

Introduction au Model Checking

Maximilien Colange & Alexandre Duret-Lutz &
Etienne Renault

<http://www.lrde.epita.fr/~adl/ens/mc/2017/>

14 février 2017

Plan du Cours

- 13 fév. 2017 (ADL)
 - ▶ Introduction
 - ▶ Approche par automate du Model checking
 - ▶ LTL et Automate de Büchi (TP)
- 14 fév. 2017 (MC)
 - ▶ Approche par automate du Model checking
 - ▶ Construction d'un Model Checker (TP)
- 21 fév. 2017 (ADL)
 - ▶ QCM sur article à lire
 - ▶ Binary Decision Diagrams & SAT solvers
- 28 fév 20167 (MC)
 - ▶ QCM sur article à lire
 - ▶ Automates Temporisés
- 21 mars 2016 (ER)
 - ▶ Test de vacuité des automates de Büchi
 - ▶ Fiche de lecture à rendre
- 23 mars 2016 (ER)
 - ▶ Réduction des systèmes par ordre partiel

système un modèle d'ascenseur, de ses usagers, et du contrôleur de l'ascenseur

spécification une propriété que les comportements du système doivent vérifier

Le **model checker** vérifie (de façon **exhaustive** et **automatique**) que tous les comportement du système satisfont le modèle.

Approche par automate du model checking

High-level
model M

model checker

LTL
property φ

$M \models \varphi$
or counter-
example

Approche par automate du model checking

High-level
model M

State-space
generation

State-space
automaton
 A_M

Product
Automaton
 $A_M \otimes A_{\neg\varphi}$

Synchronized
product

$$\mathcal{L}(A_M \otimes A_{\neg\varphi}) = \mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$$

Emptiness check
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL
property φ

LTL
translation

Negated
property
automaton $A_{\neg\varphi}$

$M \models \varphi$
or counter-
example

Approche par automate du model checking

High-level
model M

State-space
generation

State-space
automaton
 A_M

Product
Automaton
 $A_M \otimes A_{\neg\varphi}$

Synchronized
product

$$\mathcal{L}(A_M \otimes A_{\neg\varphi}) = \mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$$

Emptiness check
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL
property φ

LTL
translation

Negated
property
automaton $A_{\neg\varphi}$

$M \models \varphi$
or counter-
example

Approche par automate du model checking

High-level
model M

On-the-fly generation
of state-space automaton
 A_M

Product
Automaton
 $A_M \otimes A_{\neg\varphi}$

Synchronized
product
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) =$
 $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$

Emptiness check
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL
property φ

LTL
translation

Negated
property
automaton $A_{\neg\varphi}$

$M \models \varphi$
or counter-
example

Approche par automate du model checking

High-level
model M

On-the-fly generation
of state-space automaton
 A_M

On-the-fly
synchronized product
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) =$
 $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$

Emptiness check
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL
property φ

LTL
translation

Negated
property
automaton $A_{\neg\varphi}$

$M \models \varphi$
or counter-
example

Approche par automate du model checking

High-level
model M

On-the-fly generation
of state-space automaton
 A_M

Custom Model Checker

SPOT

On-the-fly
synchronized product
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) =$
 $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$

Emptiness check
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL
property φ

LTL
translation

Negated
property
automaton $A_{\neg\varphi}$

$M \models \varphi$
or counter-
example

Logique des propositions : l'instant présent

La logique propositionnelle peut caractériser **un** instant.

r : feu rouge allumé

o : feu orange allumé

v : feu vert allumé

$$r \wedge o \wedge v = \begin{array}{|c|} \hline \text{rouge} \\ \hline \text{orange} \\ \hline \text{vert} \\ \hline \end{array}, r \wedge \neg o \wedge \neg v = \begin{array}{|c|} \hline \text{rouge} \\ \hline \text{noir} \\ \hline \text{noir} \\ \hline \end{array}, \neg r \wedge \neg o \wedge v = \begin{array}{|c|} \hline \text{noir} \\ \hline \text{noir} \\ \hline \text{vert} \\ \hline \end{array}, \neg r \wedge \neg o \wedge \neg v = \begin{array}{|c|} \hline \text{rouge} \\ \hline \text{orange} \\ \hline \text{noir} \\ \hline \end{array}.$$

Comment dire que  précède  ?

