Logique du premier ordre
Logique temporelle et vérification

Richard Lassaigne

Institut de Mathématiques de Jussieu
Equipe de 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. Elle est très insuffisante pour représenter les procédés des langages effectivement utilisés en informatique, linguistique, mathématiques ou pour formaliser des fragments significatifs du raisonnement courant, 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* ?

Ce qui manque dans le langage propositionnel, c'est d'abord la possibilité 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* ;

Introduire la notion de **relation** unaire, binaire, etc. c'est se donner les moyens de traiter celle de **variable**, c'est-à-dire de la place que viendra occuper tel ou tel objet dans un énoncé. 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.*

Cette partie de la logique utilise ce que l'on appelle les **langages du premier ordre** pour plusieurs raisons :

- ces langages partagent avec le langage naturel, les langages de programmation et les langages d'interrogation des bases de données relationnelles certaines caractéristiques essentielles ;
- il existe une grande variété de langages du premier ordre, chacun étant déterminé par son vocabulaire propre ;
- ces langages permettent de représenter non seulement des relations, mais aussi des fonctions.

Enfin, l'expression **premier ordre** distingue ces langages, qui sont les plus utilisés, de ceux d'**ordre supérieur**, comme du **second ordre**, dans lesquels il est possible de quantifier également les relations et les fonctions. L'étude des langages du premier ordre suit la même progression que celle du langage propositionnel. La première section concerne la définition des **langages** et la construction des **formules**. Dans la seconde, la **sémantique** est présentée en terme de **structures** et d'interprétation des formules dans les structures. La troisième section présente le système de **déduction naturelle** pour la logique du premier ordre.

La principale qualité de la logique du premier ordre est de posséder un pouvoir d'expression très général. Mais le principal défaut, d'un point de vue informatique, est qu'il n'existe pas d'algorithme de décision général pour l'ensemble des théories du premier ordre, résultat fondamental dû à A. Turing (1936). Pour cette raison, si l'on s'intéresse aux applications potentielles de la logique, par exemple aux problèmes de vérification, il est très utile de considérer certaines formes de la logique temporelle. La dernière section est constituée par la présentation de la logique temporelle linéaire et d'une méthode de vérification pour les propriétés exprimables dans cette logique, appelée model checking.

1 Langages du premier ordre

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**.

En général, les langages du premier ordre peuvent comporter également des symboles permettant de représenter des fonctions. Comme l'on s'intéresse essentiellement à des applications comme les bases de données relationnelles, on ne considère ici que des symboles de relations et de constantes. Tous ces symboles sont destinés à être interprétés par la suite dans des structures. Pour le moment, la construction de l'ensemble des formules d'un langage est définie de manière purement syntaxique.

Exemple 1 *Le langage $\mathcal{L}$ comporte un symbole de relation R et un symbole de constante c et est noté $\mathcal{L} = \{R, c\}$.*

Un langage comporte le plus souvent un symbole de relation binaire supplémentaire, noté $=$. Ce symbole sera toujours interprété par l'égalité et le langage considéré sera dit **égalitaire**. Comme pour le langage propositionnel, la construction de l'ensemble des **formules** utilise le procédé de **définition par induction**. Mais le pouvoir d'expression de la logique du premier ordre est bien plus grand. En effet, les **termes**, ici les variables et les constantes représentent les objets, et les **formules atomiques** les relations entre objets.

1.1 L'ensemble des formules

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 le plus petit ensemble qui contient les formules atomiques et qui est clos par application des connecteurs et des quantificateurs.

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 $(F \wedge G)$, $(F \vee G)$,$(F \rightarrow G)$ 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.*

Exemple 2 *L'expression suivante est une formule du langage* $\mathcal{L}$ *:*

$$\forall x \forall y \forall z \ ((Rxy \wedge Ryz) \rightarrow Rxz)$$

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,*
- $(F \wedge G)$, $(F \vee G)$,$(F \rightarrow G)$ *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 démonstration est analogue à celle du calcul propositionnel et est laissée au lecteur. La décomposition complète d'une formule peut être obtenue récursivement : on fait apparaître toutes les formules entrant dans cette décomposition.

