point theorem

A function $F \in P(S) \rightarrow P(S)$ is **monotone** iff
$X \subseteq Y$ implies $F(X) \subseteq F(Y)$ for all subsets X and Y of S.

Let $F^i(X) = F(F^{i-1}(X))$ for $i > 0$ and $F^0(X) = X$.

Given a collection of sets $C \subseteq P(S)$, and a set $X \in C$
X is the **least element** of C iff $\forall Y \in C$, $X \subseteq Y$,and
X is the **greatest element** of C iff $\forall Y \in C$, $Y \subseteq X$.

Fix point theorem

Theorem (Knaster-Tarski Theorem (special case))

Let S be a set with n elements and $F \in P(S) \to P(S)$ be a monotone function. Then

- *$F^n(\varnothing)$ is the least fixed point of F ,and*
- *$F^n(S)$ is the greatest fixed point of F*

This theorem justifies $F^n(\varnothing)$ and $F^n(S)$ being fixed points of F without the need, to appeal to further details about F.

Denotational semantics for CTL temporal operator

When $F \in P(S) \to P(S)$ a monotone function, let us write $\mu Y.F(Y)$ for the least fixed point of F ,and $\nu Y.F(Y)$ for the greatest fixed point of F .
With this notation, we can make the definitions

$$\begin{aligned}
[\![\mathsf{EF}\,\phi]\!] &= \mu Y.[\![\phi]\!] \cup \mathit{pre}_{\exists}(Y) \\
[\![\mathsf{EG}\,\phi]\!] &= \nu Y.[\![\phi]\!] \cap \mathit{pre}_{\exists}(Y) \\
[\![\mathsf{AF}\,\phi]\!] &= \mu Y.[\![\phi]\!] \cup \mathit{pre}_{\forall}(Y) \\
[\![\mathsf{AG}\,\phi]\!] &= \nu Y.[\![\phi]\!] \cap \mathit{pre}_{\forall}(Y) \\
[\![\mathsf{E}[\phi U \psi]]\!] &= \mu Y.[\![\psi]\!] \cup ([\![\phi]\!] \cap \mathit{pre}_{\exists}(Y)) \\
[\![\mathsf{A}[\phi U \psi]]\!] &= \mu Y.[\![\psi]\!] \cup ([\![\phi]\!] \cap \mathit{pre}_{\forall}(Y))
\end{aligned}$$

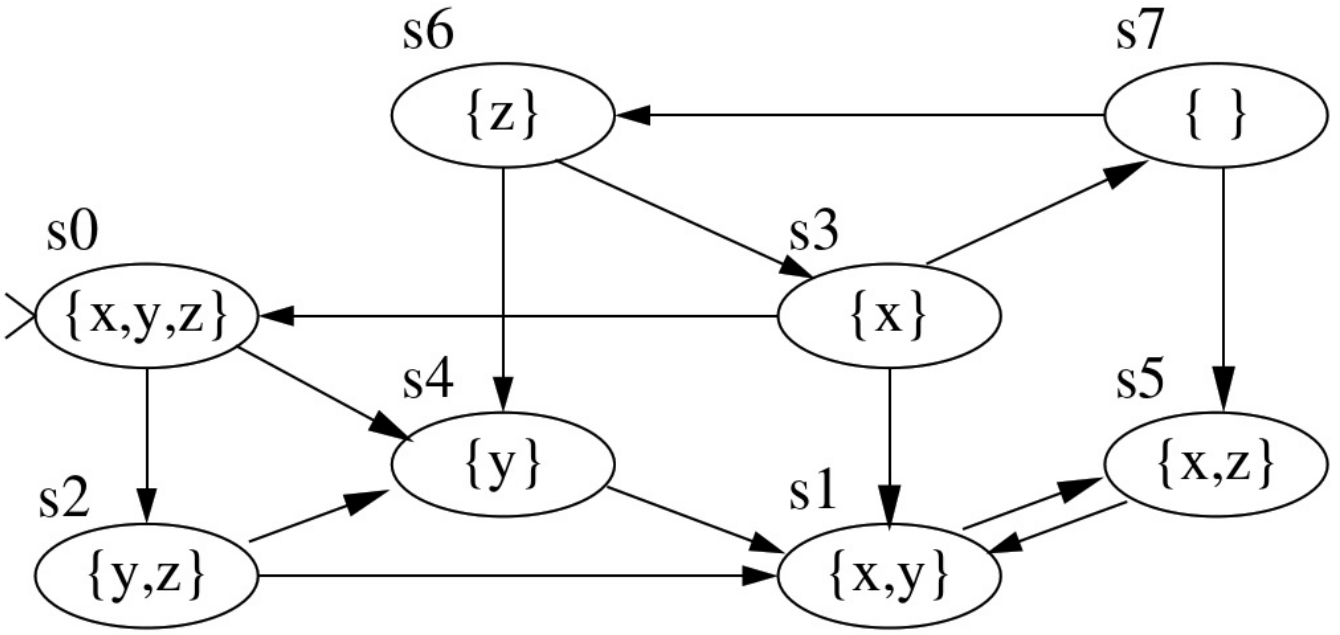
In every case the $F(Y)$ is monotone, so the KT theorem assures us the fixed point exists and can be computed.

