Cours de Model Checking

Leçon 2.1 : logiques temporelles

Sébastien Bardin

CEA-LIST, Laboratoire de Sûreté Logicielle

sebastien.bardin@cea.fr
http://sebastien.bardin.free.fr/

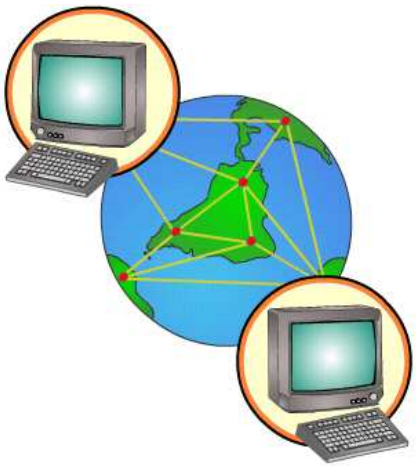
Plan

Cours 2.1 : logiques temporelles

- Rappel
- Logiques temporelles : intuition
- Préambule technique
- LTL, CTL* et CTL
- Comparaison
- Disgressions

Cours 2.2 : algorithmes

Systèmes réactifs



Systèmes réactifs II

Programme classique

- termine
- retourne un résultat
- données complexes, contrôle séquentiel ($\approx$ simple)

exemple : compilateur, algo de tri

Système réactif

- ne doit pas terminer
- ne retourne pas de résultat
- données simples, contrôle distribué ($\approx$ complexe)

exemple : protocole, OS

Systèmes réactifs III

Vérifier un programme classique : Aspect temporel toujours identique, mais les prédicats sur les données peuvent être complexes.

- *"Le programme termine et le tableau est trié"*.

Systèmes réactifs III

Vérifier un programme classique : Aspect temporel toujours identique, mais les prédicats sur les données peuvent être complexes.

- *"Le programme termine et le tableau est trié".*

Vérifier un système réactif : Aspect temporel très varié mais les prédicats sur les données sont souvent simples.

- *"Si un processus demande infiniment souvent à être exécuté, alors l'OS finira par l'exécuter".*
- *"Il est toujours possible de revenir à l'état initial".*
- *"Chaque fois qu'une panne est détectée, une alarme est émise".*
- *"Chaque fois qu'une alarme est émise, une panne a été détectée".*

Le model checking

Model Checking

Technique de vérification automatique de systèmes réactifs

Ingrédients

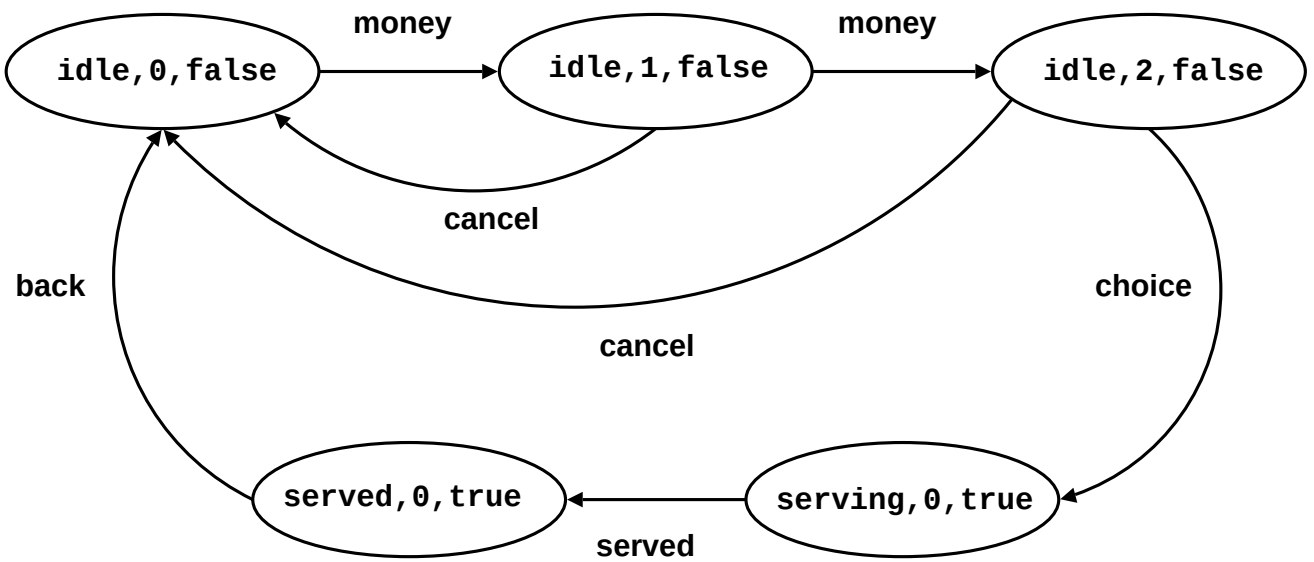
- $\mathcal{M}$ = système de transitions
- φ = formule temporelle
- MC = est-ce que $\mathcal{M} \models \varphi$?

Avantages

- Phases amonts : spécifs et design
- Automatisé
- Trouve mieux les bugs que le test
- Cost efficient (pour certains domaines)

Système de transitions

Système de transitions = comportement du système réactif



Propriétés sur les systèmes réactifs

- Accessibilité *Une certaine situation peut être atteinte*

  `x` peut valoir 0, toute instruction peut être exécutée
- Invariance *Chaque état local respecte une bonne propriété*

  `x` ne vaut jamais 0, le tableau ne déborde jamais
- Sûreté *Quelquechose de mauvais n'arrive jamais*

  j'accède au fichier uniquement si j'ai entré le bon PIN
- Vivacité *Quelquechose de bon finit par arriver*

  le programme termine, le message finit toujours par être transmis

  le programme revient toujours à l'état initial
- Équité *Quelquechose de bon se répète infiniment souvent*

  si un processus demande toujours la main, il l'aura infiniment

Logiques temporelles

On exprimera les propriétés grâce à des logiques temporelles

A, E, F, G, U, X

Avantages

- non ambigue
- générique
- ouvre la voie à la vérification automatique

Plusieurs logiques temporelles possibles

- LTL, CTL, CTL*, ...