Comment dire que le système ne reste pas toujours sur  ?

⇒ besoin de faire apparaître le temps

F1S : Logique monadique du 1^{er} ordre à un succ.

Les prop. deviennent des prédicats unaires, paramétrés par le temps.

$r(t), o(t), v(t)$: feux allumés à l'instant t

$t + 1$: instant successeur immédiat

$t \leq u$: ordre total sur les instants

$\exists t, \forall t$: quantificateurs du premier ordre

$\neg \forall t. (r(t) \wedge \neg o(t) \wedge \neg v(t))$: le système ne reste pas toujours .

$\forall t. ((\neg r(t) \wedge o(t) \wedge \neg v(t)) \rightarrow (r(t+1) \wedge \neg o(t+1) \wedge \neg v(t+1)))$:
toute configuration  est immédiatement suivie de .

$\forall t. \exists u. (t \leq u) \wedge (\neg r(u) \wedge \neg o(u) \wedge v(u))$:
le système passe infiniment souvent par la configuration .

S1S : Logique monadique du 2nd ordre à un succ.

$r(t), o(t), v(t)$: feux allumés à l'instant t

0 : instant initial

$t + 1$: instant successeur immédiat

$t \leq u$: ordre total sur les instants

$\exists t, \forall t$: quantificateurs du premier ordre

$\exists^2 X, \forall^2 X$: quantificateurs du second ordre

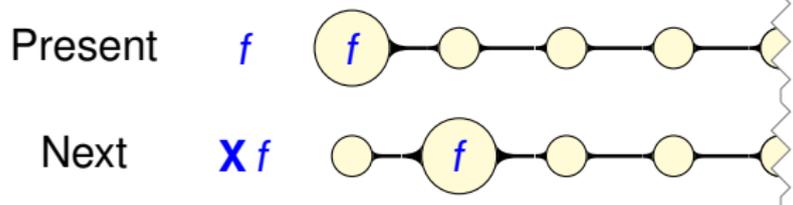
$t \in X$: appartenance d'une variable du premier ordre a une variable du second

$\exists^2 X. \underbrace{(0 \in X \wedge (\forall t. (t \in X \rightarrow (\neg(t + 1 \in X) \wedge (t + 1 + 1 \in X))))))}_{Pair(X)}$

$\exists^2 X. Pair(X) \wedge \forall t. (t \in X \rightarrow r(t))$: le feu rouge doit toujours être allumé aux instants pairs.

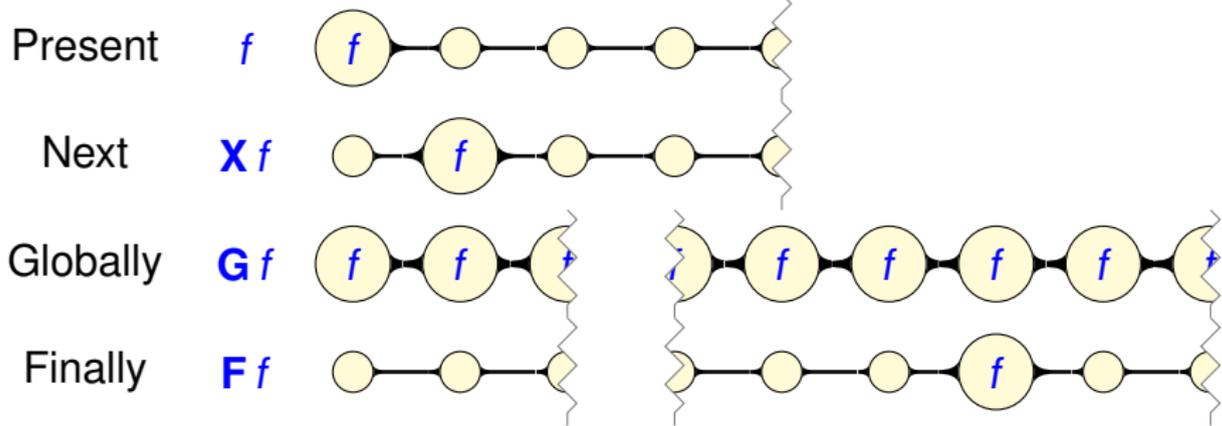
LTL : Logique Temporelle à temps Linéaire

