CONCEPTION ET VÉRIFICATION DE SYSTÈMES CRITIQUES

LA SPÉCIFICATION DES PROPRIÉTÉS AVEC LA LOGIQUE CTL

2A Cursus Ingénieurs - ST5 : Modélisation fonctionnelle et régulation

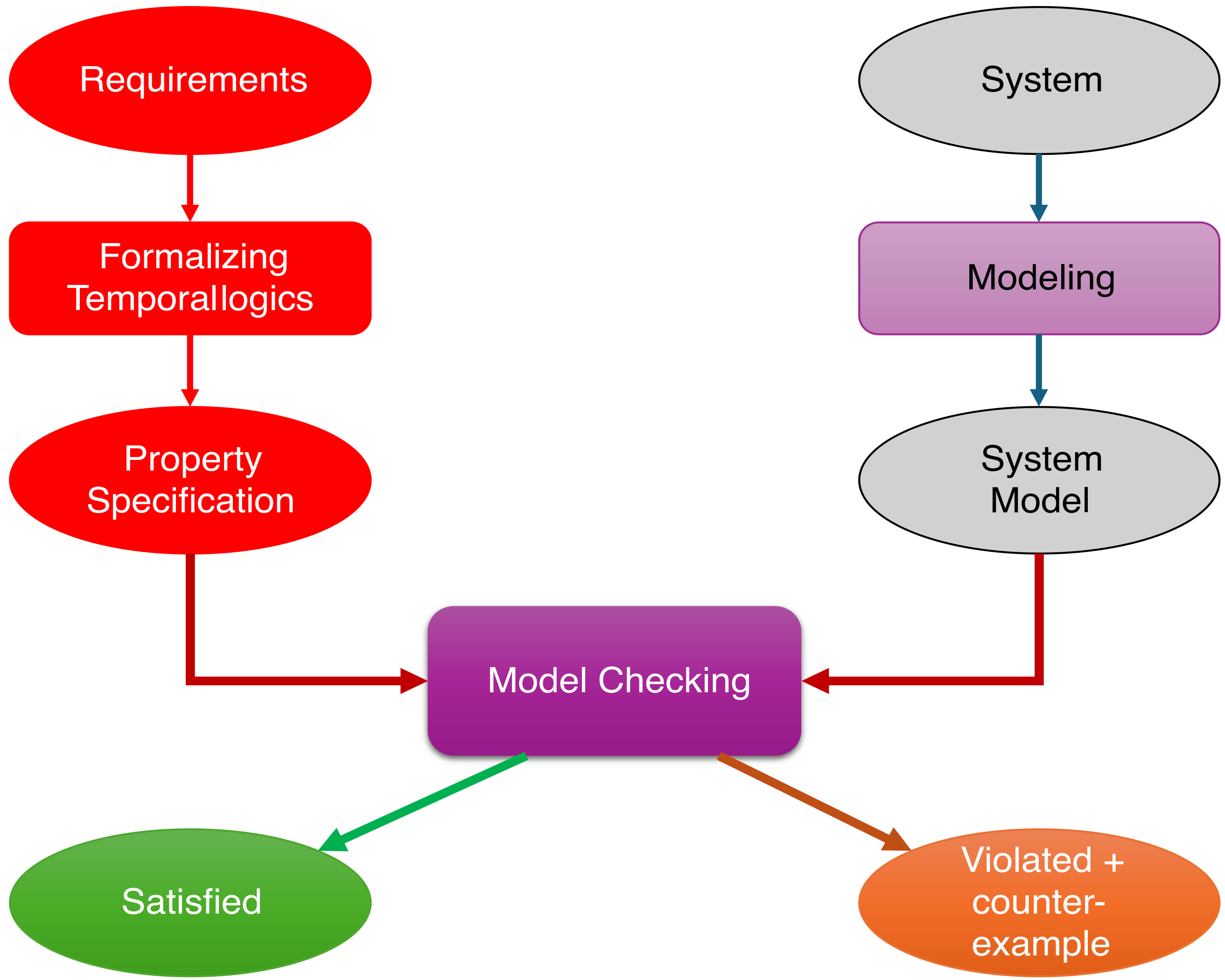
CentraleSupelec - Université Paris-Saclay - 2025/2026

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

PLAN

Retour au plan - Retour à l'accueil

PRINCIPE DU MODEL-CHECKING



LOGIQUES TEMPORELLES

POURQUOI ?

- Pas de variable pour gérer le temps (instants implicites)
- **Temporel $\neq$ temporisé**
  la logiques temporelles ne quantifient pas écoulement du temps.
- Deux approches :
  1. **temps linéaire** : propriétés des séquences d'exécutions (futur déterminé)
  2. **temps arborescent** : propriétés de l'arbre d'exécutions (tous les futurs possibles)

PLAN

Retour au plan - Retour à l'accueil

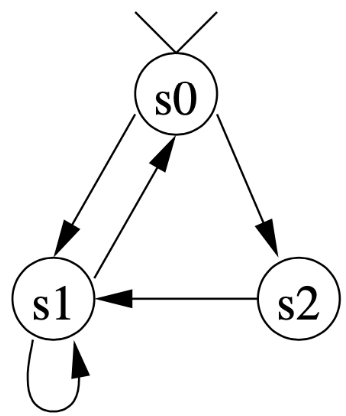
ARBRE DE CALCUL

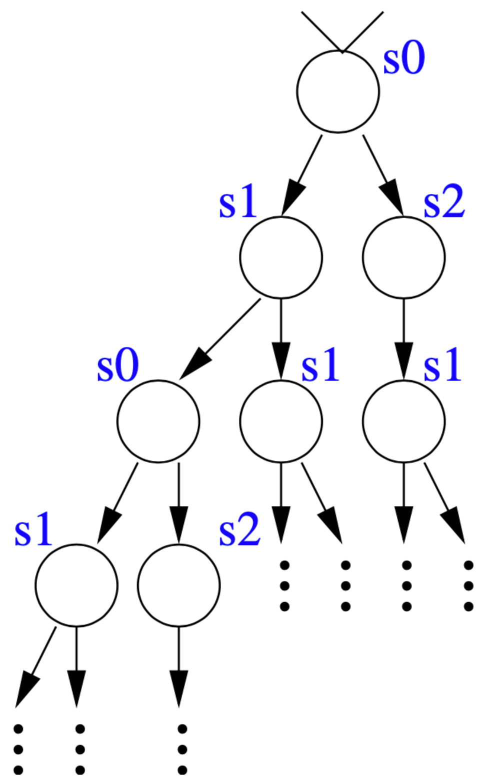
- Soit $\mathcal{T} = (S, \rightarrow, s^0)$ un système de transition.
  Intuitivement, l'**arbre de calcul** de $\mathcal{T}$ est le dépliage acyclique de $\mathcal{T}$.
- Formellement, le dépliage est le plus petit système de transition $(U, \rightarrow', u^0)$ (éventuellement infini) avec un étiquetage $l : U \rightarrow S$ tel que :
  - $u^0 \in U$ et $l(u^0) = s^0$
  - si $u \in U$, $l(u) = s$, et $s \rightarrow s'$ pour certains u, s, s'
    alors il existe $u' \in U$ avec $u \rightarrow' u'$ et $l(u') = s'$
  - u^0 n'a pas de prédécesseur direct, et tous les autres états de U ont exactement un prédécesseur direct

ARBRE DE CALCUL

EXEMPLE

Un système de transition et son arbre de calcul (étiquetage l donné en bleu)





ARBRE DE CALCUL

REMARQUES

- Pour la vérification de propriétés **CTL**, la construction de l'arbre de calcul n'est pas nécessaire ($\longrightarrow$ voir le cours du Model-Checking).
- Cependant, cette définition permet de clarifier les concepts sous-jacents aux **opérateurs de la logique CTL**.