La difficulté du model checking

2 difficultés principales

1. système de transitions fini
2. système de transitions suffisament petit

La difficulté du model checking

2 difficultés principales

1. système de transitions fini
2. système de transitions suffisament petit

Système fini : problème (parfois non trivial) de modélisation

- variables à domaines finis, nombre borné de tâches, etc.

La difficulté du model checking

2 difficultés principales

1. système de transitions fini
2. système de transitions suffisament petit

Système fini : problème (parfois non trivial) de modélisation

- variables à domaines finis, nombre borné de tâches, etc.

Système petit : problème algorithmique + modélisation

- 10 variables sur 8 bits : 10^{256} possibilités
- Stocké une possibilité = 10 octets
- Tout stocker $\approx 10^{245}$ To

La difficulté du model checking

2 difficultés principales

1. système de transitions fini
2. système de transitions suffisament petit

Système fini : problème (parfois non trivial) de modélisation

- variables à domaines finis, nombre borné de tâches, etc.

Système petit : problème algorithmique + modélisation

- 10 variables sur 8 bits : 10^{256} possibilités
- Stocké une possibilité = 10 octets
- Tout stocker $\approx 10^{245}$ To

Model Checking = gérer l'explosion d'états

En pratique

- standard en vérification de hardware (Intel, IBM, etc.)
- quelques beaux succès en software (drivers, fuzzing)

It has been an exciting twenty years, which has seen the research focus evolve [. . .] from a dream of automatic program verification to a reality of computer-aided design debugging.

- Thomas A. Henzinger (2001)

Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability.

- Bill Gates (2002)

Historique

1975	Constat : vérif. inadaptée à systèmes réactifs
1977	Pnueli propose d'utiliser les logiques temporelles
1981	Model checking de CTL par Clarke et al., Sifakis et al.
1980-1990	Nombreux résultats théoriques
1990-2000	Énorme amélioration des performances Extensions : proba, temps, infini
2000-...	MC adopté par les principaux fondeurs (Intel, etc.) Standardisation du langage temporel PSL Débuts du software model checking (Microsoft) Prix ACM Paris Kanellakis Award 1998 et 2005
2008	Prix Turing décerné à Clarke, Sifakis et Emerson

Bibliographie

Livres

 Principles of Model Checking [Baier-Kaoten 08]

 Model Checking [Clarke-Grumberg-Peled 99]

 Vérification de logiciels [LSV 99]

Plan

Exprimer des propriétés temporelles

Besoin d'exprimer des familles de propriétés et pas juste quelques cas particuliers

Langages naturels

- imprécis (donc pas d'automatisation)
- verbeux

Formalismes graphiques

- plus précis
- concis
- faciles à apprendre et à communiquer
- manque d'expressivité ou/et de précision

Exprimer des propriétés temporelles

Besoin d'exprimer des familles de propriétés et pas juste quelques cas particuliers

Langages naturels

- imprécis (donc pas d'automatisation)
- verbeux

Formalismes graphiques

- plus précis
- concis
- faciles à apprendre et à communiquer
- manque d'expressivité ou/et de précision

On se tourne vers des spécifications logiques

Logiques temporelles

Pourquoi des logiques ?

- exprimer sans ambiguité les propriétés attendues
- prouver la correction du système

Logique temporelle

- logique classique + opérateurs dédiés au temps
- connecteurs temporels + quantificateurs de chemins

Pourquoi des logiques temporelles ?

- concision, expressivité, simplicité
- algorithmique : décision et complexité

Logiques temporelles

Pourquoi des logiques ?

- exprimer sans ambiguité les propriétés attendues
- prouver la correction du système

Logique temporelle

- logique classique + opérateurs dédiés au temps
- connecteurs temporels + quantificateurs de chemins

Pourquoi des logiques temporelles ?

- concision, expressivité, simplicité
- algorithmique : décision et complexité

Attention

Temporel versus Temporisé

Logiques temporelles II

Deux sortes d'opérateurs dédiés au temps

Connecteurs temporels : sur un chemin

- suite d'évènements attendus le long d'un seul chemin
- nous verrons entre autre **U**, **X**, **G**, **F**

insuffisant : en général on veut savoir si tout ou partie des chemins partant d'un état donné vérifient une propriété

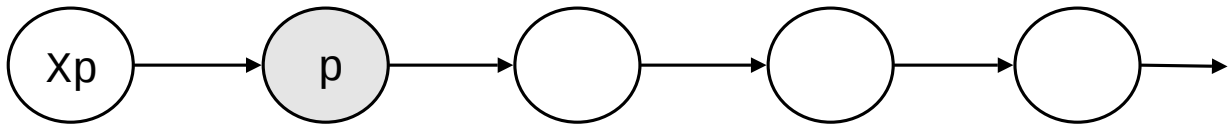
Quantificateurs de chemin : sur un (dépliage du) système de transition

- quantifie les chemins partant d'un état qui doivent vérifier la propriété
- nous verrons **A**, **E**

Connecteurs temporels

Exprimer le séquencement d'évènements le long d'un chemin

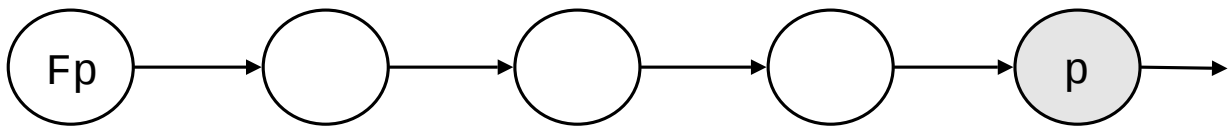
Opérateur **X** "next"



