Logique du premier ordre
Logique temporelle et vérification

Richard Lassaigne
IMJ/Logique mathématique
CNRS-Université Paris Diderot

La logique propositionnelle ne permet de décrire que des constructions extrêmement simples du langage : les opérations booléennes sur les propositions.
Comment formaliser un fragment du raisonnement, comme par exemple :

- *certains étudiants assistent à tous les cours* ;
- *aucun étudiant n'assiste à un cours non intéressant* ;
- peut-on en conclure que *tous les cours sont intéressants* ?

Introduire la notion de **relation** unaire, binaire, etc. c'est se donner les moyens de dissocier un fait élémentaire sous la forme d'un objet possédant un attribut, ou d'une relation entre plusieurs objets, comme dans les énoncés :

- *tel cours est intéressant* ;
- *tel étudiant assiste au cours d'informatique* ;

Le second type de procédés du langage que **la logique du premier ordre** permet de représenter est la **quantification** sur les objets, comme dans l'énoncé : *tous les cours sont intéressants*.

L'**alphabet** d'un langage du premier ordre comporte d'abord les symboles suivants, qui sont communs à tous les langages :

- les connecteurs $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$,
- les parenthèses $(,)$,
- les **quantificateurs universels** $\forall$ et **existentiels** $\exists$,
- un ensemble infini V de symboles de **variables** , choisis habituellement parmi les lettres x, y, z, u, v éventuellement indicées.

Définition 1 *Un* **langage** $\mathcal{L}$ *de la logique du premier ordre est caractérisé par l'ensemble de symboles suivants :*

- *les symboles de* **relations** *(ou* **prédicats***) ; à chaque symbole est associé un entier strictement positif, appelé l'***arité***,*
- *les symboles de* **constantes***.*

Le langage $\mathcal{L}$ est supposé égalitaire.

Définition 2 *L'ensemble des* **formules atomiques** *est l'ensemble des formules de la forme :*

- $t_1 = t_2$ *où* t_1, t_2 *sont des termes,*
- $Rt_1t_2...t_n$ *où* R *est un symbole de relation d'arité* n *et* $t_1, t_2, ..., t_n$ *sont des termes ; cette formule peut aussi être notée* $R(t_1, t_2, ..., t_n)$.

L'ensemble des formules est défini par induction.

Définition 3 *L'ensemble des formules, que l'on désigne par* $\mathcal{F}$*, est le plus petit ensemble satisfaisant :*

- *toute formule atomique est une formule,*
- *si* F *est une formule, alors* $\neg F$ *est une formule,*
- *si* F, G *sont des formules, alors* $(F \wedge G)$, $(F \vee G)$,$(F \to G)$ *et* $(F \leftrightarrow G)$ *sont des formules,*
- *si* F *est une formule et* v *une variable, alors* $\forall v\ F$ *et* $\exists v\ F$ *sont des formules.*

Comme pour le calcul propositionnel, l'existence et l'unicité de la **décomposition** d'une formule, aussi compliquée soit-elle, sont assurées grâce à la définition par induction de l'ensemble des formules.

Proposition 1 *Toute formule d'un langage du premier ordre se décompose de manière unique sous l'une, et une seule, des formes suivantes :*

- *une formule atomique,*
- $\neg F$ *où* F *est une formule,*
- $(F \wedge G)$, $(F \vee G)$,$(F \to G)$ *ou* $(F \leftrightarrow G)$, *où* F, G *sont des formules,*
- $\forall v\ F$ *ou* $\exists v\ F$, *où* F *est une formule et* v *une variable.*

La décomposition complète d'une formule peut être obtenue récursivement : on fait apparaître toutes les formules entrant dans cette décomposition.

Une même variable peut apparaître en plusieurs endroits dans une formule. Il est nécessaire de distinguer ces différentes situations.

Définition 4 *Une* **occurrence** *d'une variable dans une formule est un couple constitué de cette variable et d'une place effective, c'est-à-dire qui ne suit pas un quantificateur.*

Dans la formule F : $(Rxz \to \forall z\ (Ryz \vee y = z))$, la variable x possède une occurrence, la variable y deux et la variable z trois.