PLAN

Retour au plan - Retour à l'accueil

LTL vs CTL

- **LTL - (Linear-Time Logic)**
  - Décrit les propriétés des exécutions individuelles.
  - Sémantique définie comme un ensemble d'exécutions.
- **CTL - (Computation Tree Logic)**
  - Décrit les propriétés d'un arbre de calcul.
    - ➠ les formules peuvent traiter plusieurs exécutions simultanément.
  - Sémantique définie en termes d'états.

COMPUTATION TREE LOGIC - CTL

APERÇU

- Combine les opérateurs temporels avec une quantification sur les exécutions
- Les opérateurs ont la forme suivante $\rightarrow Q\ T$
  - Q
    - E : there **exists** an execution
    - A : for **all** executions
  - T
    - $X \equiv \bigcirc$: next
    - $F \equiv \Diamond$: finally
    - $G \equiv \Box$: globally
    - $U \equiv \bigcup$: until
    - (et peut-être d'autres)

COMPUTATION TREE LOGIC - CTL

LA SYNTAXE

- Nous définissons d'abord une syntaxe minimale. Nous définissons ensuite des opérateurs supplémentaires à l'aide de cette syntaxe minimale.

- Soit AP un ensemble de propositions atomiques.
  L'ensemble des formules CTL sur AP est le suivant :

  si $a \in AP$, alors a est une formule CTL ;

  si $\phi_1, \ \phi_2$ sont des formules CTL, alors le sont aussi

  $$\neg\phi_1, \qquad \phi_1 \vee \phi_2,$$
  $$EX\ \phi_1, \qquad EG\ \phi_1, \qquad E\ (\phi_1\ U\ \phi_2),$$

COMPUTATION TREE LOGIC - CTL

LA SÉMANTIQUE

- soit $\mathcal{K} = (S, \rightarrow, s^0, AP, v)$ une structure de **Kripke**.
  - S : un ensemble d'états, $\rightarrow \ \in S \times S$: une relation entre états, s^0 : l'état initial, AP : ensemble des propositions atomiques, $v \in S \rightarrow 2^{AP}$: une fonction d'étiquetage

- Nous définissons la sémantique de chaque formule CTL ϕ sur AP par rapport à $\mathcal{K}$ comme un ensemble d'états $[\![\phi]\!]_{\mathcal{K}}$, comme suit :

$$
\begin{aligned}
[\![a]\!]_{\mathcal{K}} &= \{s \mid a \in v(s)\} \qquad a \in AP \\
[\![\neg\phi_1]\!]_{\mathcal{K}} &= S \backslash [\![\phi_1]\!]_{\mathcal{K}} \\
[\![\phi_1 \vee \phi_2]\!]_{\mathcal{K}} &= [\![\phi_1]\!] \cup [\![\phi_2]\!]_{\mathcal{K}} \\
[\![EX\ \phi_1]\!]_{\mathcal{K}} &= \{s \mid \text{il existe un état } t \text{ tel que } s \rightarrow t \text{ et } t \in [\![\phi_1]\!]_{\mathcal{K}}\} \\
[\![EG\ \phi_1]\!]_{\mathcal{K}} &= \{s \mid \text{il existe un chemin } \sigma \text{ tel que } \sigma(0) = s \text{ et } \sigma(i) \in [\![\phi_1]\!]_{\mathcal{K}}\ \forall i \geq 0\} \\
[\![E\ (\phi_1\ U\ \phi_2)]\!]_{\mathcal{K}} &= \{s \mid \text{il existe un chemin } \sigma \text{ tel que } \sigma(0) = s \text{ et } k \geq 0,\ \sigma(i) \in [\![\phi_1]\!]_{\mathcal{K}}\ \forall i < k,\ \sigma(k) \in [\![\phi_2
\end{aligned}
$$