Connecteurs temporels

Exprimer le séquencement d'évènements le long d'un chemin

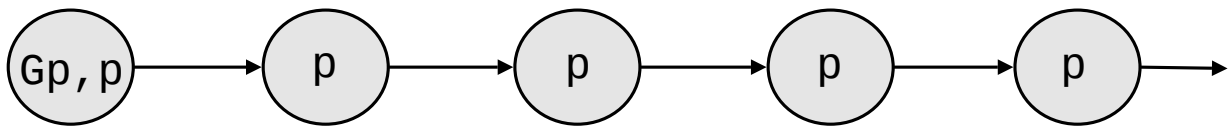
Opérateur **F** "sometimes in the future"



Connecteurs temporels

Exprimer le séquencement d'évènements le long d'un chemin

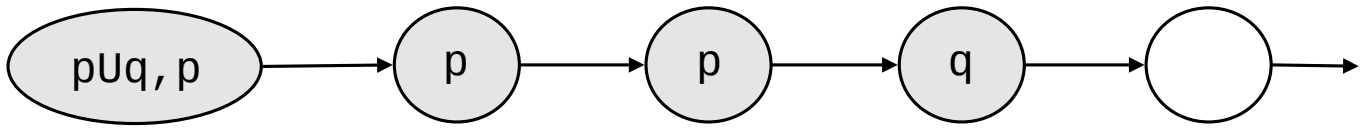
Opérateur **G** "always in the future"



Connecteurs temporels

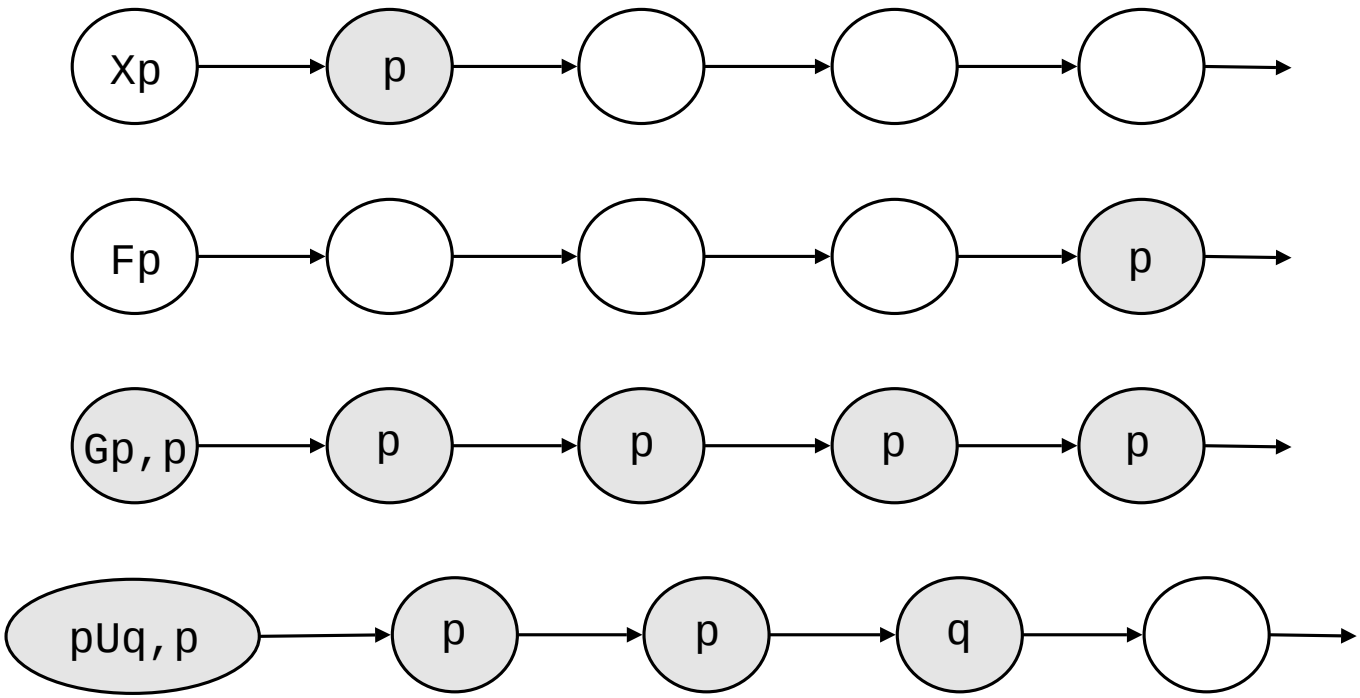
Exprimer le séquencement d'évènements le long d'un chemin

Opérateur **U** "p true until q true"



Connecteurs temporels

Exprimer le séquencement d'évènements le long d'un chemin



Quantificateurs de chemin

Les connecteurs temporels :

- considèrent une exécution à la fois
- exécutions indépendantes les unes des autres
- exécutions organisées en un ensemble
- Le futur est déterminé

Quantificateurs de chemin

Les connecteurs temporels :

- considèrent une exécution à la fois
- exécutions indépendantes les unes des autres
- exécutions organisées en un ensemble
- Le futur est déterminé

On peut vouloir parler des futurs possibles, selon les choix d'action du système

- certains états d'exécution ont le choix entre $\neq$ futurs
- exécutions interdépendantes
- exécutions organisées en arbre

Quantificateurs de chemin

Les connecteurs temporels :

- considèrent une exécution à la fois
- exécutions indépendantes les unes des autres
- exécutions organisées en un ensemble
- Le futur est déterminé

On peut vouloir parler des futurs possibles, selon les choix d'action du système

- certains états d'exécution ont le choix entre $\neq$ futurs
- exécutions interdépendantes
- exécutions organisées en arbre