Pour f et g deux formules propositionnelles :



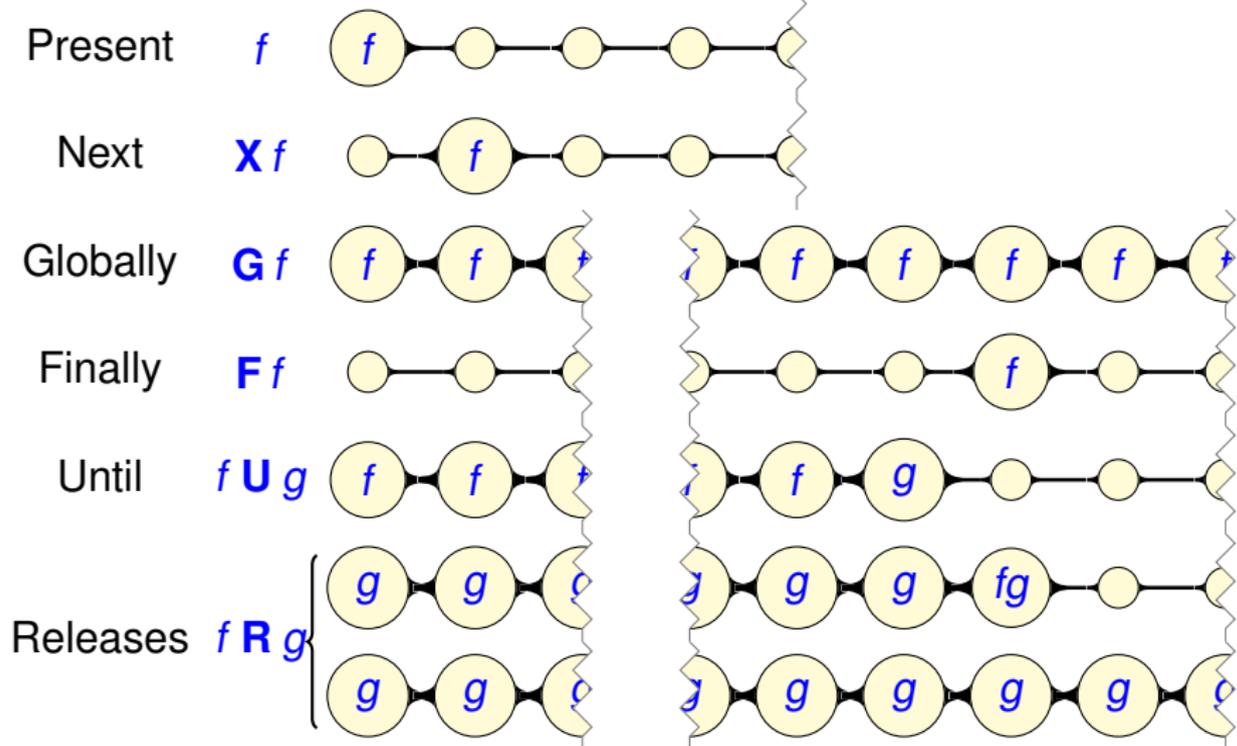
LTL : Logique Temporelle à temps Linéaire

Pour f et g deux formules propositionnelles :



LTL : Logique Temporelle à temps Linéaire

Pour f et g deux formules propositionnelles :



LTL : Logique Temporelle à temps Linéaire

Next	X f	f est vraie à l'instant suivant
Always	G f	f est vraie a tout instant
Eventually	F f	f sera vraie à un instant (présent ou futur)
Until	f U g	f est toujours vraie jusqu'à ce que g le soit

Équivalente à la logique monadique du premier ordre à un successeur.

