Cours de Model Checking

Leçon 2 : Logiques temporelles

Sébastien Bardin

CEA-LIST, Laboratoire de Sûreté Logicielle

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

Model Checking

Technique de vérification automatique de systèmes réactifs

Ingrédients

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

Vu au dernier cours

- modélisation
- calcul de l'espace des états
- accessibilité, invariance, sûreté

Types de propriétés temporelles

- Accessibilité

  Une certaine situation peut être atteinte
- Invariance

  Tous les états du système satisfont une bonne propriété
- Sûreté

  Quelquechose de mauvais n'arrive jamais
- Vivacité

  Quelquechose de bon finit par arriver
- Équité

  Quelquechose de bon se répète infiniment
- Équivalence comportementale

  Est-ce que 2 systèmes sont équivalents ?

Types de propriétés temporelles

- Accessibilité

  Une certaine situation peut être atteinte
- Invariance

  Tous les états du système satisfont une bonne propriété
- Sûreté

  Quelquechose de mauvais n'arrive jamais
- Vivacité

  Quelquechose de bon finit par arriver
- Équité

  Quelquechose de bon se répète infiniment
- Équivalence comportementale

  Est-ce que 2 systèmes sont équivalents ?

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

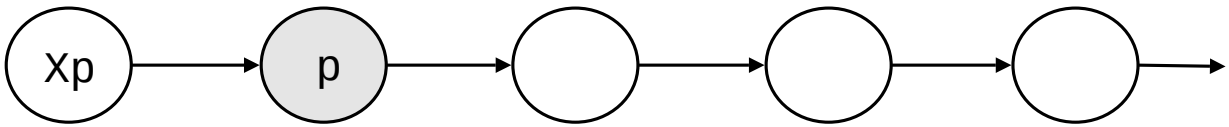
Quantificateurs de chemin : sur un (dépliage du) système de transition

- 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

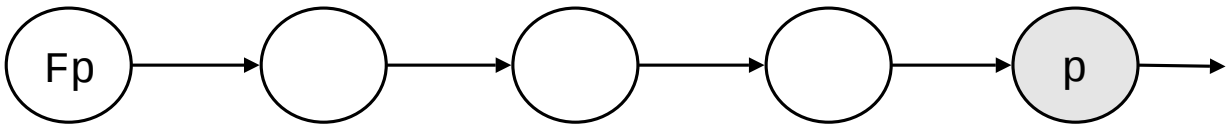
Opérateur **X** "next"



Connecteurs temporels

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

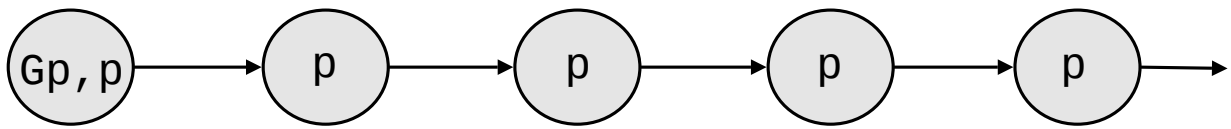
Opérateur **F** "sometimes in the future"



Connecteurs temporels

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

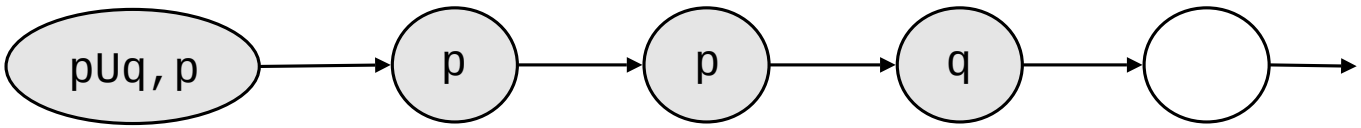
Opérateur **G** "always in the future"



