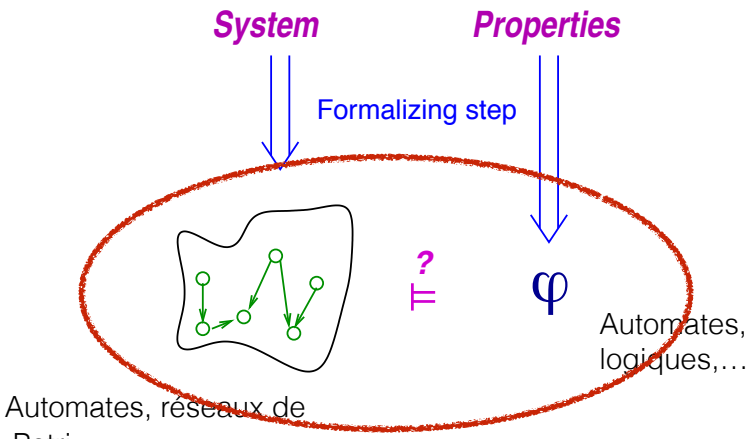
Méthodes formelles

Année 2018-2019
Master MIC

F. Laroussinie & A. Sangnier

Page web du cours:
https://www.irif.fr/~francoisl/m2mic.html

Model-checking



des objets précis !

Plan de la première partie

- Modèles
- Linear-time Temporal Logic (LTL)
  - syntaxe et sémantique
  - exemple de spécification
  - procédures de décision
- Computation Tree Logic (CTL)
  - syntaxe et sémantique
  - exemple de propriétés
  - procédures de décision
- Expressivité comparée
  - pouvoir de distinction, pouvoir d'expression, lien avec les équivalences comportementales,…
- Algorithmique du model-checking
  - méthodes symboliques, SAT-based MC.

+ Utilisation de l'outil PRISM (http://www.prismmodelchecker.org)

Modèles

Modéliser et spécifier un système réactif

système réactif = système qui interagit avec un environnement.

- Il ne calcule pas un résultat en un temps fini.
- Il maintient une interaction avec son environnement.
- Sa correction se base sur l'ordre des actions/évènements tout au long de son exécution.

Quels modèles ?

Soit AP un ensemble de propositions atomiques.

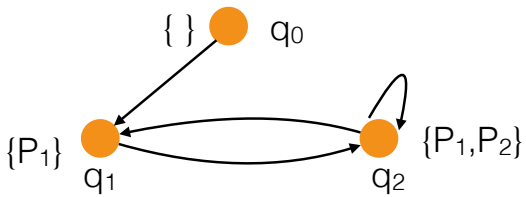
Définition:
Une structure de Kripke est un quadruplet $\mathbf{S}=(S,R,\ell,q_0)$ avec:
- S est un ensemble (fini) d'états,
- $R \subseteq S \times S$ est la relation de transition (hyp.: R totale),
- $\ell : S \rightarrow 2^{AP}$ est un étiquetage des états par des sous-ensemble de propositions atomiques,
- q_0 est un état initial.

Exécutions:
séquence infinie $\rho = s_0\, s_1\, s_2\, s_3 \ldots$ telle que:
- $s_i \in S \quad \forall\, i = 0,1,2,\ldots$
- $(s_i,s_{i+1}) \in R$ (notation: $s_i \rightarrow_{\mathbf{S}} s_{i+1}$ ou $s_i \rightarrow s_{i+1}$)

Notation: Exec(s) = ensemble des exécutions issues de s.