COMPUTATION TREE LOGIC - CTL

OPÉRATEURS SUPPLÉMENTAIRES

$$false \equiv \neg true$$

$$\phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2)$$

$$\phi_1 \Rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$$

$$EF\ \phi \equiv E\ (true\ U\ \phi)$$

$$AX\ \phi \equiv \neg EX\ \neg\phi$$

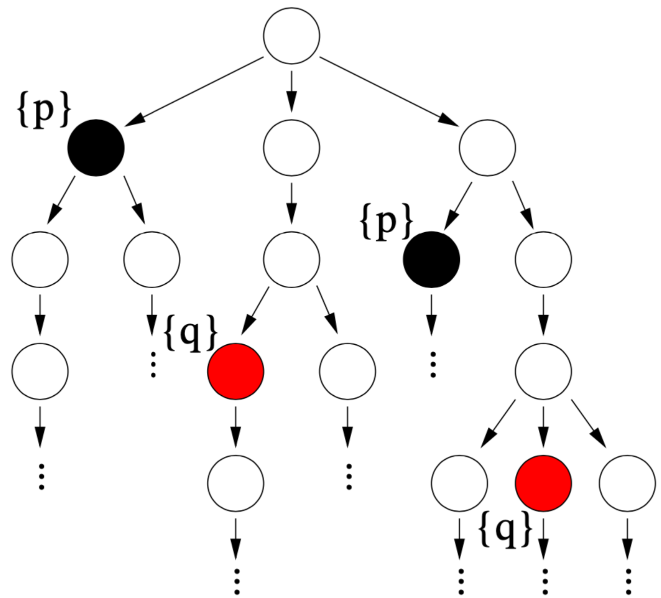
$$AG\ \phi \equiv \neg EF\ \neg\phi$$

$$AF\ \phi \equiv \neg EG\ \neg\phi$$

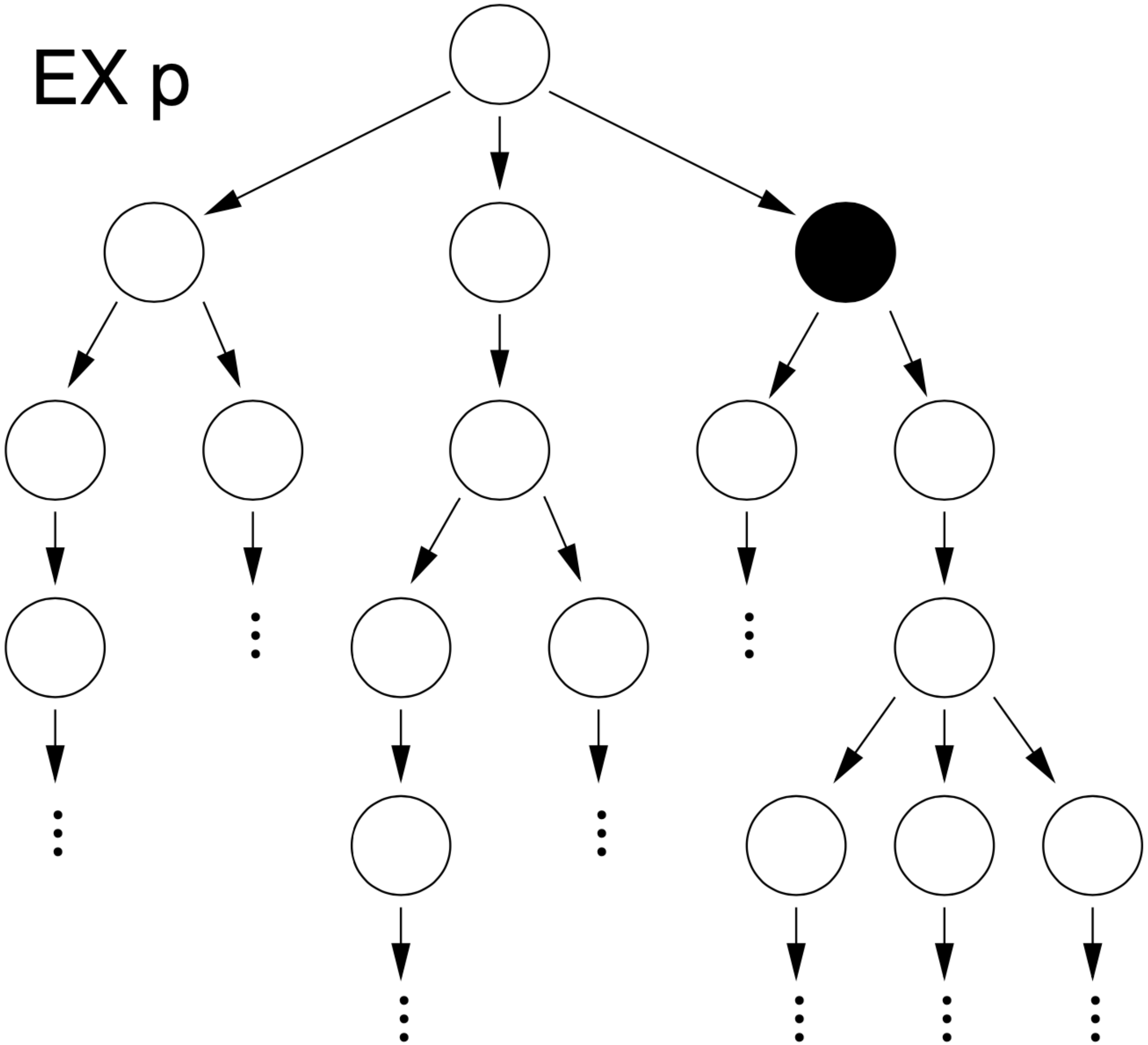
$$A\ (\phi_1\ U\ \phi_2) \equiv \neg E\ \neg(\phi_1\ U\ \phi_2)$$

OPÉRATEURS CTL ET ARBRES DE CALCUL

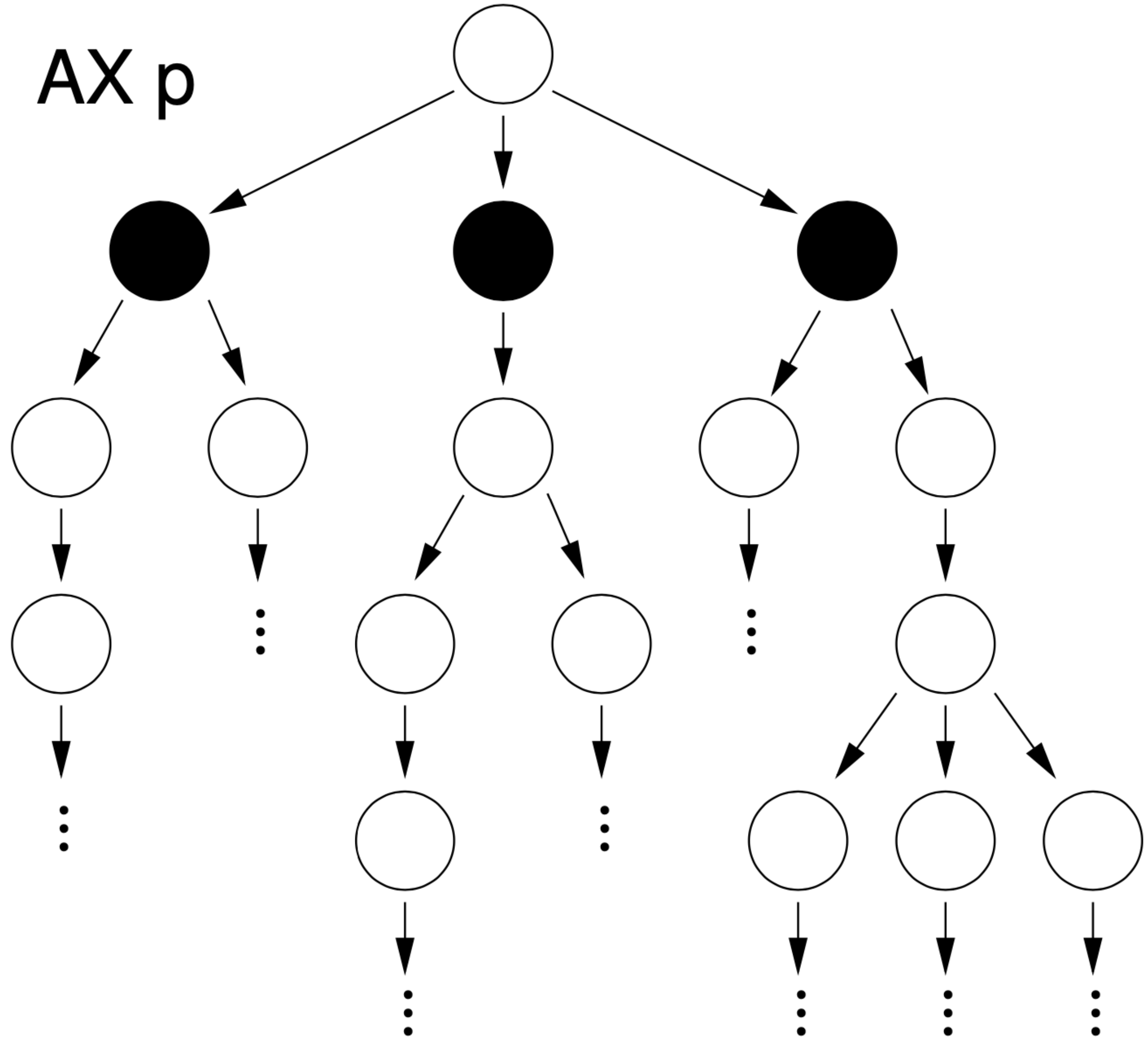
Nous utilisons l'arbre de calcul suivant comme exemple d'exécution (avec des distributions variables des états **rouge** et **noir**).