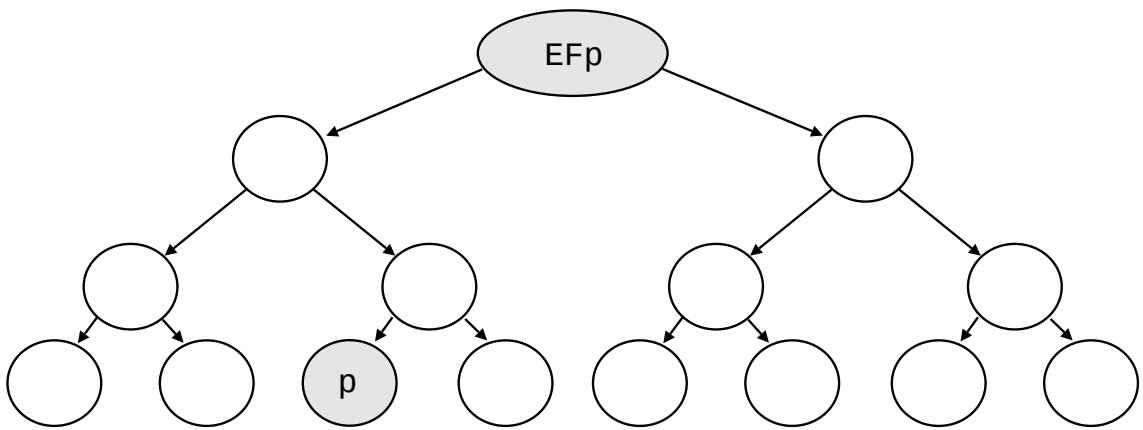
On introduit les quantificateurs de chemins

- **A** : tous les chemins futurs vérifient la propriété
- **E** : il existe un chemin futur qui vérifie la propriété

Quantificateurs de chemin II

quantificateurs de chemins + connecteurs temporels

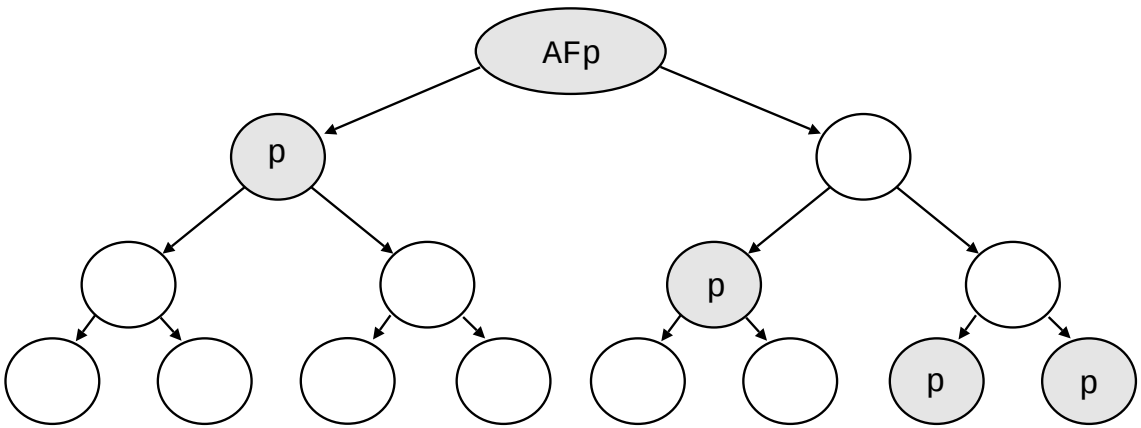
EF*p* : *p* vrai dans au moins un état



Quantificateurs de chemin II

quantificateurs de chemins + connecteurs temporels

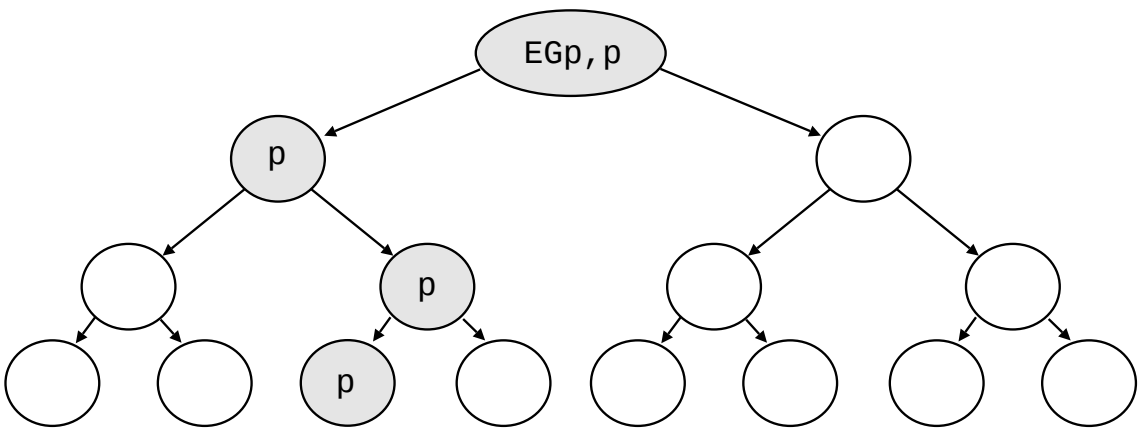
AF*p* : *p* atteignable par tous les chemins



Quantificateurs de chemin II

quantificateurs de chemins + connecteurs temporels

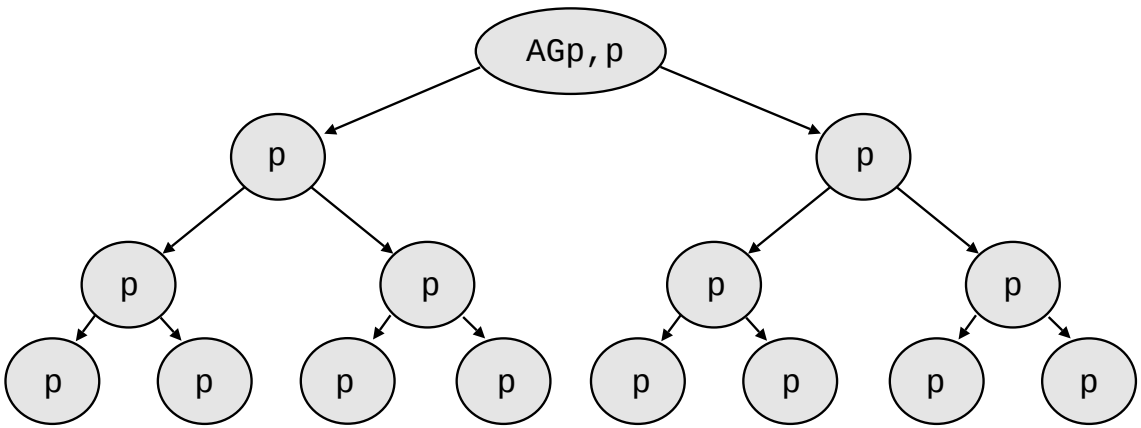
EG*p* : il existe un chemin avec *p* toujours vrai