Connecteurs temporels

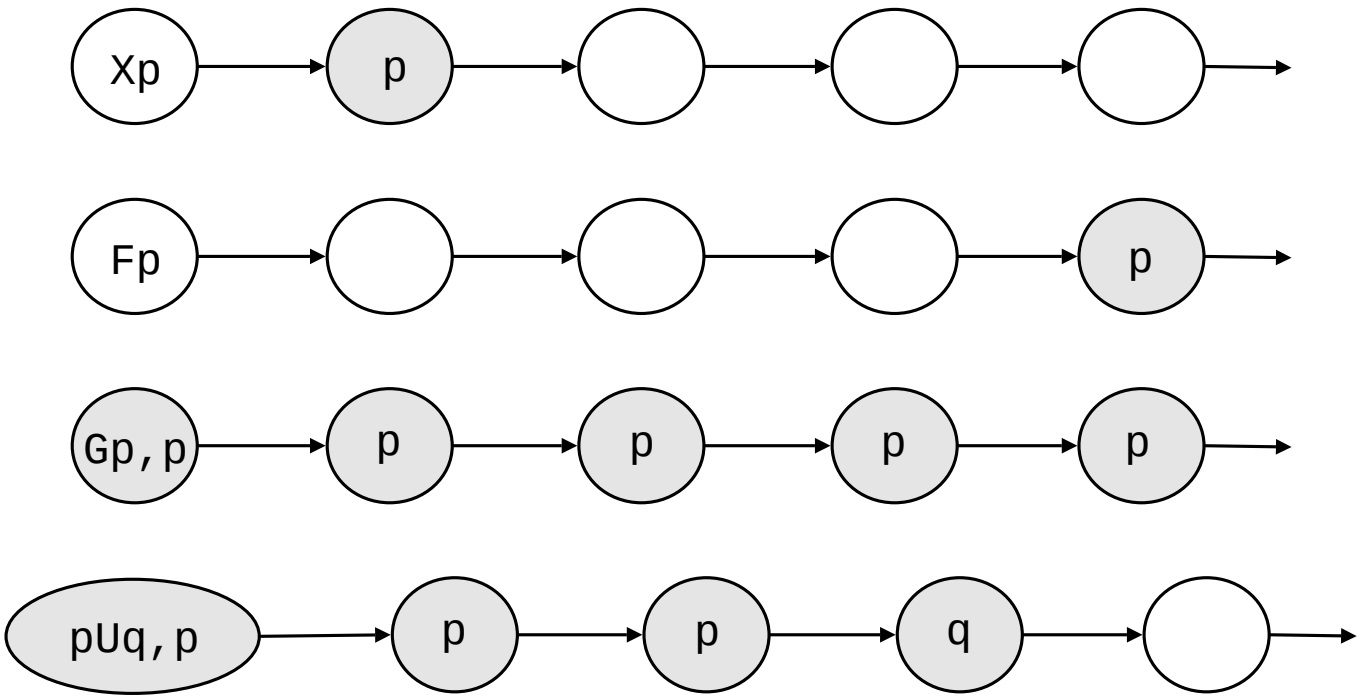
Exprimer le séquencement d'évènements le long d'un chemin

Opérateur **U** "p true until q true"



Connecteurs temporels

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



Connecteurs temporels II

Exemples pour un seul chemin

- Accessibilité : $\mathbf{F}(x = 0)$
- Invariance : $\mathbf{G}\neg(x = 0)$
- Vivacité : $\mathbf{G}(p \rightarrow \mathbf{F}q)$
- Correction totale :
  $(\text{init} \wedge \text{precondition}) \rightarrow \mathbf{F}(\text{end} \wedge \text{postcondition})$

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

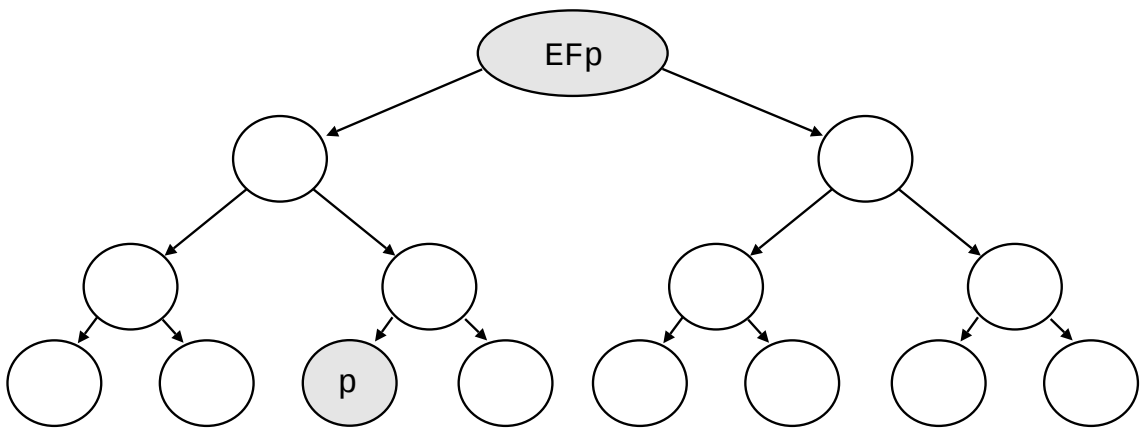
On introduit les quantificateurs de chemins