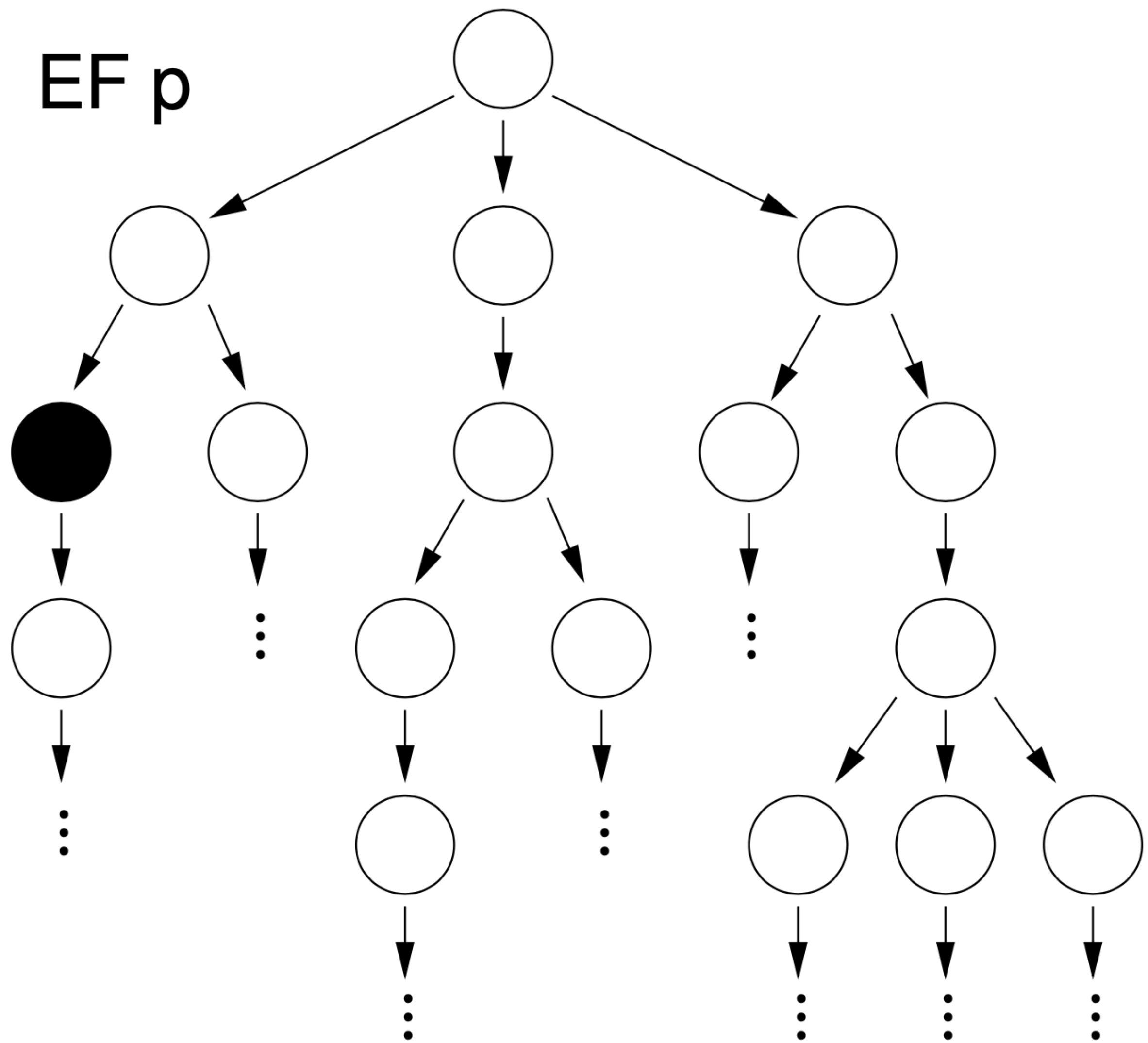


Dans les diapositives suivantes, l'état le plus élevé satisfait une formule donnée Les états **noirs satisfont** p et les états **rouges satisfont** q.