Exemple

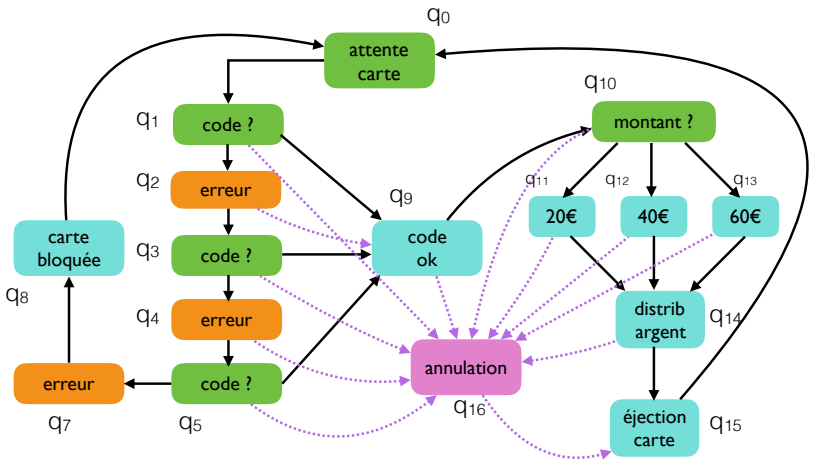
AP = {P_1, P_2}



Exec(q_1)= { ($q_1q_2q_1q_2$...,…, $q_1q_2q_2q_2$…, …}
ie Exec(q_1) = $(q_1q_2^+)^+q_2^\omega$ ∪ $(q_1q_2^+)^\omega$

Exemple 2

$\mathbf{S} = (Q,\rightarrow,\ell,q_0)$
AP={att. carte, code?, c. bloquée, code ok, …}



Exemple 3 algorithme d'exclusion mutuelle

```
boolean D1:= False
boolean D2 := False
```

```
Processus P1:
loop forever:
p1: Section NC
p2: D1 := True
p3: await (not D2)
p4: section critique
p5: D1 := False
```

```
Processus P2:
loop forever:
p1: Section NC
p2: D2 := True
p3: await (not D1)
p4: section critique
p5: D2 := False
```

+ hypothèse d'atomicité

→ un STE

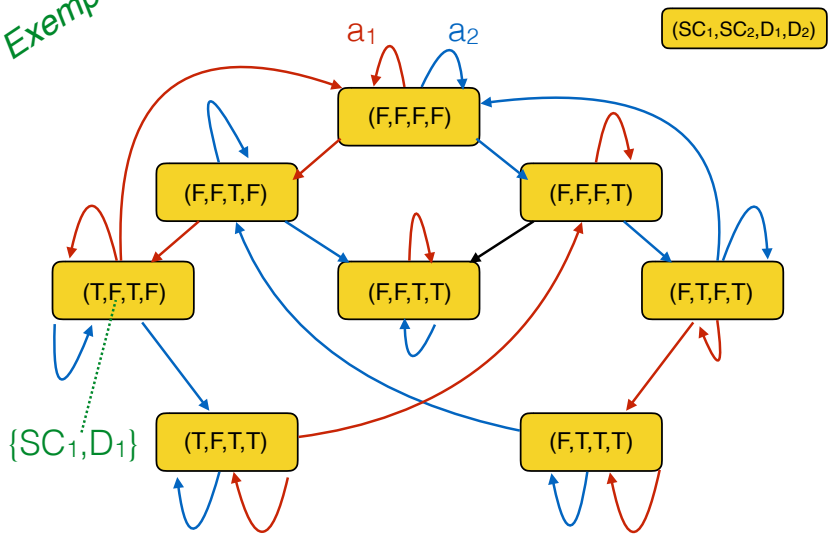
Exemple 3 algorithme d'exclusion mutuelle

→ une structure de Kripke avec AP={D_1,D_2,SC_1,SC_2}

- D_1 étiquette les états où la variable D1 est vraie,
- D_2 étiquette les états où la variable D2 est vraie,
- SC_1 étiquette les états où le processus 1 est en section critique,
- SC_2 étiquette les états où processus 2 est en section critique.

Et Q contient 8 états.

Exemple 3 algorithme d'exclusion mutuelle



Un modèle… et ensuite ?

Des propriétés !

algorithme d'exclusion mutuelle
propriétés attendues

- Exclusion mutuelle: Jamais les deux processus ne peuvent se trouver en SC au même moment.
- Il n'y a jamais pas de blocage.
- Absence de famine: Si un processus demande l'accès à la SC, il y arrivera un jour.
- Attente bornée: Si un processus demande l'accès à la SC, l'autre processus ne peut pas passer avant lui plus d'une fois.