Définition 5

- *Une occurrence d'une variable x dans une formule F est une* **occurrence libre** *si elle ne se trouve dans aucune sous-formule de F, qui commence par une quantification $\forall x$ ou $\exists x$. Dans le cas contraire, la variable x est une* **variable liée** *dans F.*
- *Une* **variable** *est* **libre** *dans une formule si elle a au moins une occurrence libre dans cette formule.*
- *Une* **formule close** *est une formule sans variable libre.*

La sémantique de la logique du premier ordre est définie en interprétant les formules d'un langage donné dans les structures correspondantes.

Exemple 1 *Un graphe G d'origine s, i.e. un ensemble D muni d'une relation binaire E et d'un sommet distingué s, est une structure pour le langage $\mathcal{L} = \{R, c\}$, notée $G = (D, E, s)$.*

La satisfaction d'une formule dans une structure est définie par **induction**. Par exemple :

- la formule atomique Rcx est satisfaite par le sommet $b \in D$ si b est un sommet relié à l'origine s,
- la formule $\forall x\ (x \neq c \to Rcx)$ est satisfaite dans le graphe G si l'origine est reliée (par une arête) à tous les autres sommets,
- la formule $\exists y \forall x\ (x \neq y \to Ryx)$ est satisfaite dans le graphe G s'il existe un sommet relié à tous les autres.

Règles d'introduction et d'élimination des quantificateurs :

- $\forall$ – introduction

$$\frac{\Gamma \vdash F}{\Gamma \vdash \forall x\ F}$$

  si x n'est pas libre dans Γ.

- $\forall$ – élimination

$$\frac{\Gamma \vdash \forall x\ F}{\Gamma \vdash F(t/x)}$$

  pour tout terme t du langage.

- $\exists$ – introduction

$$\frac{\Gamma \vdash F(t/x)}{\Gamma \vdash \exists x\ F}$$

  où t est un terme du langage.

- $\exists$ – elimination

$$\frac{\Gamma \vdash \exists x\ F \qquad \Gamma, F \vdash G}{\Gamma \vdash G}$$

  si x n'est libre ni dans Γ ni dans G.

Une théorie T est un ensemble de formules closes. Une formule close F est **conséquence** de la théorie T si tout modèle de T est modèle de F.

Proposition 2 *Une formule close F est conséquence de la théorie T si et seulement si la théorie $T \cup \{\neg F\}$ n'a pas de modèle.*

Si F est conséquence de T, alors par définition, tout modèle de T est un modèle de F, i.e. il n'existe pas de modèle de $T \cup \{\neg F\}$. L'inverse est clair.

Le théorème de **complétude** de la logique du premier ordre est dû à K. Gödel et sera admis ici sans démonstration. La signification de ce résultat fondamental est que les notions de formule prouvable et de formule valide, i.e. vraie dans tout modèle, sont équivalentes.

Théorème 1 *Pour toute théorie T et toute formule close F, F est prouvable à partir de T si et seulement si F est conséquence de T.*

Vérification **automatique** de propriétés temporelles sur des systèmes réactifs.

Système réactif : modélisé par un système de transition

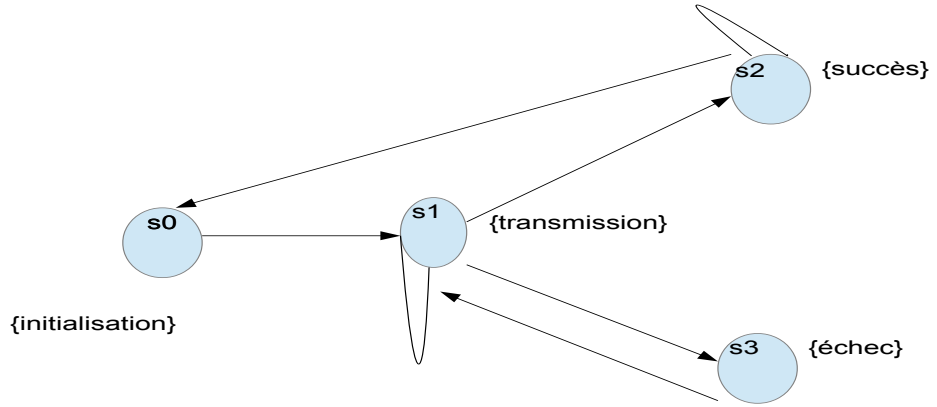
Système de transition $\mathcal{M} = \{S, R, s_0, l\}$:

- un ensemble d'états S,
- un état initial s_0,
- une relation de transition $R \subseteq S^2$,
- un ensemble de propositions atomiques AP,
- une fonction d'étiquetage $l : S \longrightarrow 2^{AP}$.

Exemples de propriétés **temporelles** :

- un état intéressant sera toujours atteint dans le futur (accessibilité) ;
- un état mauvais ne sera jamais atteint dans le futur (sureté) ;
- si un processus intéressant demande à être exécuté, alors le système finira par l'exécuter (vivacité).

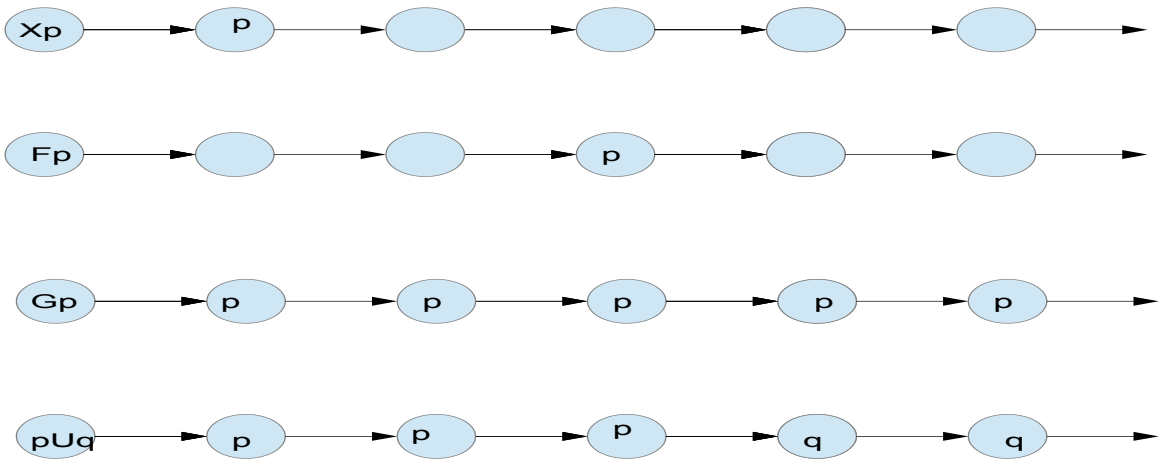
Exemple :



Propriétés **temporelles** :

- $Fsuccès$: *dans le futur, on atteindra un état de succès*,
- $X(transmission\ U succès)$: *on effectue une transmission jusqu' à atteindre un état de succès*.

Un **chemin** est une suite infinie $\sigma \in S^\omega$, notée $\sigma = s_0 s_1 \dots s_i \dots$ telle que pour tout i, $(s_i, s_{i+1}) \in R$.



On introduit 4 nouveaux connecteurs (temporels) :

- X (next) Xp signifie p *est vrai dans l'état suivant*,
- F (future) Fp signifie p *est vrai dans un état futur*,
- G (globally) Gp signifie p *est vrai dans tout état futur*,
- U (until) pUq signifie p *est vrai jusqu'à un état où* q *est vrai*.

L'ensemble des **formules de chemin** :

- contient l'ensemble AP des variables propositionnelles, ainsi qu'une formule $vrai$,
- est clos par application des connecteurs propositionnels,
- est clos par application des opérateurs temporels X et U,

Interprétation d'une formule φ sur un **chemin** σ :

- $(\mathcal{M}, \sigma) \models p$ ssi $p \in l(s_0)$,
- l'interprétation des connecteurs propositionnels est standard,
- $(\mathcal{M}, \sigma) \models X\psi$ ssi $(\mathcal{M}, \sigma^1) \models \psi$,
- $(\mathcal{M}, \sigma) \models \varphi_1 U \varphi_2$ ssi il existe $i \geq 0$ tel que $(\mathcal{M}, \sigma^i) \models \varphi_2$ et que pour tout $0 \leq j < i$, $(\mathcal{M}, \sigma^j) \models \varphi_1$.