Fixed-point identities

The fixed-point characterisations of the CTL temporal operators justify the CTL identities

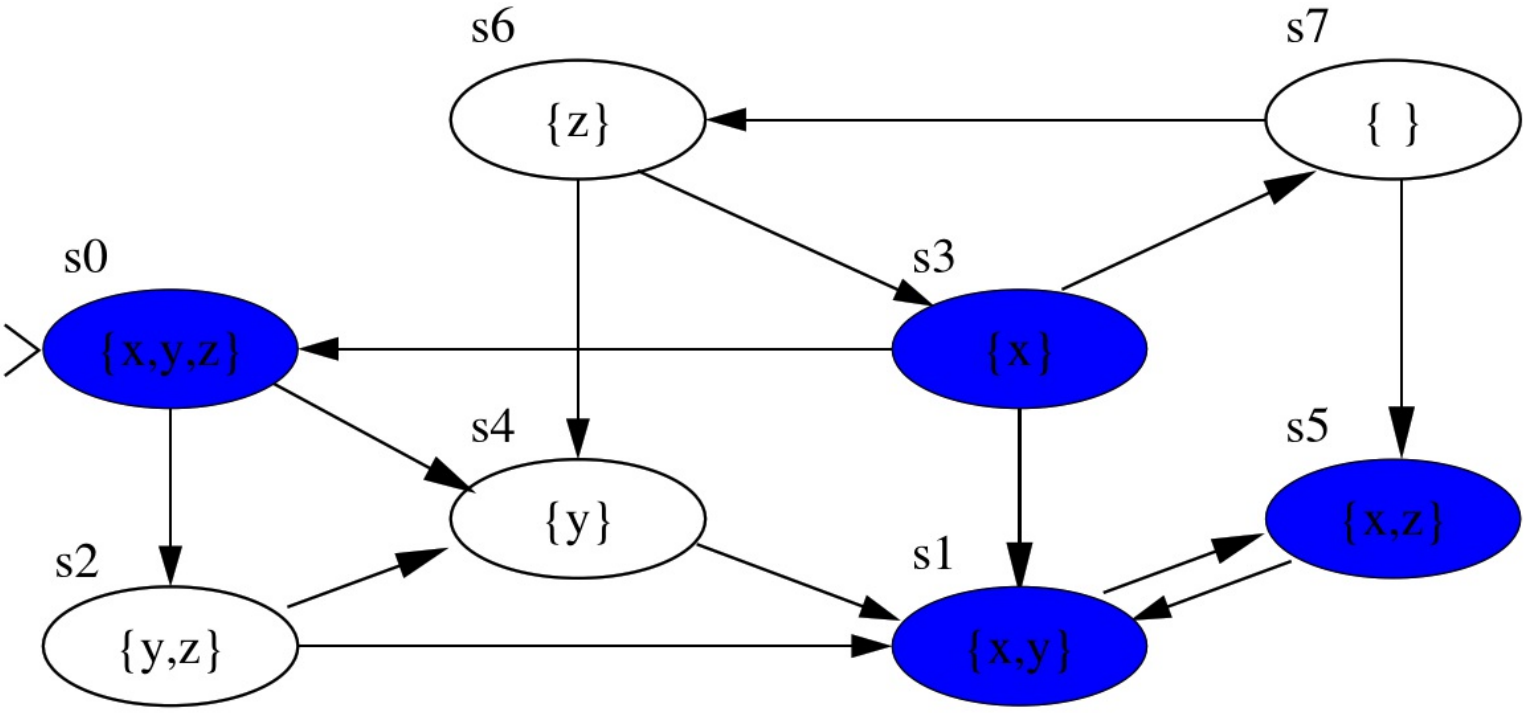
$$
\begin{aligned}
\mathsf{EF}\,\phi &= \phi \vee \mathsf{EX}(\mathsf{EF}\,\phi) \\
\mathsf{EG}\,\phi &= \phi \wedge \mathsf{EX}(\mathsf{EG}\,\phi) \\
\mathsf{AF}\,\phi &= \phi \vee \mathsf{AX}(\mathsf{AF}\,\phi) \\
\mathsf{AG}\,\phi &= \phi \wedge \mathsf{AX}(\mathsf{AG}\,\phi) \\
\mathsf{E}[\phi\,\mathsf{U}\,\psi] &= \psi \vee (\phi \wedge \mathsf{EX}(\mathsf{E}[\phi\,\mathsf{U}\,\psi])) \\
\mathsf{A}[\phi\,\mathsf{U}\,\psi] &= \psi \vee (\phi \wedge \mathsf{AX}(\mathsf{A}[\phi\,\mathsf{U}\,\psi]))
\end{aligned}
$$