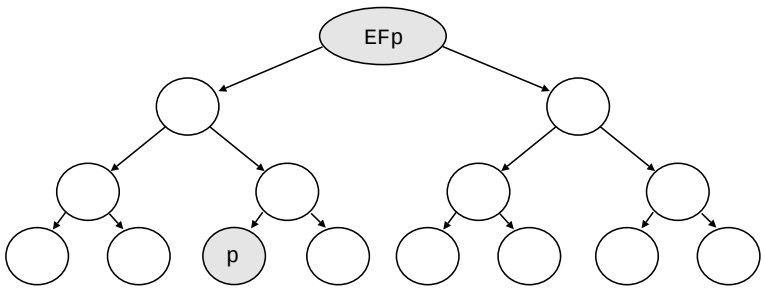
Quantificateurs de chemin II

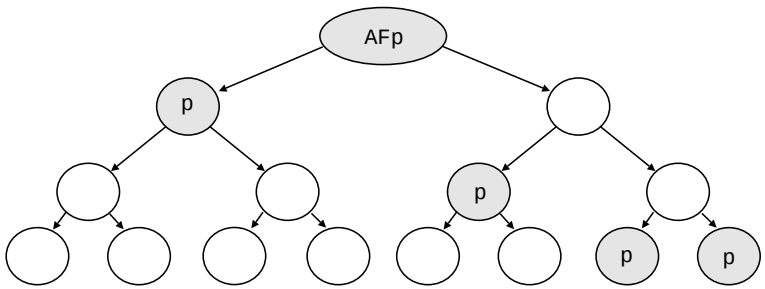
quantificateurs de chemins + connecteurs temporels

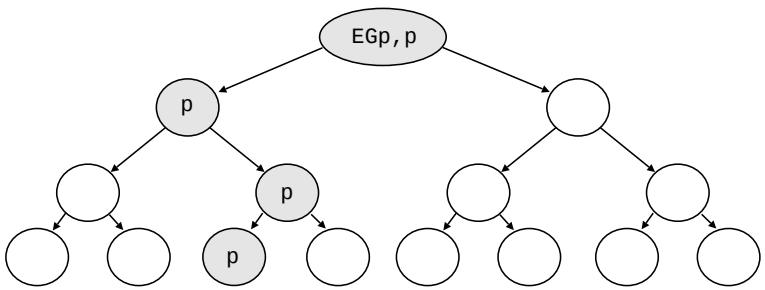
AGp : p est toujours vrai

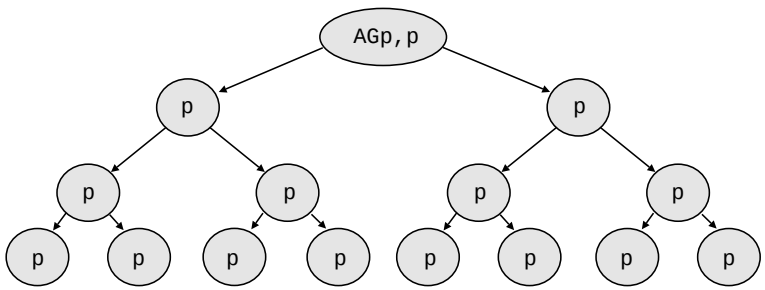


Quantificateurs de chemin III









Quelques exemples

- Accessibilité : $\mathbf{EF}(x = 0)$
- Invariance : $\mathbf{AG}\neg(x = 0)$
- Vivacité : $\mathbf{AG}(p \rightarrow \mathbf{F}q)$
- Correction totale :
  $\mathbf{A}((\text{init} \wedge \text{precondition}) \rightarrow \mathbf{F}(\text{end} \wedge \text{postcondition}))$
- Équité :
  $\mathbf{A}(\mathbf{GF}(\textit{execution request}) \rightarrow \mathbf{GF}(\textit{process scheduled}))$

Différentes logiques temporelles

On distingue plusieurs logiques temporelles

- Linéaire vs Branchant
  - capacité à parler des futurs possibles
- Expressivité
  - syntaxique ou/et sémantique
- Concision
- Avec ou sans passé
- Complexité de la vérification

Plan

Structure de Kripke

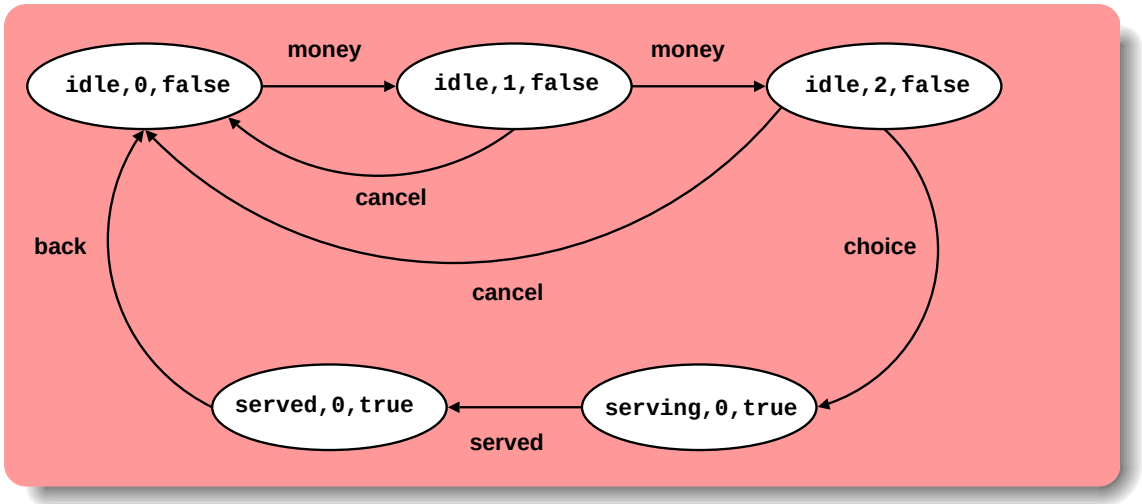
Pour le model checking on va considérer des structures de Kripke $\mathcal{M}$ plutôt que des systèmes de transitions S