$\neg \mathbf{G}(r \wedge \neg o \wedge \neg v)$: le système ne reste pas tout le temps .

$\mathbf{G}((\neg r \wedge o \wedge \neg v) \rightarrow \mathbf{X}(r \wedge \neg o \wedge \neg v))$:  est tjs imm. suivi de .

$\mathbf{GF}(\neg r \wedge \neg o \wedge v)$: le système passe infiniment souvent par .

LTL : Logique Temporelle à temps Linéaire

Next	X f	f est vraie à l'instant suivant
Always	G f	f est vraie a tout instant
Eventually	F f	f sera vraie à un instant (présent ou futur)
Until	f U g	f est toujours vraie jusqu'à ce que g le soit

F, **G** et **R** (Release) peuvent être vus comme du sucre :

$$\mathbf{F} f = \top \mathbf{U} f$$

$$f \mathbf{R} g = \neg(\neg f \mathbf{U} \neg g)$$

$$\mathbf{G} f = \neg \mathbf{F} \neg f = \neg(\top \mathbf{U} \neg f) = \perp \mathbf{R} f$$

D'autre part on a :

$$\neg \mathbf{X} f = \mathbf{X} \neg f$$

$$\neg \mathbf{F} f = \mathbf{G} \neg f$$

$$\neg \mathbf{G} f = \mathbf{F} \neg f$$

$$\neg(f \mathbf{U} g) = (\neg f) \mathbf{R} (\neg g)$$

$$\neg(f \mathbf{R} g) = (\neg f) \mathbf{U} (\neg g)$$

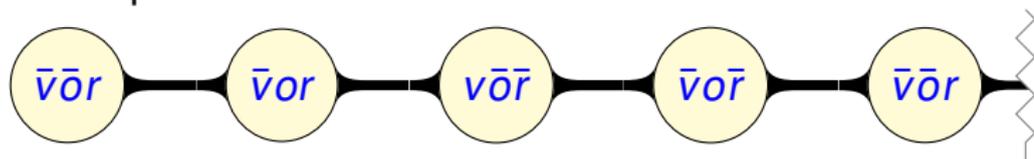
Vers une définition formelle de LTL

- ▶ Une formule LTL décrit une contrainte sur les scénarios du système.

Comment définir un scénario ?

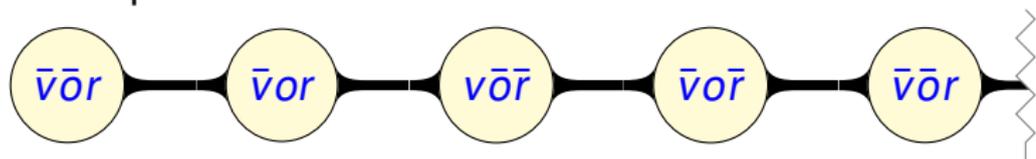
Vers une définition formelle de LTL

- ▶ Une formule LTL décrit une contrainte sur les scénarios du système.
Comment définir un scénario ?
- ▶ Un scénario est une séquence infinie dont chaque étape est étiquetée par les valuations de toutes les propositions atomiques.



Vers une définition formelle de LTL

- ▶ Une formule LTL décrit une contrainte sur les scénarios du système.
Comment définir un scénario ?
- ▶ Un scénario est une séquence infinie dont chaque étape est étiquetée par les valuations de toutes les propositions atomiques.



- ▶ On voit un scénario comme un ω -mot dont les lettres sont des sous-ensembles des propositions atomiques. La lettre (i.e., le sous-ensemble) donne les propositions vraies à cet instant.