Exemple 3 *La décomposition de la formule* F *suivante est donnée par :*

$$F \ : \ (Rxz \rightarrow \forall z \ (Ryz \vee y = z))$$

$$Rxz$$

$$\forall z \ (Ryz \vee y = z)$$

$$(Ryz \vee y = z)$$

$$Ryz$$

$$y = z$$

Définition 4 *Une formule* G *est une* **sous-formule** *de la formule* F *si elle apparaît dans la décomposition de* F*.*

Exemple 4 *Toutes les formules apparaissant dans la décomposition de* F *sont des sous-formules de* F*.*

1.2 Variables libres, variables liées

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

Définition 5 *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.*

Exemple 5 *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 (l'apparition de* z *dans* $\forall z$ *ne constitue pas une occurrence de* z*).*

La définition de l'interprétation d'une formule dans une structure dépendra de façon essentielle de certaines occurrences des variables dans la formule, celles sur lesquelles les quantificateurs n'agissent pas.

Définition 6

- *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.*

Exemple 6 *Dans la formule* $F : (Rxz \to \forall z\ (Ryz \vee y = z))$*, la seule occurrence de* x *est libre, les deux occurrences de* y *sont libres, la première occurrence de* z *est libre et les autres sont liées. Les variables* x, y, z *sont donc libres dans* F*. Par contre, la formule* $\forall x\ \forall z\ (Rxz \to \exists y\ (Ryz \vee y = z))$ *est close.*

La notation $F(x_1, x_2, ..., x_k)$ signifie que les variables libres de la formule F sont parmi $x_1, x_2, ..., x_k$.

2 Sémantique

La sémantique de la logique du premier ordre est définie en interprétant les formules d'un langage donné dans les structures correspondantes. Ces structures peuvent être d'ordre mathématique ou représenter des types de données en informatique, les domaines des relations dans une base de données relationnelle, ou encore l'univers du discours en linguistique.

2.1 Structures et langages

Définition 7 *Une* **structure** $\mathcal{M}$ *pour un langage* $\mathcal{L}$ *se compose d'un ensemble non vide* M*, appelé* **domaine** *ou ensemble de base et :*

- *d'un sous-ensemble de* M^n*, notée* $R^{\mathcal{M}}$*, pour chaque symbole de prédicat* R *d'arité* n *;*
- *d'un élément de* M*, noté* $c^{\mathcal{M}}$*, pour chaque symbole de constante* c*.*

Exemple 7 *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* $G = (D, E, s)$*.*

2.2 Structures et satisfaction des formules

L'objet de ce paragraphe est de définir la **satisfaction** et la **valeur d'une formule** $F(x_1, x_2, ..., x_k)$ dans une structure $\mathcal{M}$ pour une suite $a_1, a_2, ..., a_k$ d'éléments de M interprétant les variables libres de F.

La **valeur d'un terme** t est définie de manière évidente :

- si t est une constante c, c'est l'élément distingué associé à c ;
- si t est une variable x_i, c'est l'élément correspondant a_i.

Soit $F(x_1, x_2, ..., x_k)$ une formule, $\mathcal{M}$ une structure et $s = (a_1, a_2, ..., a_k)$ une suite d'éléments de M de longueur k. La notion de **satisfaction** de la formule F par s dans $\mathcal{M}$ est définie par induction sur la formule F et est notée : $\mathcal{M} \models F[a_1, a_2, ..., a_k]$.

Définition 8 *L'expression* **la suite** s **satisfait la formule** F **dans** $\mathcal{M}$ *est définie par induction sur la formule* F *:*

- *si* F *est une formule atomique de la forme* $Rt_1t_2...t_n$ *et si* $b_1, b_2, ..., b_n$ *sont les valeurs respectives des termes* $t_1, t_2, ..., t_n$ *pour la suite* s*,* s *satisfait* F *dans* $\mathcal{M}$ *si et seulement si* $(b_1, b_2, ..., b_n) \in R^{\mathcal{M}}$ *;*
- *si* F *est de l'une des formes* $\neg G$*,* $(G \wedge H)$*,* $(G \vee H)$*,*$(G \rightarrow H)$ *ou* $(G \leftrightarrow H)$*, la satisfaction de* F *est définie à partir de celles de* G *et* H *comme dans le cas du calcul propositionnel ;*
- *si* F *est de la forme* $\exists x_0\ G(x_0, x_1, x_2, ..., x_k)$*,* s *satisfait* G *dans* $\mathcal{M}$ *si et seulement si il existe un élément* $a_0 \in M$ *tel que la suite* $s' = (a_0, a_1, a_2, ..., a_k)$ *satisfait* G *dans* $\mathcal{M}$ *;*
- *si* F *est de la forme* $\forall x_0\ G(x_0, x_1, x_2, ..., x_k)$*,* s *satisfait* G *dans* $\mathcal{M}$ *si et seulement si pour tout élément* $a_0 \in M$*, la suite* $s' = (a_0, a_1, a_2, ..., a_k)$ *satisfait* G *dans* $\mathcal{M}$*.*