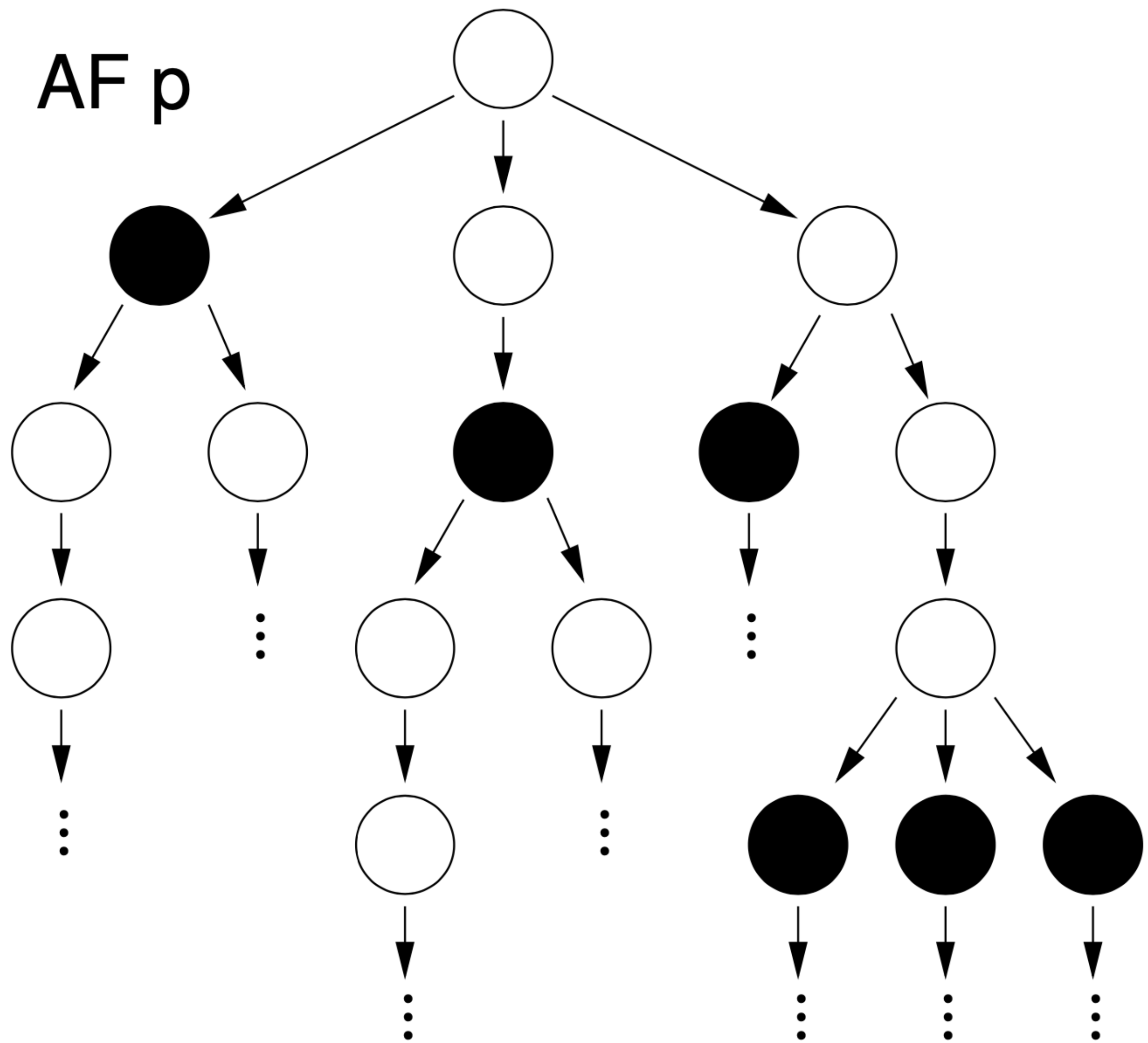

EX p

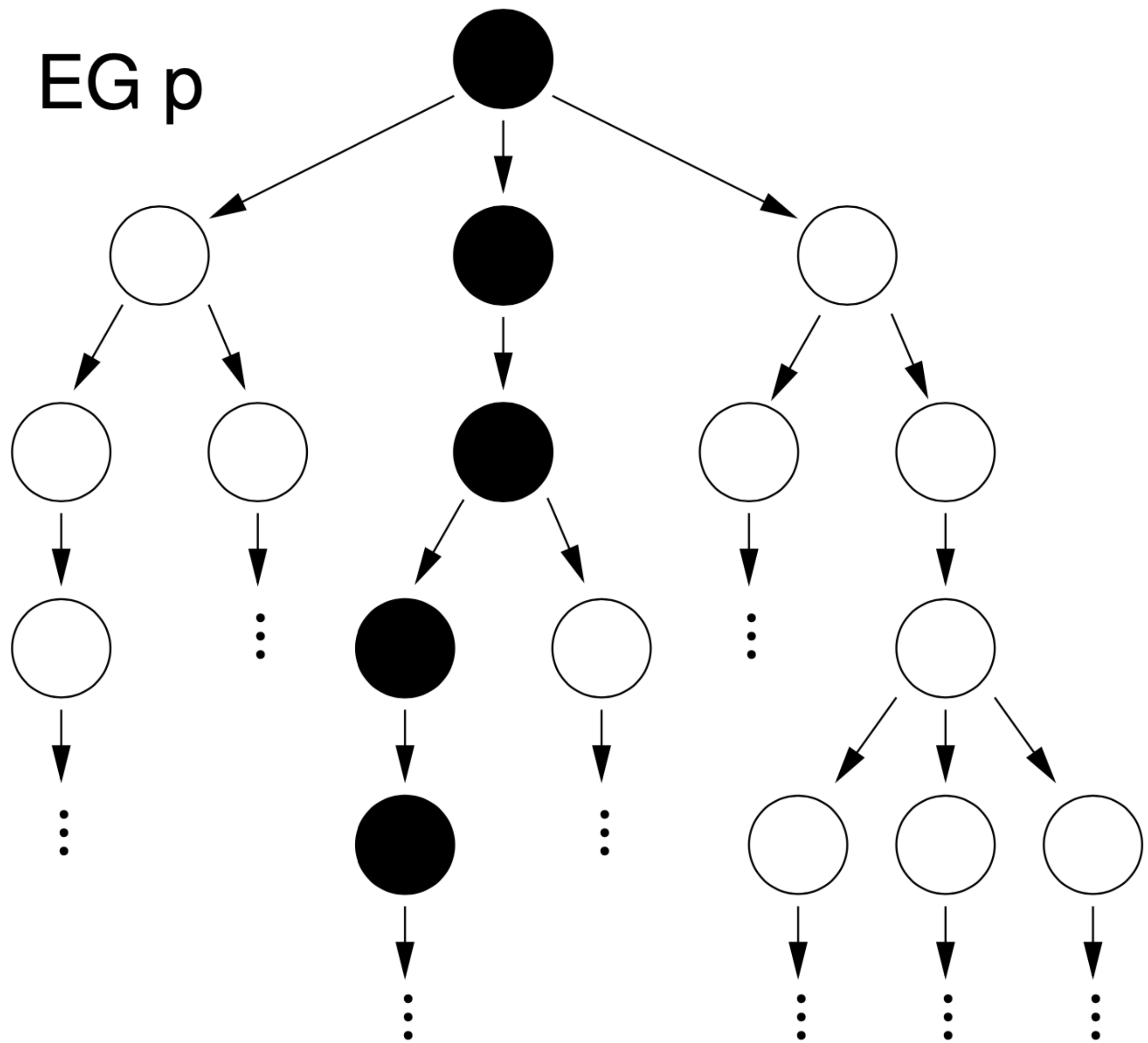
AX p



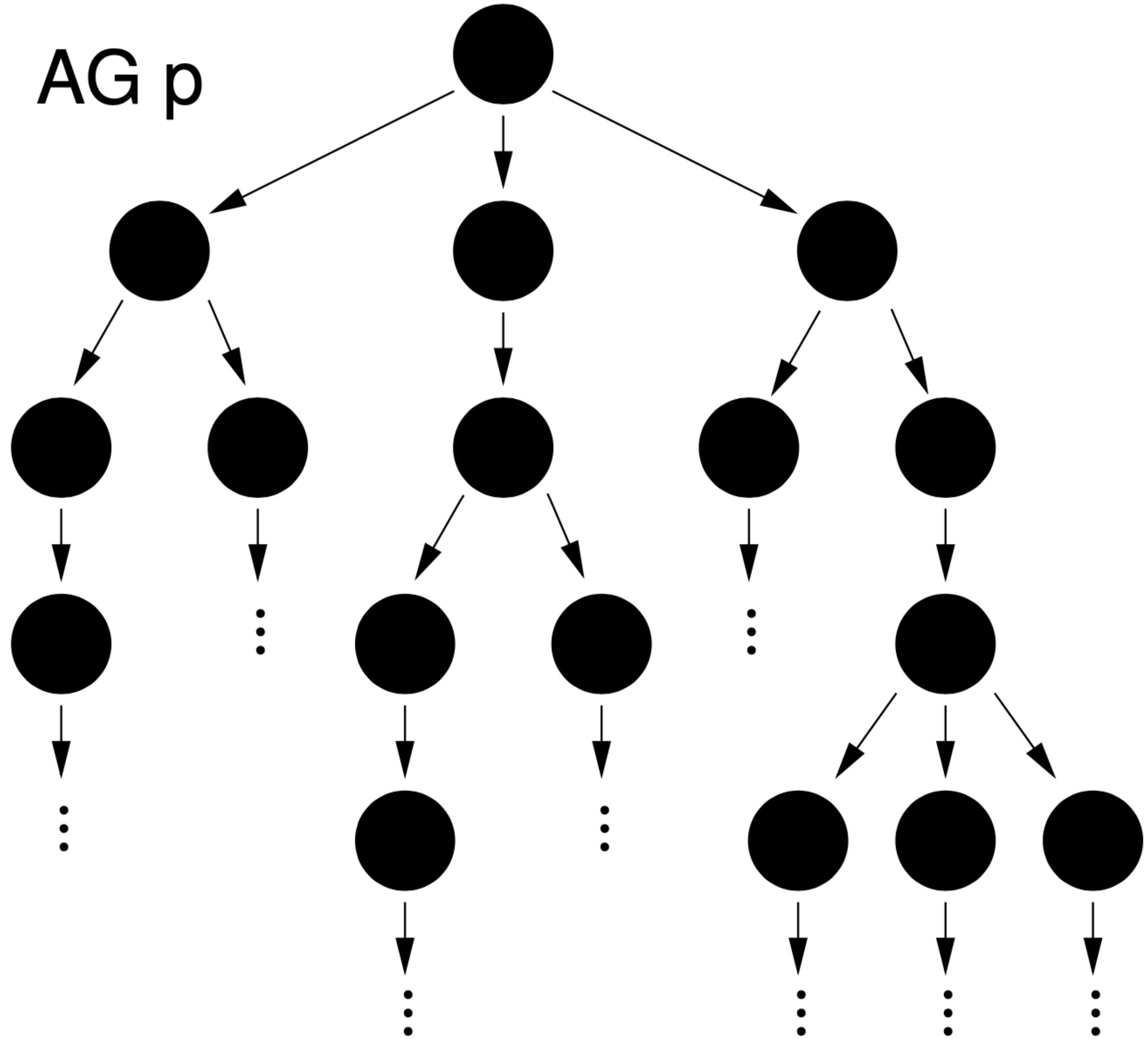

EF p

AF p

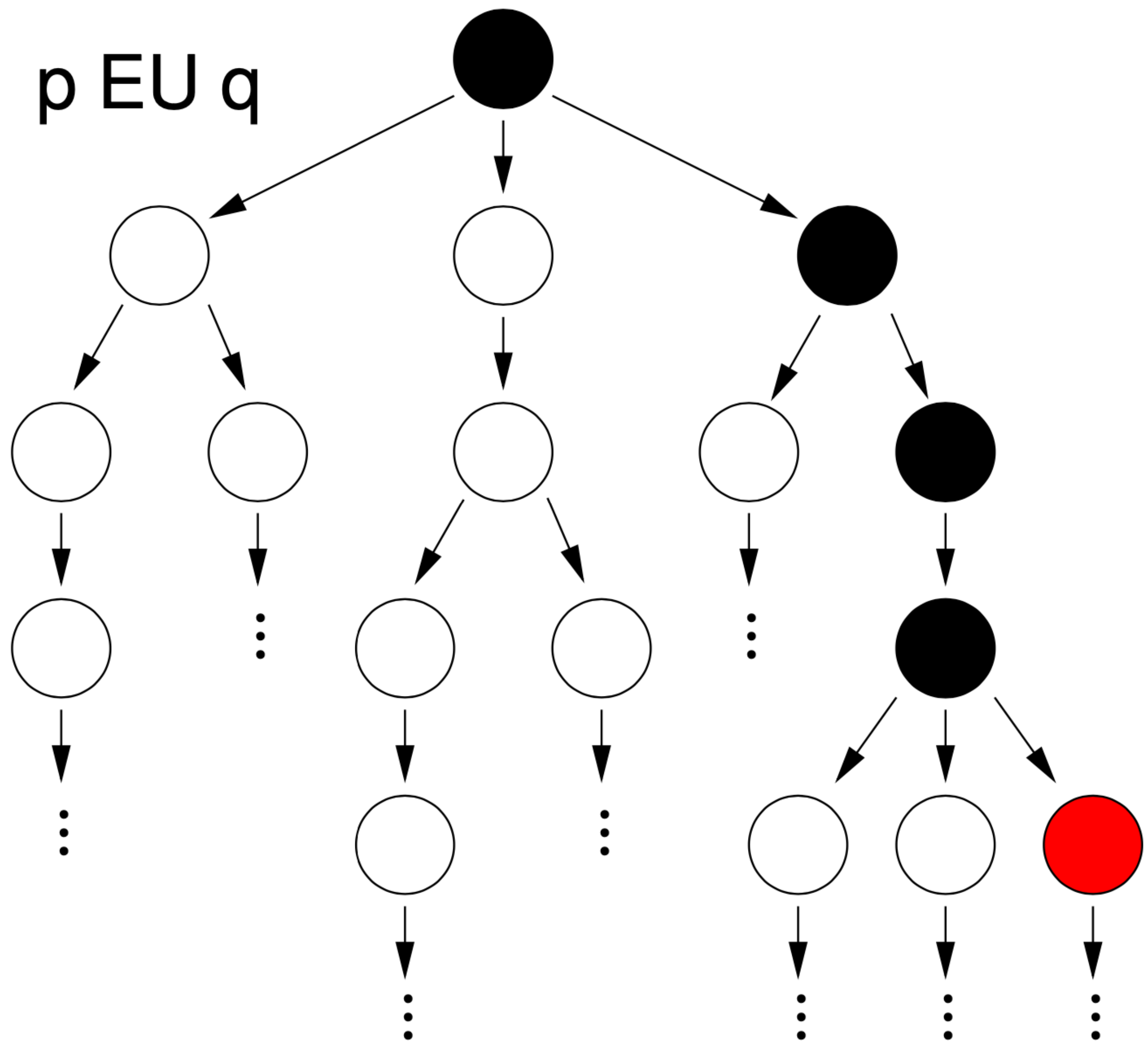


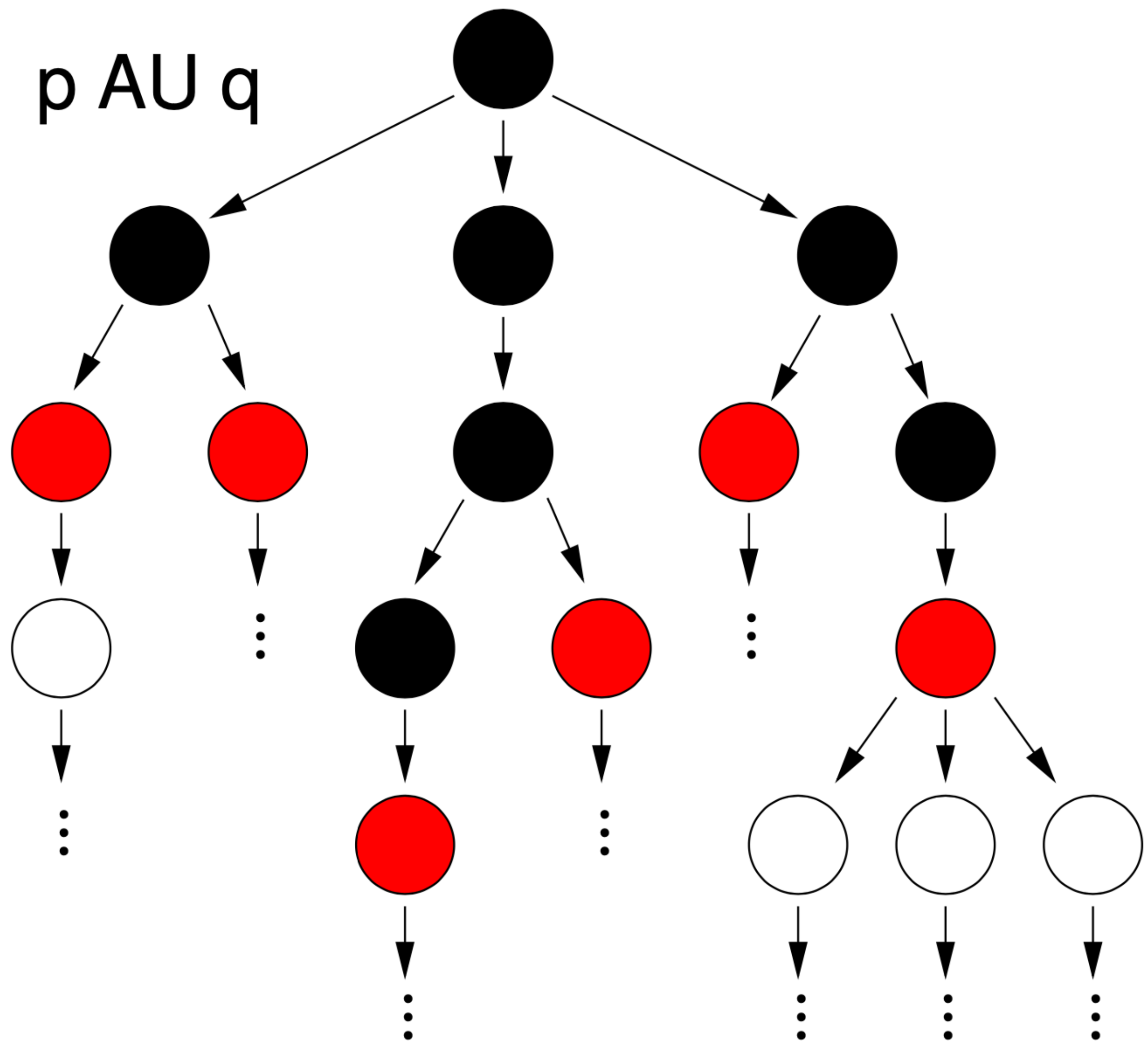

EG p

AG p



p EU q




p AU q

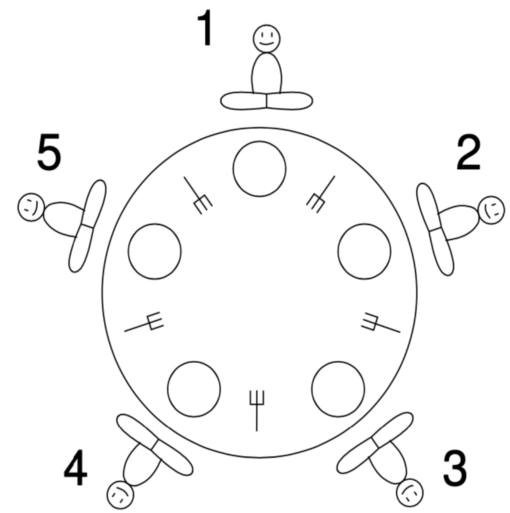
PLAN

- Arbre de calcul
- Présentation de la logique CTL
- Exemple : le dîner des philosophes
- Résolution de formules CTL

Retour au plan - Retour à l'accueil

LA SPÉCIFICATION

- Le problème du dîner des philosophes est un problème de **synchronisation** introduit par **Dijkstra** en 1965.