- **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

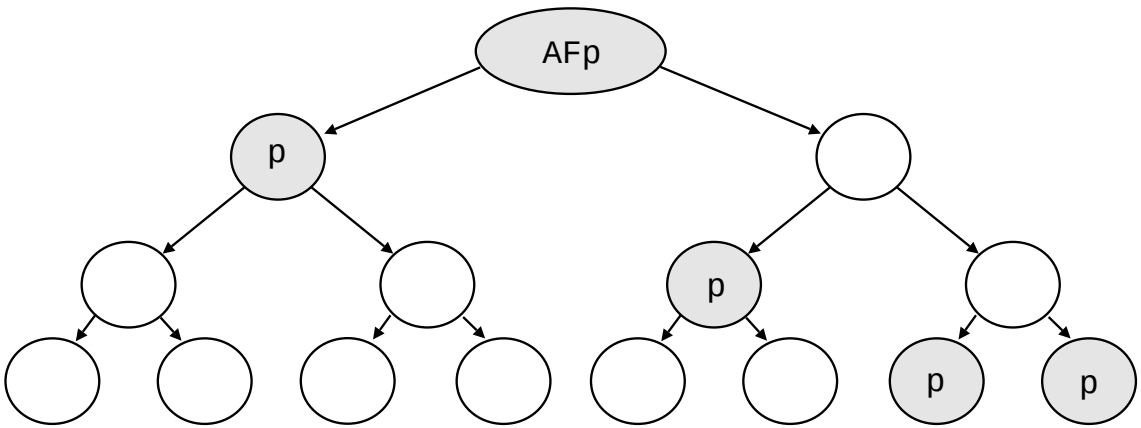
EF*p* : *p* vrai dans au moins un état



Quantificateurs de chemin II

quantificateurs de chemins + connecteurs temporels

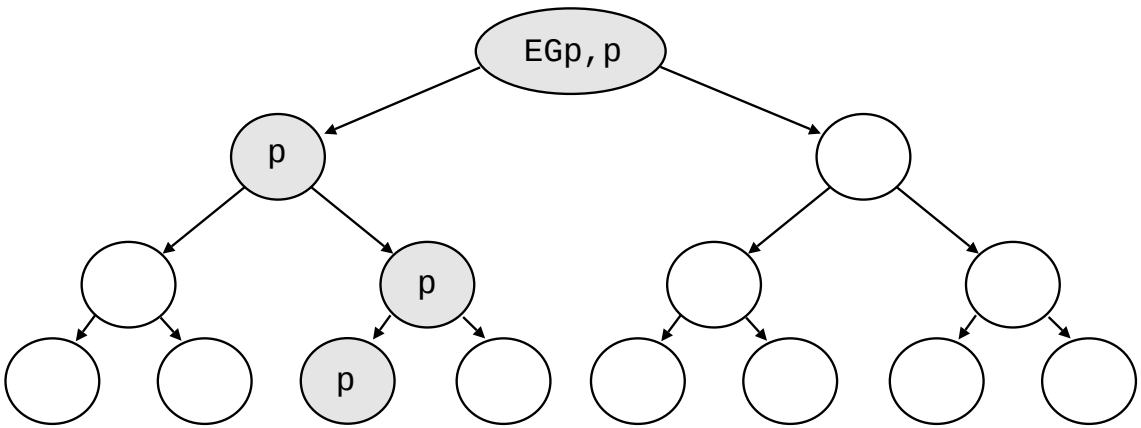
AFp : p atteignable par tous les chemins