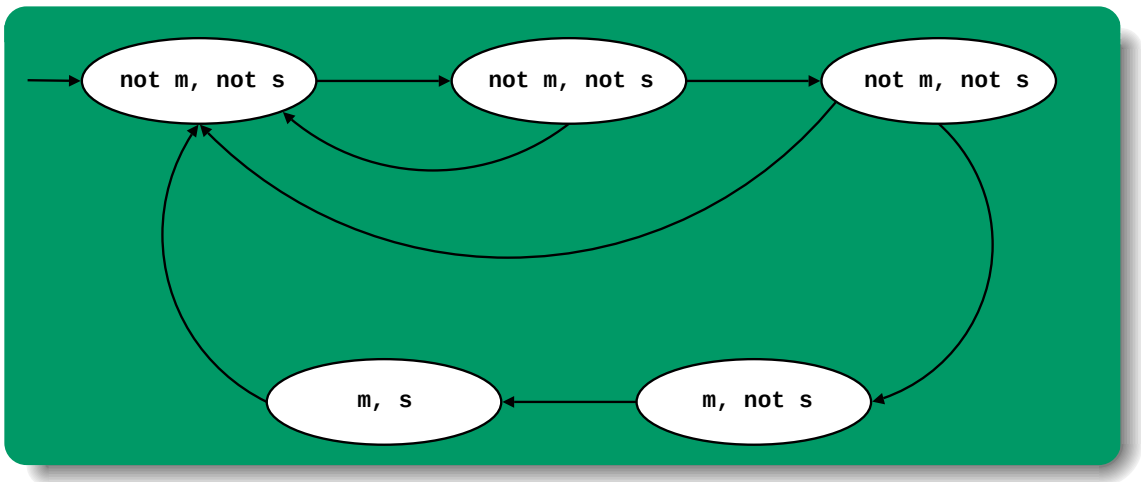
$\mathcal{M} = \langle Q, \rightarrow, \mathrm{AP}, l, s_0 \rangle$

- Q ensemble des états
- $\rightarrow \subseteq Q \times Q$ fonction de transition
- AP ensemble des *propositions atomiques*
- $l : Q \mapsto 2^{\mathrm{AP}}$ fonction d'étiquetage des états
- $s_0 \in Q$ état initial

Dualité propriétés sur états / transitions

Structure de Kripke II





Logique

Ingrédients d'une logique

- Un ensemble L de formules φ
- Un domaine D d'interprétations $\mathcal{I}$
- Une relation de satisfaisabilité $\models \subseteq D \times L$

On dit que

- $\mathcal{I}$ est un modèle de φ si $\mathcal{I} \models \varphi$
- φ est satisfaisable si il existe $\mathcal{I}$ tq $\mathcal{I} \models \varphi$
- φ est valide si pour tout $\mathcal{I}$, $\mathcal{I} \models \varphi$
- φ est contradictoire si aucun $\mathcal{I}$ ne la satisfait

Soit $\mathcal{L}$ une logique et $\varphi \in \mathcal{L}$

- $[\![\varphi]\!]$ = ensemble des solutions de φ

Exemple : Logique propositionnelle classique

ensemble fini $A_1, \ldots, A_n$ de propositions atomiques

Langage des formules logiques

$$p ::= A_i \mid \top \mid \bot$$

$$\varphi ::= \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid p$$

Domaine d'interprétation. $\mathcal{I}$ = valuation booléenne des A_i

Satisfaisabilité $\mathcal{I} \models \varphi$

$\mathcal{I} \models \top$

$\mathcal{I} \not\models \bot$

$\mathcal{I} \models A_i$ si $\mathcal{I}(A_i) = 1$

$\mathcal{I} \models f_1 \wedge f_2$ si $\mathcal{I} \models f_1$ **et** $\mathcal{I} \models f_2$

$\mathcal{I} \models f_1 \vee f_2$ si $\mathcal{I} \models f_1$ **ou** $\mathcal{I} \models f_2$

$\mathcal{I} \models \neg f$ si $\mathcal{I} \not\models f$

Exemples : $A \vee \neg A$ valide, A satisfaisable, $A \wedge \neg A$ contradictoire

Plan

Pour les logiques temporelles

Domaine d'interprétation principal : structure de Kripke $\mathcal{M}$

On définit la relation de satisfaction en trois étapes

- def de satisfaction d'une formule de chemin ($\approx$ sans **A** ni **E**) sur un chemin σ
- def de satisfaction d'une formule d'état ($\approx$ avec **A** ou **E**) sur $(\mathcal{M}, s)$ (à partir des formules de chemin)
- ensuite, $\mathcal{M}$ satisfait φ si $(\mathcal{M}, s_0)$ satisfait φ

Linear Temporal Logic

LTL est une logique dite linéaire

- **A** très restreint, **E** interdit
- formules $\mathbf{A}\varphi_p$, avec φ_p sans **A**, **E**
- *"tous les chemins vérifient φ_p"*

Exemples

- **AFG**p dans LTL
- **EF**p pas dans LTL

LTL sur un seul chemin

LTL sur un seul chemin σ

Formules

$\varphi ::= p \in \mathrm{AP} \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi\mathbf{U}\varphi$