$\{r\}; \{o, r\}; \{v\}; \{o\}; \{r\}; \dots$

ω : premier ordinal infini

Les entiers naturels peuvent être construits avec des ensembles :

$$0 = \{\}$$

$$1 = \{0\} = \{\{\}\}$$

$$2 = \{0, 1\} = \{\{\}, \{\{\}\}\}$$

$$3 = \{0, 1, 2\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}$$

⋮

Tout entier correspond à un ensemble.

L'inclusion sur les ensembles se traduit par un ordre sur les entiers.

ω : premier ordinal infini

Les entiers naturels peuvent être construits avec des ensembles :

$$0 = \{\}$$

$$1 = \{0\} = \{\{\}\}$$

$$2 = \{0, 1\} = \{\{\}, \{\{\}\}\}$$

$$3 = \{0, 1, 2\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}$$

⋮

$$n + 1 = n \cup \{n\}$$

⋮

Tout entier correspond à un ensemble.

L'inclusion sur les ensembles se traduit par un ordre sur les entiers.

ω : premier ordinal infini

Les entiers naturels peuvent être construits avec des ensembles :

$$0 = \{\}$$

$$1 = \{0\} = \{\{\}\}$$

$$2 = \{0, 1\} = \{\{\}, \{\{\}\}\}$$

$$3 = \{0, 1, 2\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}$$

\vdots

$$n + 1 = n \cup \{n\}$$

\vdots

$$= \mathbb{N}$$

Tout entier correspond à un ensemble.

L'inclusion sur les ensembles se traduit par un ordre sur les entiers.

ω : premier ordinal infini

Les entiers naturels peuvent être construits avec des ensembles :

$$0 = \{\}$$

$$1 = \{0\} = \{\{\}\}$$

$$2 = \{0, 1\} = \{\{\}, \{\{\}\}\}$$

$$3 = \{0, 1, 2\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}$$

\vdots

$$n + 1 = n \cup \{n\}$$

\vdots

$$\omega = \mathbb{N}$$

Tout entier correspond à un ensemble.

L'inclusion sur les ensembles se traduit par un ordre sur les entiers.

ω : premier ordinal infini

Les **ordinaux** peuvent être construits avec des ensembles :

$$0 = \{\}$$

$$1 = \{0\} = \{\{\}\}$$

$$2 = \{0, 1\} = \{\{\}, \{\{\}\}\}$$

$$3 = \{0, 1, 2\} = \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}$$

\vdots

$$n + 1 = n \cup \{n\}$$

\vdots

$$\omega = \mathbb{N}$$

$$\omega + 1 = \mathbb{N} \cup \{\omega\}$$

Tout **ordinal** correspond à un ensemble.

L'inclusion sur les ensembles se traduit par un ordre sur les **ordinaux**.

(Paradoxe : les ordinaux ne forment pas un ensemble.)

Notations sur les ω -mots

Soient Σ un alphabet et $n \in \mathbb{N} \cup \{\omega\}$ un ordinal.

Dans notre cas, l'alphabet est $\Sigma = 2^{AP}$ où $AP = \{v, o, g\}$

Une séquence de taille n (ou n -mot) de Σ est une fonction $\sigma: \llbracket 0, n \llbracket \mapsto \Sigma$ associant une lettre chaque entier naturel inférieur à n .