Quantificateurs de chemin II

quantificateurs de chemins + connecteurs temporels

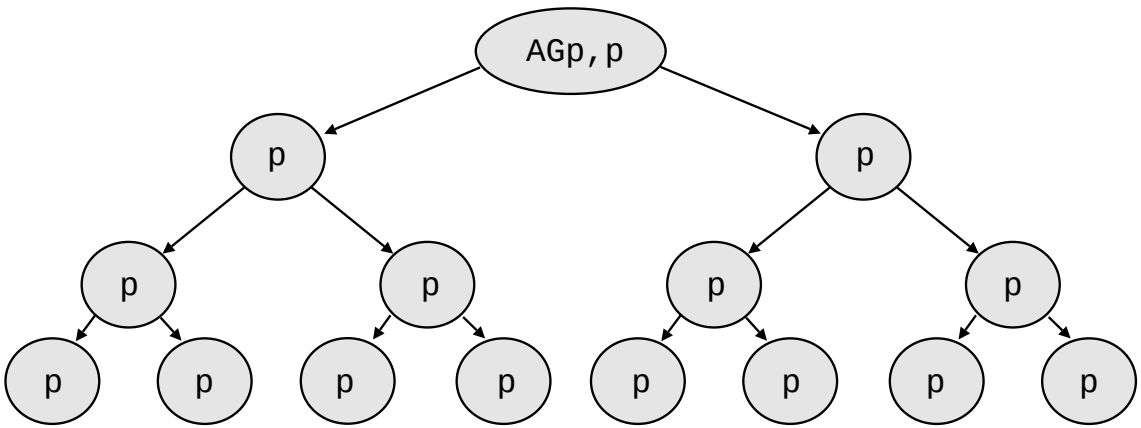
EG*p* : il existe un chemin avec *p* toujours vrai



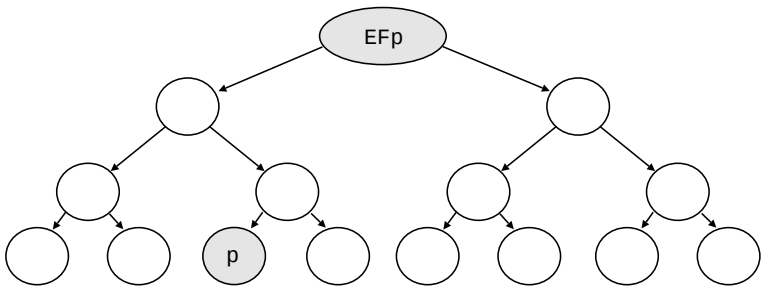
Quantificateurs de chemin II

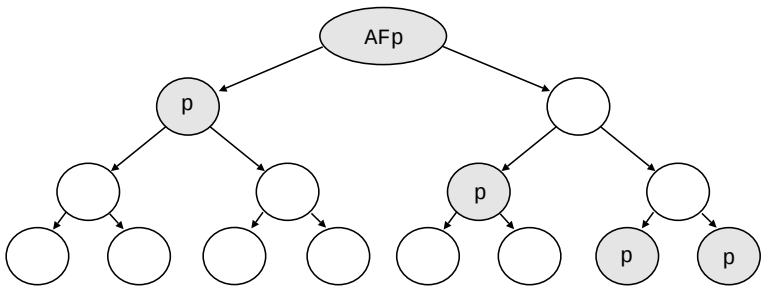
quantificateurs de chemins + connecteurs temporels

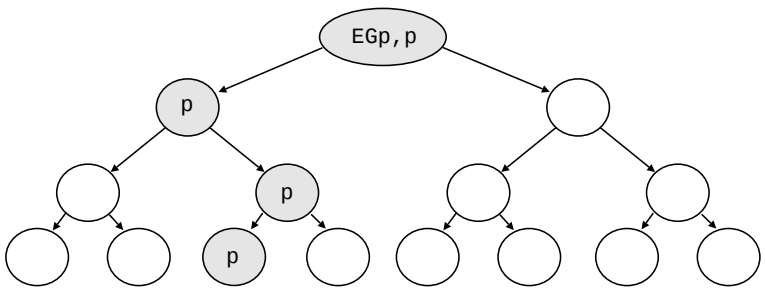
AGp : p est toujours vrai

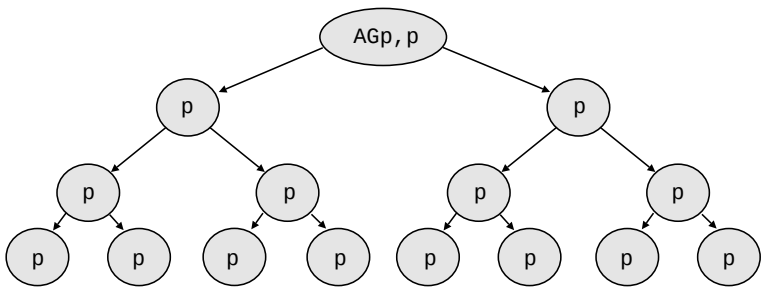


Quantificateurs de chemin III









Quelques exemples

Sur un modèle complet :

- 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
- Autres facteurs moins formels
  - apprentissage, compréhension, adéquation besoins réels, ...