Domaine d'interprétation= chemins σ

Satisfaisabilité

- $\sigma \models p$ iff $p \in l(\sigma(0))$
- $\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$
- $\sigma \models \varphi_1 \vee \varphi_2$ iff $\sigma \models \varphi_1$ ou $\sigma \models \varphi_2$
- $\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ et $\sigma \models \varphi_2$
- $\sigma \models \mathbf{X}\varphi$ iff $\sigma^1 \models \varphi$
- $\sigma \models \mathbf{F}\varphi$ iff il existe $k \geq 0$ tel que $\sigma^k \models \varphi$
- $\sigma \models \mathbf{G}\varphi$ iff pour tout $k \geq 0$ on a $\sigma^k \models \varphi$
- $\sigma \models \varphi_1\mathbf{U}\varphi_2$ iff il existe $k \geq 0$ tel que $\sigma^k \models \varphi_2$ et pour tout $0 \leq j < k$ $\sigma^j \models \varphi_1$

LTL sur un seul chemin II

Quelques liens entre opérateurs de chemins :

- $\mathbf{F}\varphi \equiv \neg\mathbf{G}\neg\varphi$
- $\mathbf{F}\varphi \equiv \mathit{true}\mathbf{U}\varphi$
- $\neg\mathbf{F}\varphi \equiv \mathbf{G}\neg\varphi$
- $\neg\mathbf{X}\varphi \equiv \mathbf{X}\neg\varphi$

LTL sur un seul chemin III

D'autres opérateurs (sucre syntaxique) :

- $\mathbf{F}^{\leq k}$
- $\mathbf{F}^{\infty}$
- $\mathbf{G}^{\infty}$
- $\mathbf{W}$
- $\mathbf{R}$

LTL sur structure de Kripke

LTL sur une structure de Kripke $\mathcal{M} = \langle Q, \rightarrow, \mathrm{AP}, l, s_0 \rangle$

Formules

$\varphi_s ::= \mathbf{A}\varphi_p$

$\varphi_p ::= p \in \mathrm{AP}$

$\quad | \neg\varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \wedge \varphi_p \mid \mathbf{X}\varphi_p \mid \mathbf{F}\varphi_p \mid \mathbf{G}\varphi_p \mid \varphi_p \mathbf{U} \varphi_p$

Domaine d'interprétation= couple $(\mathcal{M}, s)$

Satisfaisabilité

. $\mathcal{M}, s \models_s \mathbf{A}\varphi_p$ ssi tous les chemins σ partant de s vérifient $\sigma \models \varphi_p$

. $\sigma \models \varphi_p$ défini comme avant

Finalement $\mathcal{M} \models \varphi$ ssi $\mathcal{M}, s_0 \models_s \varphi$

Computational Tree Logic*

CTL* est une logique *branchante* très expressive

- **E**, **A** utilisés librement
- aucune restriction sur tous les opérateurs vus

Exemples

- toute combinaison (valide) des opérateurs vus en cours est une formule CTL*

Formalisation de CTL*

Formules : distinction formules d'états / de chemins

formules d'état (φ_s), interprétées sur les états de la structure de Kripke
formules de chemin (φ_p), interprétées sur les chemins de la structure de Kripke

$$\varphi_s ::= p \in \mathrm{AP} \mid \neg\varphi_s \mid \varphi_s \vee \varphi_s \mid \varphi_s \wedge \varphi_s \mid \mathbf{A}\varphi_p \mid \mathbf{E}\varphi_p$$
$$\varphi_p ::= \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \wedge \varphi_p \mid \mathbf{X}\varphi_p \mid \mathbf{F}\varphi_p \mid \mathbf{G}\varphi_p \mid \varphi_p \mathbf{U} \varphi_p$$

Le domaine d'interprétation est encore un couple $(\mathcal{M}, s)$ associant une structure de Kripke et un état

Formalisation de CTL* (II)

Satisfaisabilité

formules d'état, relation $\models_s$

- $\mathcal{M}, s \models_s p \in \mathrm{AP}$ iff $p \in l(s)$
- $\mathcal{M}, s \models_s \mathbf{A}f$ ssi tous les chemins σ partant de s vérifient $\mathcal{M}, \sigma \models_p f$
- $\mathcal{M}, s \models_s \mathbf{E}f$ ssi il existe un chemins σ partant de s vérifiant $\mathcal{M}, \sigma \models_p f$

formules de chemin, relation $\models_p$

- $\mathcal{M}, \sigma \models_p f$ iff $\mathcal{M}, \sigma(0) \models_s f$ (f formule d'état)
- $\mathcal{M}, \sigma \models_p \mathbf{X}\varphi$ iff $\mathcal{M}, \sigma^1 \models_p \varphi$
- $\mathcal{M}, \sigma \models_p \mathbf{F}\varphi$ iff il existe $k \geq 0$ tel que $\mathcal{M}, \sigma^k \models_p \varphi$

Finalement $\mathcal{M} \models \varphi$ ssi $\mathcal{M}, s_0 \models_s \varphi$

Formalisation de CTL* (III)

Quelques liens entre quantificateurs de chemins :

- $\neg \mathbf{E}\varphi \equiv \mathbf{A}\neg\varphi$
- $\neg \mathbf{A}\varphi \equiv \mathbf{E}\neg\varphi$

Attention :

- $\mathbf{A}\varphi$ implique $\mathbf{E}\varphi$ seulement si il existe un chemin partant de l'état courant