- Il illustre les problèmes du **partage des ressources** dans la **programmation concurrente**.
  - le blocage, la famine et l'exclusion mutuelle...

- Énoncé du problème :
  - K ($K = 5$) philosophes sont assis autour d'une table ronde.
  - chaque philosophe alterne entre réflexion et repas.
  - il y a une baguette entre chaque philosophe (K baguettes au total).
  - un philosophe doit prendre deux baguettes (gauche et droite) pour manger.
  - un seul philosophe peut utiliser une baguette à la fois.

QUELQUES PROPRIÉTÉS CTL

- Supposons les propositions atomiques suivantes :
  - $e_i \rightarrow$ le philosophe i est en train de manger
  - $f_i \rightarrow$ le philosophe i vient de finir de manger

- Les philosophes 1 et 4 ne mangeront jamais en même temps.

$$AG\ \neg(e_1 \wedge e_4)$$

- Chaque fois que le philosophe 4 a fini de manger, il ne peut plus manger tant que le philosophe 3 n'a pas mangé.

$$AG\ (f_4 \Rightarrow A\ (\neg e_4\ W\ e_3))$$

- Le philosophe 2 sera le premier à manger.

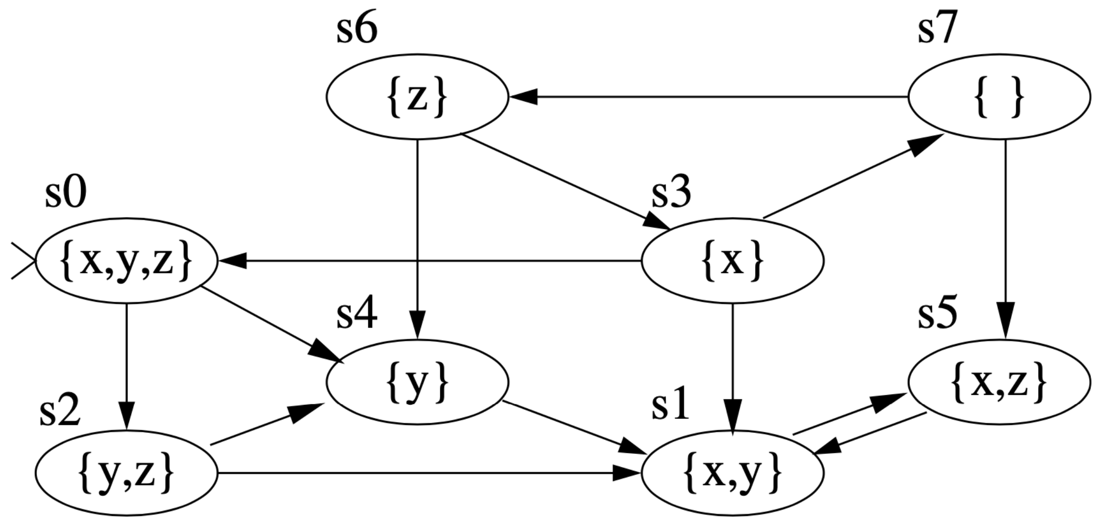
$$A(\neg(e_1 \vee e_3 \vee e_4 \vee e_5)\ U\ e_2)$$