En général, on vérifie ces propriétés sous hypothèses d'équité entre processus et en supposant que chaque section critique se termine.

Comment énoncer précisément ce genre de propriétés ?

- Exclusion mutuelle: Jamais les deux processus ne peuvent se trouver en SC au même moment.
- Il n'y a jamais de blocage.
- Absence de famine: Si un processus demande l'accès à la SC, il y arrivera un jour.
- Attente bornée: Si un processus demande l'accès à la SC, l'autre processus ne peut pas passer avant lui plus d'une fois.

Spécifier un système
exemple

Un distributeur de billets.

Un distributeur de billets « correct » ?

- Si le bon code est donné, on peut choisir une somme et obtenir de l'argent.
- Si on fait trois erreurs de code, la carte est bloquée.
- Après avoir obtenu l'argent, la carte est éjectée.
- A tout moment, si on appuie sur annulation, la carte est éjectée sans donner d'argent. sauf si 3 erreurs ont été commises… par le même utilisateur !
- Si on demande un ticket, un ticket est donné.
- On éjecte la carte avant de donner l'argent
- …

Spécifier un distributeur de billets

- Si le bon code est donné, on peut choisir une somme et on obtiendra de l'argent.
- Après avoir fait trois erreurs de code, la carte est bloquée.
- Après avoir obtenu l'argent, la carte est éjectée.
- A tout moment, si on appuie sur annulation, la carte est éjectée sans donner d'argent sauf si l'utilisateur a fait trois erreurs avant.

Spécifier un système réactif

Propriété de sureté (safety):
« Une mauvaise chose ne peut pas arriver ».
Ex: Il y a au plus un processus en section critique.

Propriétés de vivacité (liveness):
« Une bonne chose arrive (un jour) ».
Ex: Chaque demande d'accès à la SC est satisfaite un jour.

Propriétés d'équité (fairness):
→ Vérification d'exécutions équitables.
Ex: Chaque processus fait des actions « régulièrement ».

Spécifier un **système réactif**

- Sa correction se base sur l'ordre des actions/ évènements tout au long de son exécution.

avant/après/jusqu'à/depuis
+
si alors (⇒), et (∧), ou (∨)

→ la logique temporelle

Logiques temporelles

Les logiques temporelles étendent la logique propositionnelle avec:
- des « modalités temporelles » (ou des « opérateurs temporels »)
- sont interprétées dans des modèles munis d'une notion de temps.

Il y a de nombreuses logiques temporelles !

- LTL « Linear-time Temporal Logic »
- CTL « Computation Tree Logic »

Temporal logics

TLs are a good formalism for specifying properties of reactive systems.

[Pnueli 1977]

Cela permet d'énoncer des propriétés sur l'ordre des événements le long d'une exécution.

- sémantique naturelle,
- bonne expressivité,
- concis,
- procédures de décision efficaces (des outils !),
- beaucoup d'extensions (temporisés, probabilistes, pour les jeux, données,…).

Problèmes de vérification

Model-checking:
input: un modèle $\mathbf{S}$ et une formule φ
output: oui ssi $\mathbf{S} \models \varphi$.

Satisfaisabilité:
input: une formule φ
output: oui ssi il existe un modèle vérifiant φ.
(+ un modèle si la réponse est « oui »…)

Synthèse de contrôleur:
input: un modèle $\mathbf{S}$ et une formule φ
output: un contrôleur $\boldsymbol{C}$ t.q. $\mathbf{S} \times \boldsymbol{C} \models \varphi$.

Linear-time Temporal Logic

LTL « Linear-time Temporal Logic »

Logique de temps linéaire.
Le comportement d'un système est vu comme l'ensemble de ses exécutions prises séparément.

La notion de temps (avant/après/jusqu'à/depuis) s'interprète donc le long d'une exécution d'un STE.