Computational Tree Logic

CTL est une restriction branchante de CTL*

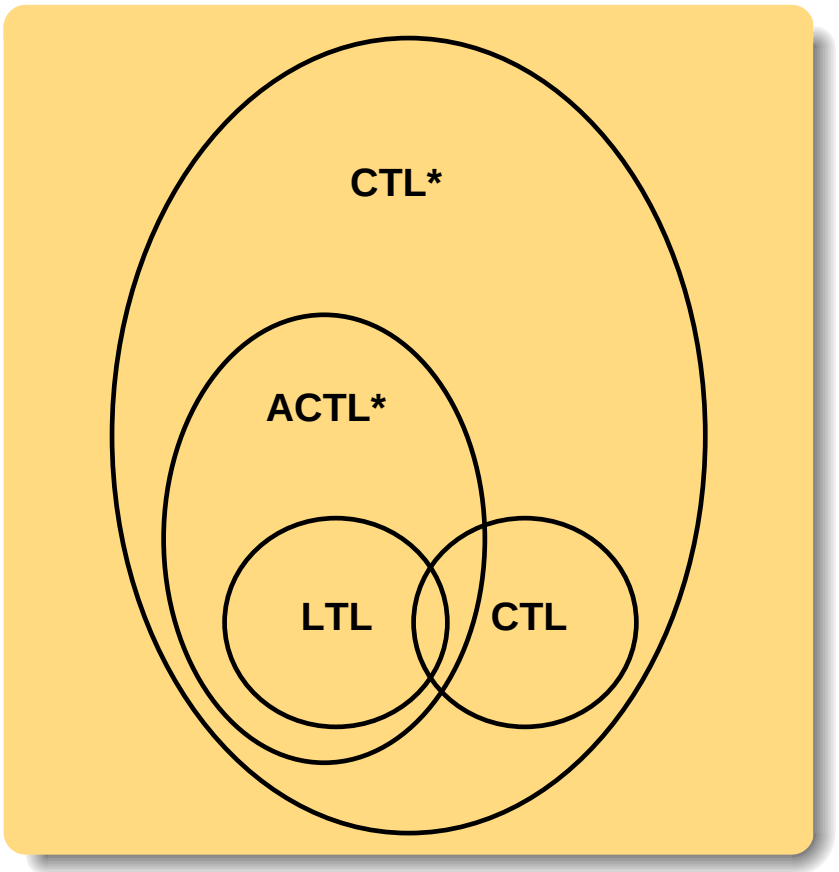
- **A**, **E** libres
- **X**, **F**, **G**, **U** doivent être précédés directement de **A**, **E**

Exemples

- **EF**(**AG**p) dans CTL
- **E**(**G**$p \wedge$ **X**q) pas dans CTL, **AFG**p pas dans CTL

Plan

Expressivité



En pratique

LTL

- expressif sur un chemin (diverses équités)
- intuitif pour interface, environnement, équivalence, contre-ex
- aucune expressivité sur les futurs possibles
- model checking coûteux (PSPACE-complet)

CTL*

- très très expressif
- model checking pas plus coûteux que LTL (mais coûteux)
- pas intuitif pour interface, environnement, équivalence, contre-ex

CTL

- le model checking est très efficace (PTIME-complet)
- on peut ajouter un peu d'équité et garder les performances
- manque un peu d'expressivité en "linéaire"
- pas intuitif pour interface, environnement, équivalence, contre-ex

Plan

Autres logiques

Modifications pour (essayer de) gagner sur le ratio expressivité / complexité

- fair CTL, ECTL, CTL+, ECTL+, ACTL*, LTL + regexp

D'autres formalismes

- automates
- (Hierchical) Message Sequences Charts
- μ-calcul

Exprimer plus que la simple causalité

- logiques temporisées
- logiques temporelles probabilistes
- logiques temporelles comptantes

Équité

Problème des chemins aberrants dus à la modélisation

- chemins impossibles ou très peu probables
- peuvent fausser l'analyse

Exemples

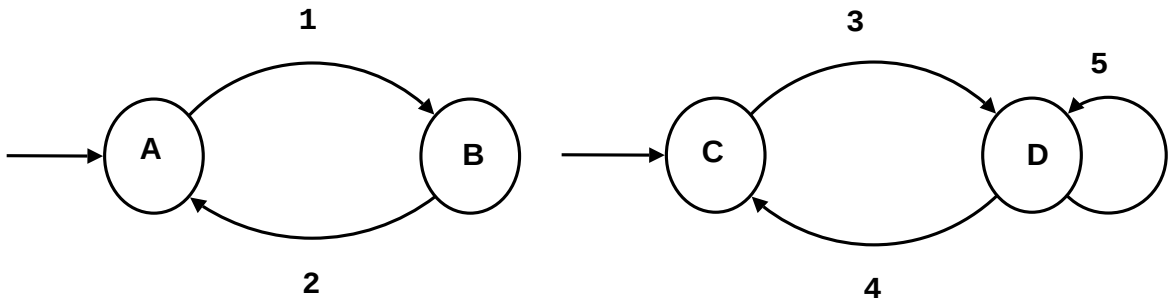
- un des composants n'a jamais la main
- un canal de communication perd systématiquement les messages

On rajoute une hypothèse d'équité H au système S

- H du genre : S "progresse" régulièrement
- permet d'écarter certains comportements aberrants
- H dans formule *(leçon 2)* ou dans modèle *(leçon 3)*

Très courant en pratique : ex des machines concurrentes

Équité II



En sémantique asynchrone

- le chemin 1.2.1.2.1.2.1.2. . . . (infini) est un chemin légal du système
- chemin certainement irréaliste
- rajoute une sorte de deadlock sur la machine 2

Contraintes d'équité

- sur le modèle : passer ∞ souvent par A, B, C, D
- dans la formule (but $= \varphi$) : $\mathbf{F}^{\infty}(A) \wedge \mathbf{F}^{\infty}(D) \implies \varphi$

Variantes sur le problème du model checking

- Model checking : $(\mathcal{M}, \varphi) \mapsto \mathcal{M} \models \varphi$?
- Inférence : $\mathcal{M} \mapsto \varphi$ tq. $\mathcal{M} \models \varphi$
- Vérification paramétrée : $(\mathcal{M}_k, \varphi) \mapsto \{k, \mathcal{M}_k \models \varphi\}$
- Synthèse : $\varphi \mapsto \mathcal{M}$ tq. $\mathcal{M} \models \varphi$