Plan

Plan

Structure de Kripke

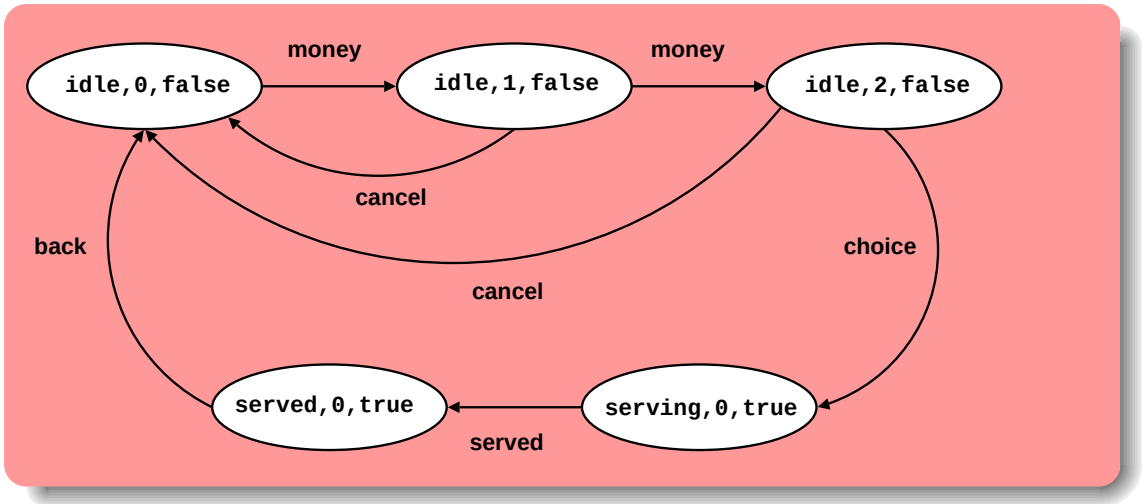
Pour le model checking on va considérer des structures de Kripke $\mathcal{M}$ plutôt que des systèmes de transitions S

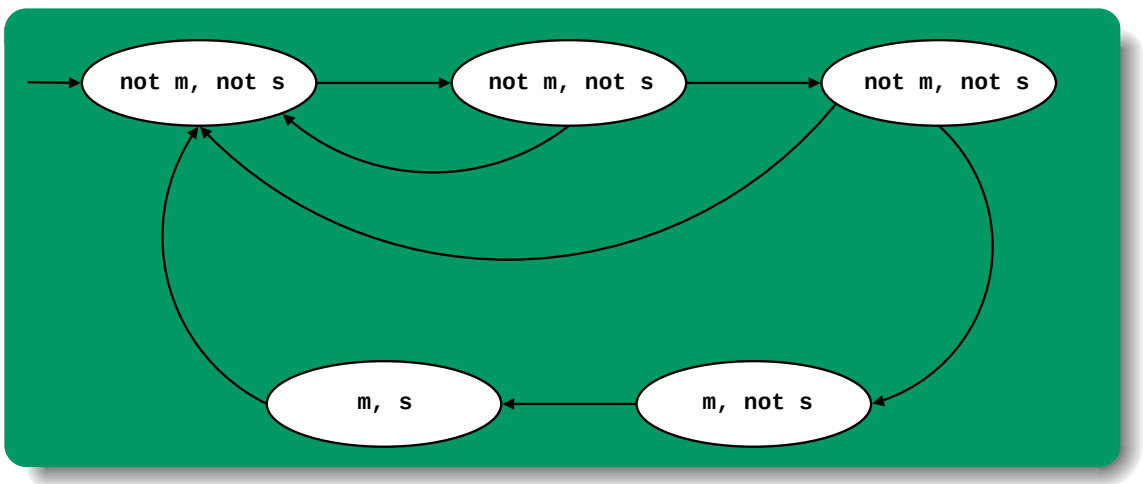
Schématiquement, $\mathcal{M}$ est obtenu de S en

- oubliant les étiquettes des arcs
- en ajoutant à un état les propriétés qu'il vérifie
- en ajoutant un état initial