Solving nested formulas : Is $s_0 \in [\![\mathsf{AF}\,\mathsf{AG}\,x]\!]$?



- To compute the semantics of formulas with nested operators, we first compute it recursively as imbrication of fix-point computations.
- In this example, we compute
  $[\![\mathsf{AF}\,\mathsf{AG}\,x]\!] = \mu Y.[\![\mathsf{AG}\,x]\!] \cup pre_\forall(Y)$
  $[\![\mathsf{AG}\,x]\!] = \nu Y.[\![x]\!] \cap pre_\forall(Y)$
  $[\![x]\!] =$? in that order.

Fix-point method (1) : Compute$[\![x]\!]$



We have here the result of $\nu(x) = \{s_0, s_1, s_3, s_5\} = [\![x]\!]$

Fix-point method (2) : Compute$[\![\mathsf{AG}\, x]\!]$

$[\![\mathsf{AG}\, x]\!] = \nu Y.[\![x]\!] \cap pre_\forall(Y)$

(1) $\{s_0, s_1, s_3, s_5\} \cap pre_\forall(\{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\})$

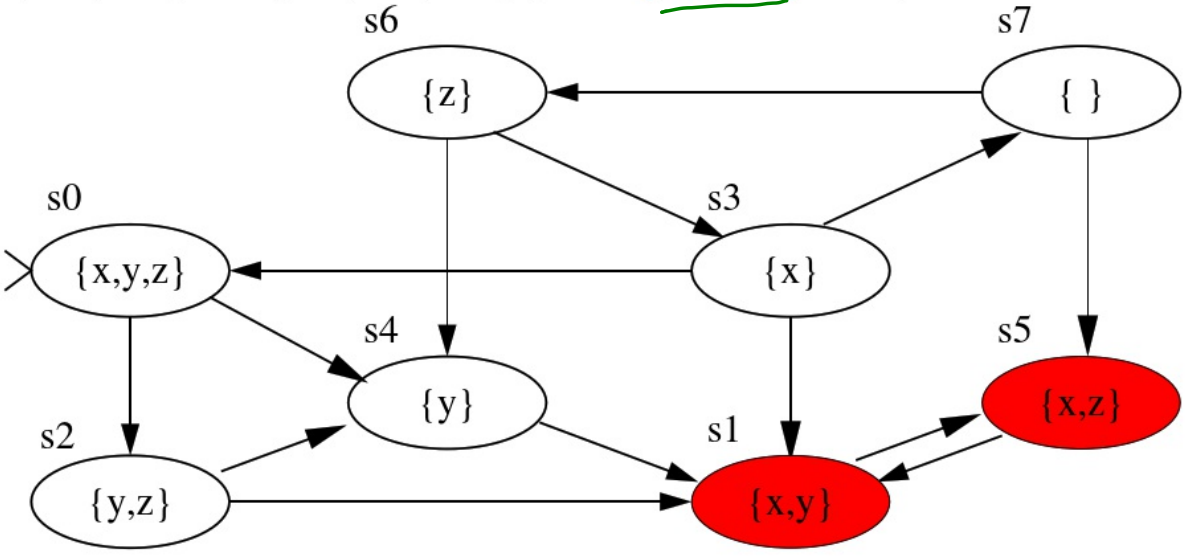
(1') $\{s_0, s_1, s_3, s_5\} \cap \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\} = \{s_0, s_1, s_3, s_5\}$