Notations :

Σ^n l'ensemble des séquences de taille n ,

Σ^\star l'ensemble des séquences finies ($n < \omega$),

Σ^ω l'ensemble des séquences infinies ($n = \omega$),

σ^i suffixe de σ commençant à la position i :
 $\sigma^i(j) = \sigma(i + j)$ pour les j tels que $i + j < n$,

LTL : Interprétation sur un ω -mot

Pour toute proposition atomique p_i et toutes formules LTL f_1 et f_2 , la satisfaction d'une formule LTL f par rapport à un mot $\sigma \in (2^{AP})^\omega$ est notée $\sigma \models f$ et définie inductivement par :

$$\sigma \models p \quad \text{ssi } p \in \sigma(0)$$

$$\sigma \models \neg f_1 \quad \text{ssi } \neg(\sigma \models f_1)$$

$$\sigma \models f_1 \wedge f_2 \quad \text{ssi } \sigma \models f_1 \text{ et } \sigma \models f_2$$

$$\sigma \models \mathbf{X} f_1 \quad \text{ssi } \sigma^1 \models f_1$$

$$\sigma \models f_1 \mathbf{U} f_2 \quad \text{ssi } \exists i \geq 0 \text{ tel que } \sigma^i \models f_2 \text{ et } \forall j \in \llbracket 0, i - 1 \rrbracket, \sigma^j \models f_1$$

Le langage de la formule φ est l'ensemble des séquences infinies sur 2^{AP} qui satisfont φ .

$$\mathcal{L}_{AP}(\varphi) = \{ \sigma \in (2^{AP})^\omega \mid \sigma \models \varphi \}$$

“Presque comme”(TM) les automates vus en THLR. Mais :

- ▶ reconnaissent des mots infinis : la notion d'état final n'a plus de sens et est remplacée par une **condition d'acceptation**.
- ▶ (dans notre cas) les lettres sont tirées de 2^{AP} , et souvent représentées par des formules booléennes sur AP

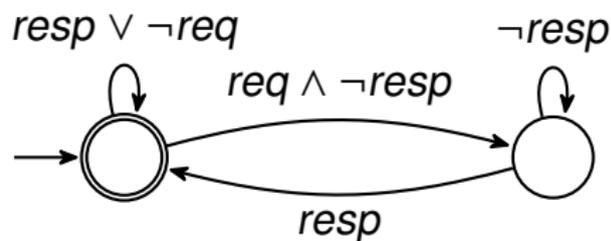
Différentes conditions d'acceptation existent, et portent des noms tels que Automate de Büchi, de Rabin, de Streett, de Müller, automates à parité, etc.

Automates de Büchi

Un mot est *accepté* par l'automate s'il visite **infiniment souvent** au moins un état acceptant

Notation traditionnelle :

$\mathbf{G}(req \rightarrow \mathbf{F} resp)$

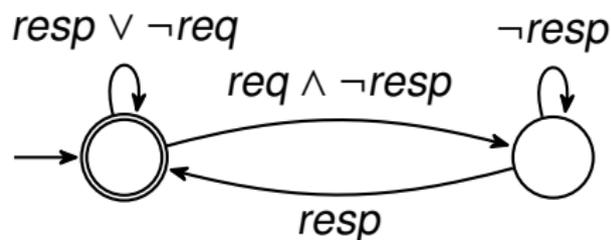


Automates de Büchi

Un mot est *accepté* par l'automate s'il visite **infiniment souvent** au moins un état acceptant

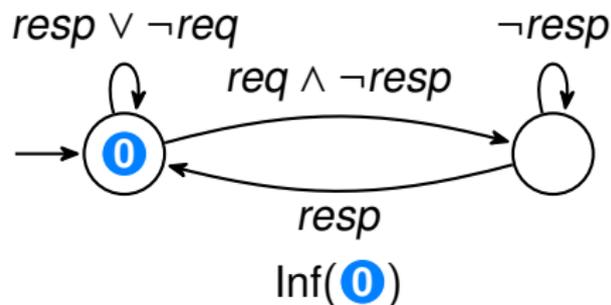
Notation traditionnelle :

$G(req \rightarrow F resp)$



Notation plus pratique pour la suite :

$G(req \rightarrow F resp)$



Automates de Büchi généralisés

Un mot est *accepté* par l'automate s'il visite **infiniment souvent** chaque ensemble d'acceptation. Les ensembles d'acceptation sont des ensembles d'états ou de transitions.

$GF a \wedge GF b$