Dualité propriétés sur états / transitions

Structure de Kripke II



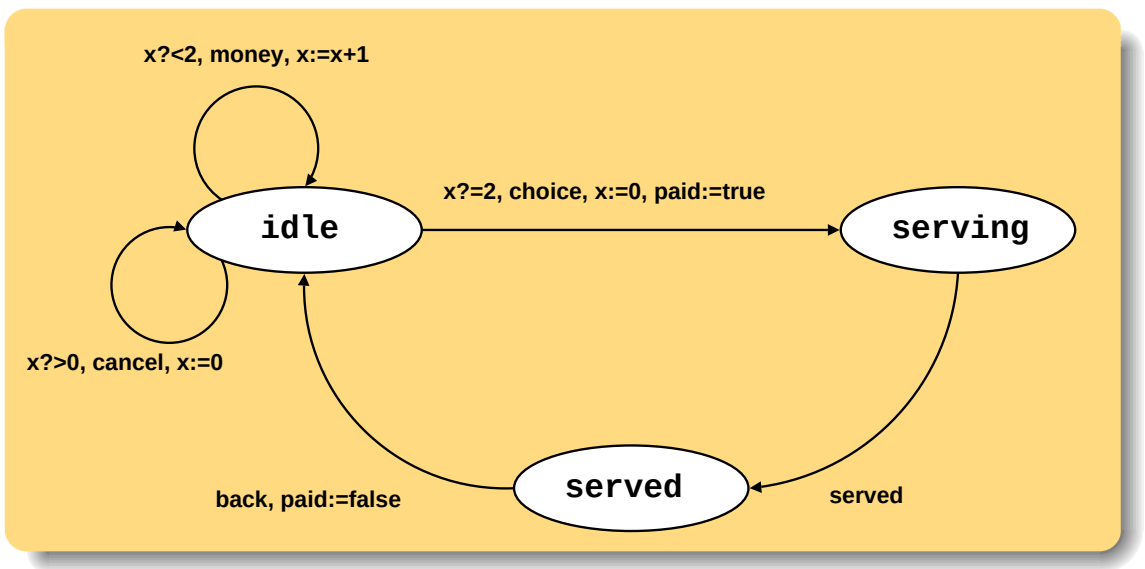


Récapitulatif

- Système réactif = système de départ, monde réel
- Machine à état P = syntaxe du modèle
- Système de transition S = sémantique de P
- Structure de Kripke $\mathcal{M}$ = adaptation de S pour MC

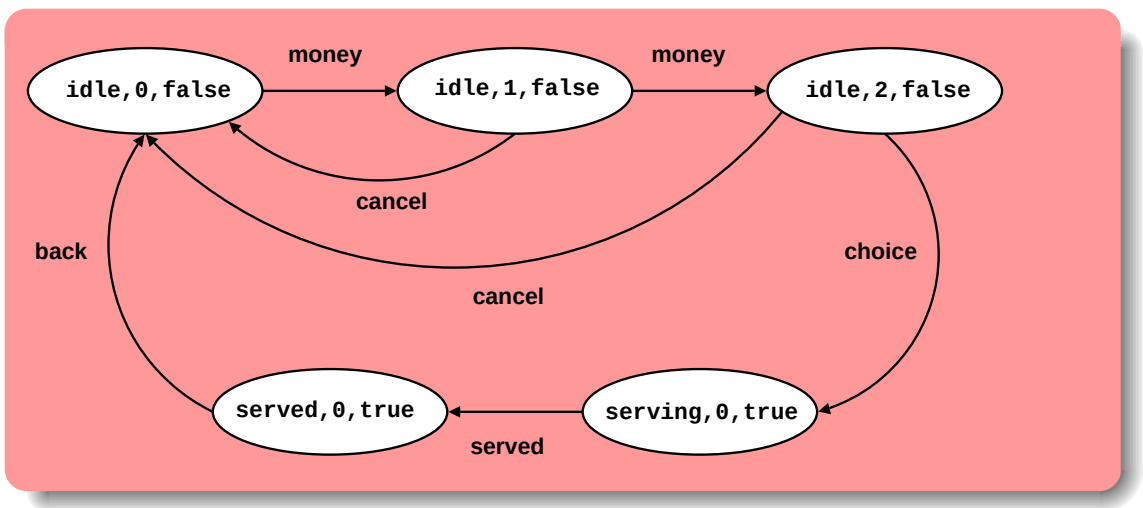
Récapitulatif

- Système réactif = système de départ, monde réel
- Machine à état P = syntaxe du modèle
- Système de transition S = sémantique de P
- Structure de Kripke $\mathcal{M}$ = adaptation de S pour MC