(2) $\{s_0, s_1, s_3, s_5\} \cap pre_\forall(\{s_0, s_1, s_3, s_5\})$

(2') $\{s_0, s_1, s_3, s_5\} \cap \{s_4, s_5, s_1\}) = \{s_1, s_5\}$

(3) $\{s_0, s_1, s_3, s_5\} \cap pre_\forall(\{s_1, s_5\})$

(3') $\{s_0, s_1, s_3, s_5\} \cap \{s_4, s_5, s_1\}) = \{s_1, s_5\}$ fixpoint reached



Fix-point method (3) : Compute$[\![\mathrm{AF\,AG}\,x]\!]$

$[\![\mathrm{AF\,AG}\,x]\!] = \mu Y.[\![\mathrm{AG}\,x]\!] \cup pre_\forall(Y)$

(1) $\{s_1, s_5\} \cup pre_\forall(\varnothing) = \{s_1, s_5\}$

(2) $\{s_1, s_5\} \cup pre_\forall(\{s_1, s_5\})$

(2') $\{s_1, s_5\} \cup \{s_1, s_4, s_5\} = \{s_1, s_4, s_5\}$

(3) $\{s_1, s_5\} \cup pre_\forall(\{s_1, s_4, s_5\})$

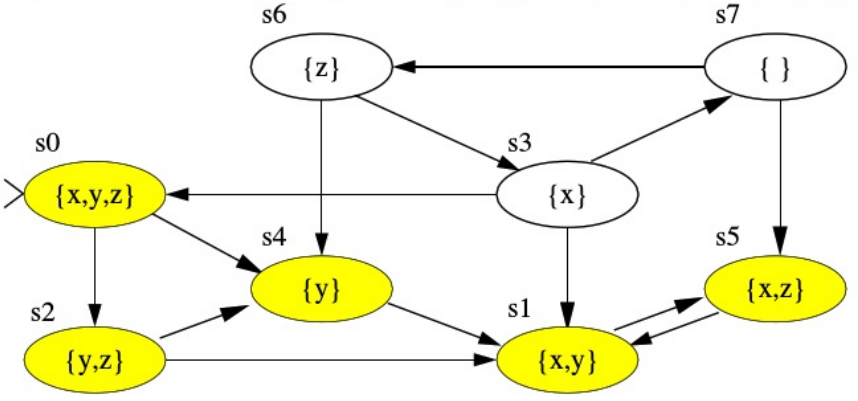
(3') $\{s_1, s_5\} \cup \{s_1, s_2, s_4, s_5\} = \{s_1, s_2, s_4, s_5\}$

(4) $\{s_1, s_5\} \cup pre_\forall(\{s_1, s_2, s_4, s_5\})$

(4') $\{s_1, s_5\} \cup \{s_0, s_1, s_2, s_4, s_5\} = \{s_0, s_1, s_2, s_4, s_5\}$

(5) $\{s_1, s_5\} \cup pre_\forall(\{s_0, s_1, s_2, s_4, s_5\})$

(5') $\{s_1, s_5\} \cup \{s_0, s_1, s_2, s_4, s_5\} = \{s_0, s_1, s_2, s_4, s_5\}$ fixpoint



Conclusion

- *Conclusion* : The model checking principles are stated as fix-point computations.
- Complexity issues are difficult to master. Next chapter will show some techniques for this.
- *Remark* : The *pre* function is fundamental for computing fix-points but it is not always necessary to memorize it with the transition system. It can be computed as the inverse of the firing functions (for instance in P/T Petri nets).

a = b

E (a)

b

QA

=

a = b

f(a) = f(b)

(succ(0) > 0) = true

size(empty) = 0

size(cons(n, empty)) = succ(0) > 0