PLAN

Retour au plan - Retour à l'accueil

EXEMPLE DE FORMULES IMBRIQUÉES

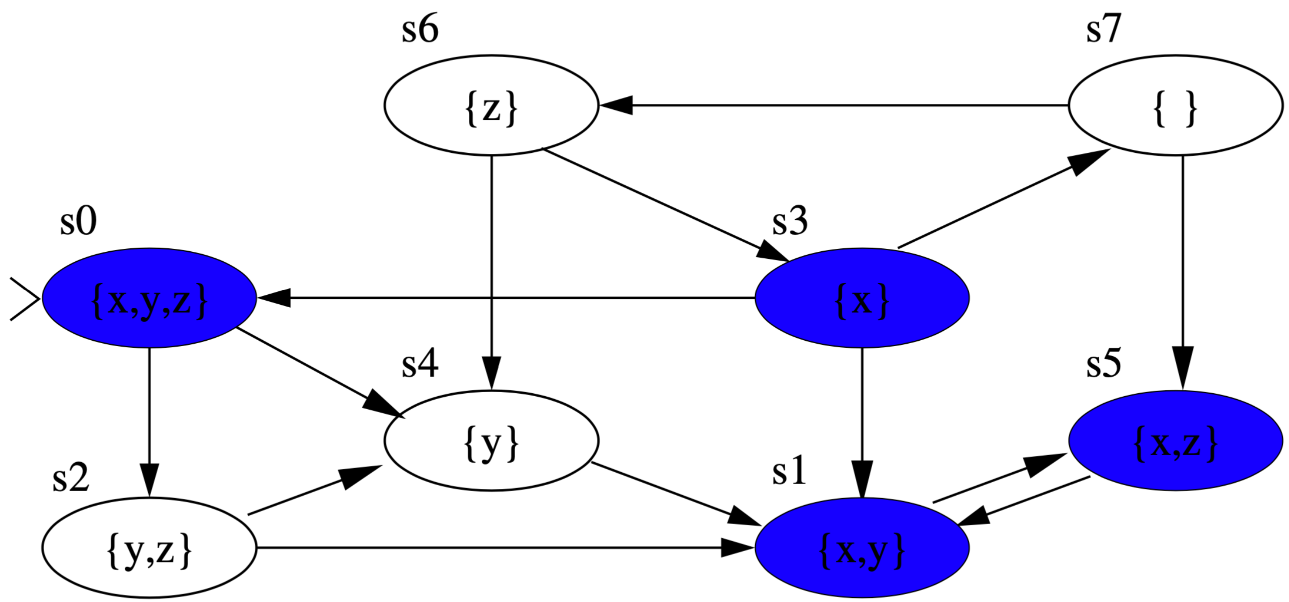
$$s^0 \in [\![AF\,AG\; x]\!]\ ?$$



- Pour calculer la sémantique des formules avec des **opérateurs imbriqués**,
  - nous calculons d'abord les états satisfaisant les formules les plus internes ;
  - ensuite, nous utilisons ces résultats pour résoudre les formules englobantes.
- Dans cet exemple, nous calculons $[\![x]\!]$, $[\![AG\; x]\!]$ et $[\![AF\,AG\; x]\!]$, dans cet ordre

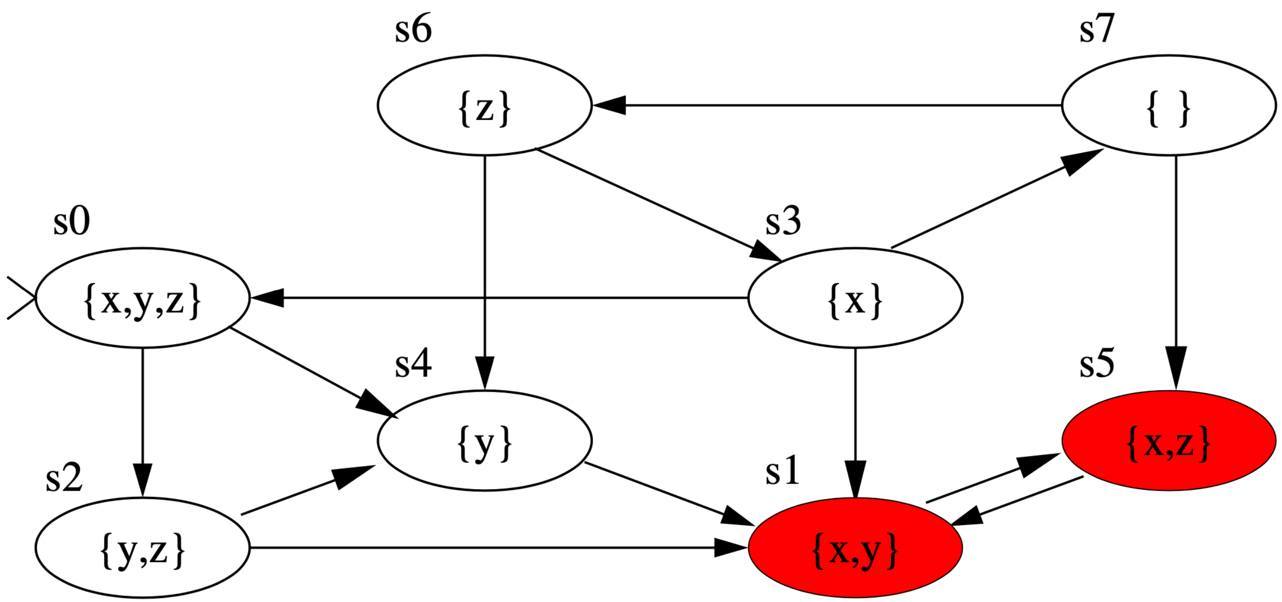
EXEMPLE DE FORMULES IMBRIQUÉES

Calcul de $[\![x]\!]$



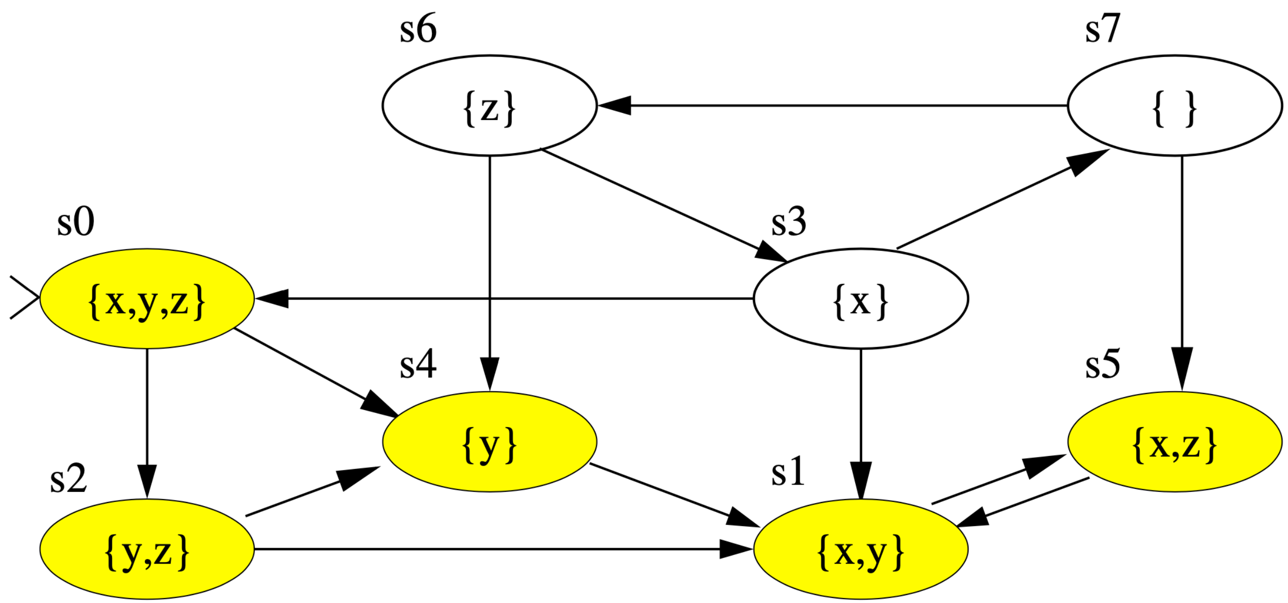
EXEMPLE DE FORMULES IMBRIQUÉES

Calcul de $[\![AG\ x\ ]\!]$ ou de $[\![\neg EF \neg x\ ]\!]$



EXEMPLE DE FORMULES IMBRIQUÉES

Calcul de $[\![AF AG\ x\ ]\!]$ ou de $[\![\neg EG \neg AG\ x\ ]\!]$



MERCI

PDF version of the slides

Retour à l'accueil - Retour au plan