Les formules $F\varphi$ et $G\varphi$ sont des abréviations pour les formules suivantes : $F\varphi \equiv vrai U \varphi$ et $G\varphi \equiv \neg F \neg \varphi$.

La formule φ est **satisfaite** par un système $\mathcal{M}$ si elle l'est par **tous** les chemins d'exécution d'origine s_0, ce qui est noté $\mathcal{M} \models \forall\ \varphi$.

Un **automate de Büchi** est un automate $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ défini par :

- un alphabet Σ fini, un ensemble d'états Q fini, un état initial $q_0 \in Q$,
- une fonction de transition $\delta : Q \times \Sigma \longrightarrow 2^Q$,
- un ensemble $F \subseteq Q$ d'états acceptants.

Exécution r de $\mathcal{A}$ sur un mot infini $w = a_0 \dots a_i \cdots \in \Sigma^\omega$: **suite infinie** d'états $q_0 \dots q_i \cdots \in Q^\omega$ telle que $q_{i+1} \in \delta(q_i, a_i)$ pour tout $i \geq 0$.

Définition 6

- *Une exécution r de $\mathcal{A}$ est acceptante s'il existe un état acceptant qui apparaît infiniment souvent dans r.*
- *Un mot w est accepté par l'automate $\mathcal{A}$ s'il existe une exécution r acceptante sur ce mot.*
- *Le langage $L(\mathcal{A})$ de l'automate $\mathcal{A}$ est l'ensemble des mots acceptés par l'automate.*

La méthode est la suivante :

1. Transformer le **système de transition** $\mathcal{M}$ en un **automate** $\mathcal{A}_{\mathcal{M}}$.
2. Construire l'automate $\mathcal{A}_{\neg\varphi}$ correspondant à la **négation** de la formule φ.
3. Calculer l'**automate** $\mathcal{B} = \mathcal{A}_{\mathcal{M}} \times \mathcal{A}_{\neg\varphi}$ **acceptant** le langage $L(\mathcal{A}_{\mathcal{M}}) \cap L(\mathcal{A}_{\neg\varphi})$.
4. Tester si le langage $L(\mathcal{A}_{\mathcal{M}} \times \mathcal{A}_{\neg\varphi})$ est **vide**.

Intuitivement, $L(\mathcal{A}_{\mathcal{M}})$ représente toutes les exécutions (infinies) du système $\mathcal{M}$ et $L(\mathcal{A}_{\neg\varphi})$ représente les exécutions ne satisfaisant pas la formule φ. Le langage $L(\mathcal{A}_{\mathcal{M}} \times \mathcal{A}_{\neg\varphi})$ est donc **vide** ssi tous les chemins de $\mathcal{M}$ **satisfont** la formule φ. Dans le cas où ce langage n'est pas vide, un mot lui appartenant est un témoin d'**erreur**.

La **complexité** de la méthode du model checking de LTL par automates est la suivante :

- la construction de l'automate $\mathcal{A}_{\mathcal{M}}$ s'effectue en $O(\mid \mathcal{M} \mid)$,
- la construction de l'automate $\mathcal{A}_{\neg\varphi}$ est en $O(2^{|\varphi|})$,
- la taille de l'automate produit $\mathcal{A}_{\mathcal{M}} \times \mathcal{A}_{\neg\varphi}$ est en $O(\mid \mathcal{M} \mid . \, 2^{|\varphi|})$,
- le test du langage vide est en $O(\mid \mathcal{M} \mid . \, 2^{|\varphi|})$.

La complexité en temps de cet algorithme de model checking est donc **linéaire** dans la taille du **modèle** et **exponentielle** dans la taille de la **formule**. Cependant, on peut remarquer que les formules exprimant des propriétés temporelles usuelles, comme par exemple l'accessibilité ou la sûreté, sont de petite taille.