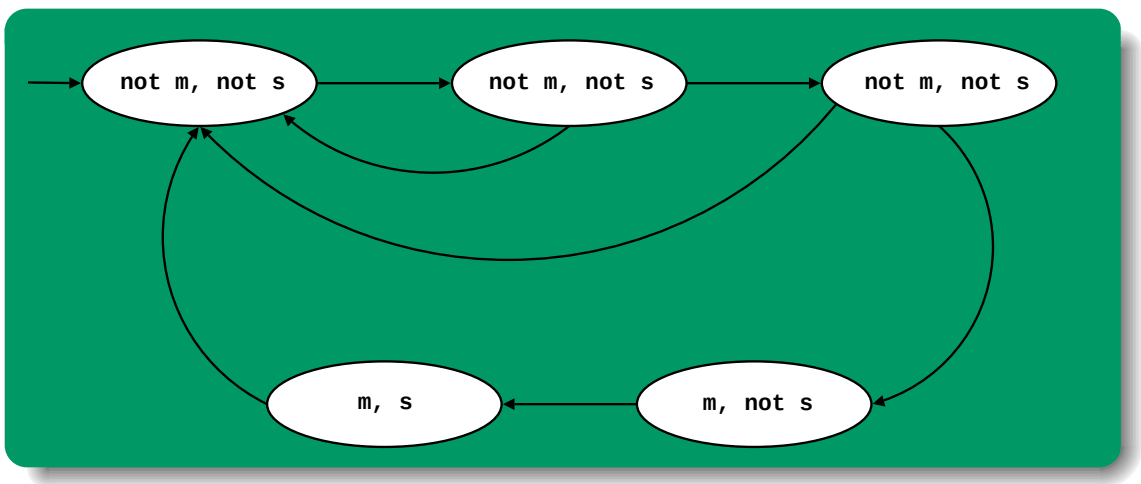
Récapitulatif

- Système réactif = système de départ, monde réel
- Machine à état P = syntaxe du modèle
- Système de transition S = sémantique de P
- Structure de Kripke $\mathcal{M}$ = adaptation de S pour MC



Récapitulatif

- Système réactif = système de départ, monde réel
- Machine à état P = syntaxe du modèle
- Système de transition S = sémantique de P
- Structure de Kripke $\mathcal{M}$ = adaptation de S pour MC



Plan

Plan

Introduction

Kripke

Bases de logique

LTL

CTL*

CTL

Comparaison

En bref

La logique

Philosophie : Logique = étude du raisonnement

Nos besoins sont plus pragmatiques

une logique = un langage de description adapté à décrire certaines propriétés et qui permet automatisation.

Exemples de logiques

- classique (maths)
- intuitioniste (programmation)
- multi-valuée, floue (incertitude)
- connaissance (IA)
- temporelle, temporisée (systèmes réactifs)

Définitions

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
- $E \subseteq D$ est $\mathcal{L}$-définissable si il existe $\varphi \in \mathcal{L}$ tq $E = [\![\varphi]\!]$

Problèmes standards de logique

Sur une formule φ

- Model checking. Est-ce que $\mathcal{I} \models \varphi$?
- Satisfaction. Est-ce que φ est satisfaisable ?
- Validité. Est-ce que φ est valide ?
- Synthèse. Donner $\mathcal{I}$ tel que $\mathcal{I} \models \varphi$.

Sur des logiques $\mathcal{L}_1$ et $\mathcal{L}_2$

- Expressivité. Est-ce que $\mathcal{L}_1$ et $\mathcal{L}_2$ définissent les mêmes ensembles ?
- Concision. Est-ce que $\mathcal{L}_1$ et $\mathcal{L}_2$ expriment les mêmes ensembles avec des formules de tailles semblables ?

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

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

Plan

Plan

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

- $\mathbf{AFG}p$ dans LTL
- $\mathbf{EF}p$ 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 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 $\mathcal{M}, s_0 \models_s \varphi$

Plan

Plan

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

Plan

Plan

Introduction

Kripke

Bases de logique

LTL

CTL*

CTL

Comparaison

En bref

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

Spécificités de CTL

On peut donner à CTL une sémantique uniquement en terme d'états de $\mathcal{M}$

- on peut ignorer les chemins
- algorithmes très spécifiques mais très efficaces *(leçon 3)*