LTL

Syntaxe: $P \in AP$

$$\phi,\psi ::= P \mid \neg\phi \mid \phi \vee \psi \mid \mathbf{X}\,\phi \mid \phi\ \mathbf{U}\ \psi$$

$\mathbf{X}\,\phi$: demain ϕ

$\phi\ \mathbf{U}\ \psi$: un jour ψ et avant ϕ : « ϕ jusqu'à ψ »

+Abbreviations: $\top$, $\bot$, $\wedge$,
$\mathbf{F}\varphi = \top\ \mathbf{U}\ \varphi$: "un jour dans le futur",
$\mathbf{G}\,\varphi = \neg\,\mathbf{F}\,\neg\varphi$: "toujours dans le futur"

LTL - sémantique

→ on interprète les formules de LTL sur une exécution d'une structure de Kripke:

+ℓ pour les val. des AP.

ρ^i est le i-ème suffixe: $s_i\ s_{i+1}\ldots$

LTL - sémantique

$\mathbf{S} = \langle S, R, \ell, q_0 \rangle$

$s_0 \in Q$, $\rho = s_0 s_1 s_2 s_3 s_4 \ldots$ une exécution de $\mathbf{S}$.

$\rho \models P$ ssi $P \in \ell(s_0)$

$\rho \models \varphi \vee \psi$ ssi ($\rho \models \varphi$ ou $\rho \models \psi$)

$\rho \models \neg\varphi$ ssi ($\rho \not\models \varphi$)

$\rho \models \mathbf{X}\,\varphi$ ssi $\rho^1 \models \varphi$ (avec $\rho^1 = s_1 s_2 s_3 \ldots$)

$\rho \models \varphi\ \mathbf{U}\ \psi$ ssi $\exists\, i \geq 0$ ($\rho^i \models \psi$ et ($\forall\ 0 \leq j < i$: $\rho^j \models \varphi$))

LTL

$\mathbf{S} = (S, R, \ell, q_0)$

$\rho \models \phi$ est désormais défini !

Et $\mathbf{S} \models \phi$?

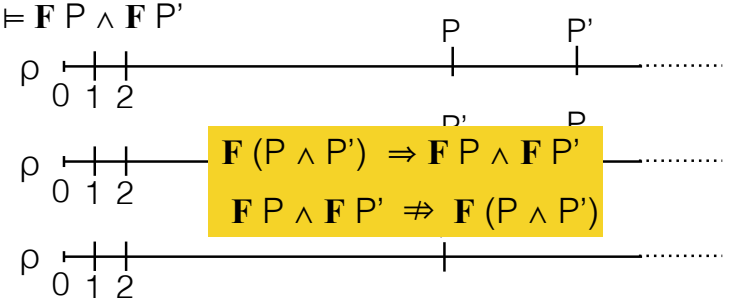
Rappel:
Avec les logiques de temps linéaire, le comportement d'un système est vu comme l'ensemble de ses exécutions prises séparément.

$\mathbf{S} \models \phi$ si et seulement si $\rho \models \phi$ $\forall\ \rho \in$ Exec(q_0)

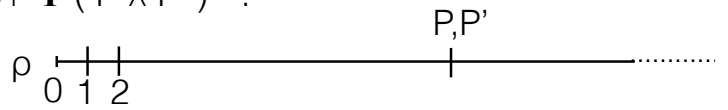
Exemples de formules

Comparer $\mathbf{F}\ P \wedge \mathbf{F}\ P'$ et $\mathbf{F}\ (P \wedge P')$

$\rho \models \mathbf{F}\ P \wedge \mathbf{F}\ P'$

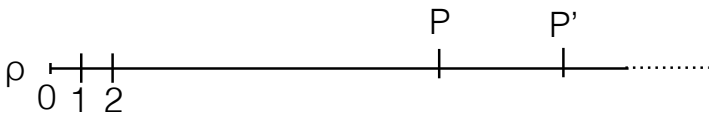


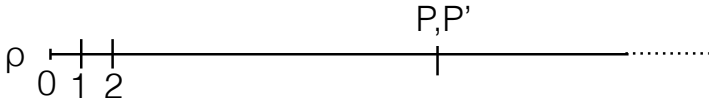
$\rho \models \mathbf{F}\ (P \wedge P')$?