CTL n'exprime pas l'équité

- on peut adapter en modifiant la sémantique de $\mathcal{M}$ *(leçon 3)*

Plan

Plan

Introduction

Kripke

Bases de logique

LTL

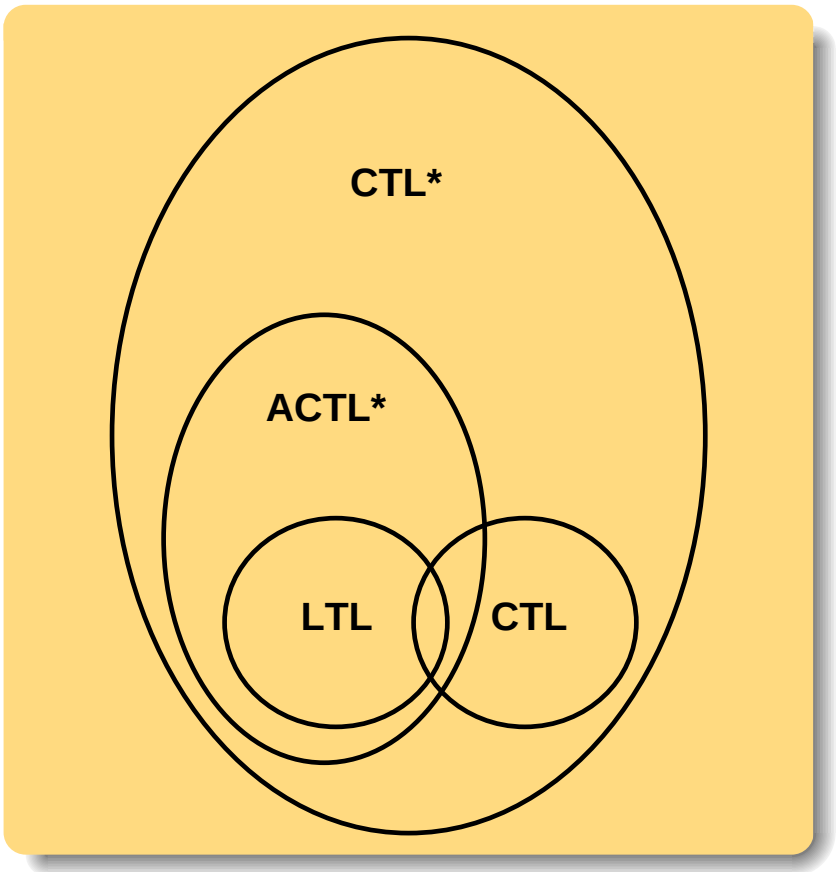
CTL*

CTL

Comparaison

En bref

Expressivité



Complexité

	CTL	LTL	CTL*
MC	PTIME	PSPACE	PSPACE
open MC	EXPTIME	PSPACE	2-EXPTIME
SAT	EXPTIME	PSPACE	2-EXPTIME

les problèmes sont complets pour les classes de complexité

MC Est-ce que $\mathcal{M} \models \varphi$?

open MC $\mathcal{M}, E \models \varphi$ pour tout E tq $E \models \psi$
avec ψ contrainte d'environnement

SAT Est-ce qu'il existe $\mathcal{M}$ tel que $\mathcal{M} \models \varphi$?

En pratique

LTL

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

CTL*

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

CTL

- le model checking est très efficace
- 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
- très coûteux si environnement

En pratique II

Industrie des processeurs s'oriente vers les logiques linéaires

- intuitif pour contre-exemples et interfaces
- permet la vérification au run-time par tests
- correspond globalement aux propriétés à spécifier
- LTL étendu avec des expressions régulières

Mais certaines propriétés branchantes sont utiles, et bonne complexité de CTL

Vers une utilisation conjointe de LTL et CTL

- CTL pour validation du modèle
- CTL pour φ simples sur tout le système
- LTL pour φ compliquées, sur une partie du système

Plan

Plan

Équivalences comportementales

Voir les exercices

Propriété très importante en pratique

- permet d'abstraire un composant par son interface
- suit le point de vue et les besoins des concepteurs systèmes
- base de techniques optimisées de vérification (modularité, abstraction)

Différentes notions pour différentes logiques

- équivalence de langages pour les logiques linéaires
- bisimulation pour les logiques branchantes

La bisimulation à un système donné est exprimable en CTL

Autres logiques

Voir les exercices

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

- CTL+, ACTL*, LTL + expressions régulières
- ajout de l'équité à CTL : fair CTL, ECTL, ECTL+, FCTL

D'autres formalismes

- automates
- HMSC
- μ-calcul

Exprimer plus que la simple causalité

- logiques temporisées
- logiques temporelles probabilistes
- logiques temporelles comptantes