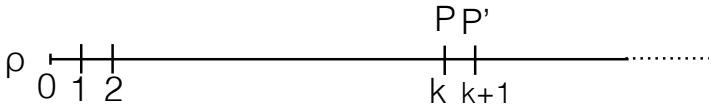
Exemples de formules

$\rho \models \mathbf{F}\,(P \wedge \mathbf{F}\,P')$?



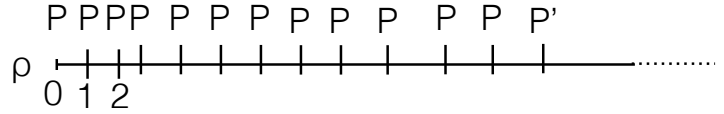


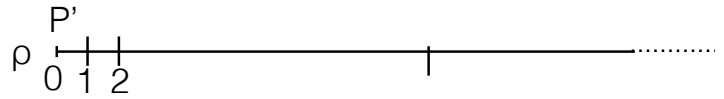
$\rho \models \mathbf{F}\,(P \wedge \mathbf{X}\,P')$?



Exemples de formules

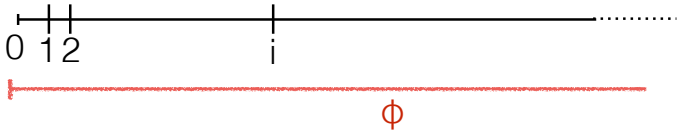
$\rho \models P\ \mathbf{U}\ P'$?





Exemples de formules

$\rho \models \neg\,\mathbf{F}\,\neg\phi$??



→ ϕ est vrai pour tous les états 0,1, 2,…
« toujours dans le futur »
On le note: **G** ϕ = ¬ **F** ¬ϕ

Exemples de formules

$\rho \models \neg\,\mathbf{X}\,\phi$

$\Leftrightarrow\ \rho \models \mathbf{X}\,\neg\,\phi$ (car exécutions infinies)

(sans cette hypothèse, on aurait:
$\rho \models \mathbf{X}\,\neg\,\phi \ \Rightarrow\ \rho \models \neg\,\mathbf{X}\,\phi$)

Exemples de formules

G (problème ⇒ **F** alarme)

G (request ⇒ **F** service)

G (¬ bug)

Exemples de formules

G F accueil

F G ok

GF request ⇒ **GF** service

GF (a ∧ b) implique **GF** a ∧ **GF** b

GF a ∧ **GF** b n'implique pas **GF** (a ∧ b)

Exclusion mutuelle (suite)

- Exclusion mutuelle: Jamais les deux processus ne peuvent se trouver en SC au même moment.

  $\neg\, \mathbf{F}\, (SC_1 \wedge SC_2)$ $\quad\quad$ $\mathbf{G}\, (\neg\, SC_1 \vee \neg\, SC_2)$

- Il n'y a jamais pas de blocage.

  $\mathbf{G}\, (\mathbf{X}\, \top)$ (NB: toujours vrai si → est totale)

- Absence de famine: Si un processus demande l'accès à la SC, il y arrivera un jour.

  $\mathbf{G}\, (D_1 \Rightarrow \mathbf{F}\ SC_1) \wedge \mathbf{G}\, (D_2 \Rightarrow \mathbf{F}\ SC_2)$

Et encore de nouvelles modalités !

Weak until :

$\rho \vDash \psi\ \mathbf{W}\ \phi$ = ssi ($\forall\, k \geq 0$, $\rho,k \vDash \psi$ ou $\exists\, j \geq 0$. ($\rho,j \vDash \phi$ et $\forall\, 0 \leq k < j$ on a $\rho,k \vDash \psi$)

$\rho \vDash \psi\ \mathbf{W}\ \phi \quad \Leftrightarrow \quad \rho \vDash \mathbf{G}\ \psi \ \vee\ \psi\ \mathbf{U}\ \phi \qquad (\forall\, \rho)$

Notation: $\psi\ \mathbf{W}\ \phi \equiv \mathbf{G}\ \psi \ \vee\ \psi\ \mathbf{U}\ \phi$

Definition:

$\phi \equiv \psi$ ssi ($\forall\, \rho$, on a : $\rho \vDash \phi \Leftrightarrow \rho \vDash \psi$)