Dans le cas où la suite s satisfait la formule F dans $\mathcal{M}$, on dit également que la formule F est **vraie** dans $\mathcal{M}$ pour s. Dans le cas contraire, F est dite **fausse** dans $\mathcal{M}$ pour s. Si F est une formule close, la satisfaction de F dans la structure $\mathcal{M}$ est indépendante de toute suite

d'éléments de M : elle est soit vraie, soit fausse dans $\mathcal{M}$. Dans le cas où elle est vraie, la structure $\mathcal{M}$ est appelée un **modèle** de F, ce que l'on note : $\mathcal{M} \models F$.

Exemple 8 *Un graphe non orienté $G = (D, E, a)$ est modèle de la formule $\forall x \forall y\ (Rxy \rightarrow Ryx)$ du langage $\mathcal{L} = \{R, c\}$ qui exprime que la relation E est symétrique. La formule $\exists x\ Rcx$ exprime que le sommet distingué a nest pas isolé.*

3 Déduction naturelle pour la logique du premier ordre

Le système de déduction utilisé ici pour montrer la complétude de la logique du 1er ordre est une extension du système de *déduction naturelle* présenté pour la logique propositionnelle. La relation de déduction est définie en utilisant les règles suivantes.

Définition 9 *La formule F est* **prouvable** *à partir d'un ensemble fini de formules Γ, ce que l'on note $\Gamma \vdash F$, si on peut l'obtenir en appliquant, un nombre fini de fois, les règles de la déduction naturelle pour la logique propositionnelle ainsi que les 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 telle suite finie d'applications des règles de déduction est aussi appelée***dérivation** *of $\Gamma \vdash F$. La formule F est prouvable à partir d'un ensemble de formules Γ s'il existe un ensemble fini $\Gamma_0 \subset \Gamma$ tel que $\Gamma_0 \vdash F$.*

Exemple 9 *On peut vérifier en exercice les propriétés suivantes de la négation vis-à-vis des quantificateurs :*

$$\Gamma \vdash \neg\forall x\ F \ \textit{iff}\ \Gamma \vdash \exists x\ \neg F$$

$$\Gamma \vdash \neg\exists x\ F \ \textit{iff}\ \Gamma \vdash \forall x\ \neg F$$

Une propriété importante de la déduction naturelle est la monotonie :
si $\Gamma \vdash F$ et si $\Gamma \subseteq \Gamma'$, alors $\Gamma' \vdash F$. La preuve se fait par une induction facile sur les règles de déduction.

Définition 10

- *Une* **théorie** *est un ensemble de formules closes.*
- *Une théorie T est* **cohérente** *s'il n'existe pas de formule F telle que $T \vdash F$ et $T \vdash \neg F$. Sinon, la théorie T est dite incohérente ou contradictoire.*
- *Une structure $\mathcal{M}$ est un modèle de la théorie T si $\mathcal{M}$ est un modèle de toutes les formules de T.*

Proposition 2 *Pour toute théorie T et toute formule close F, $T \vdash F$ si et seulement si $T \cup \{\neg F\}$ est incohérente.*

Preuve : Si $T \vdash F$, alors $T, \neg F \vdash F$ d'après la monotonie. Puisque $T, \neg F \vdash \neg F$, $T \cup \{\neg F\}$ est incohérente. Inversement, if $T \cup \{\neg F\}$ est incohérente, alors il existe une formule G telle que :

$$\dfrac{\dfrac{\dfrac{T, \neg F \vdash G \qquad T, \neg F \vdash \neg G}{T, \neg F \vdash \bot}}{T \vdash \neg\neg F}}{T \vdash F}$$

Proposition 3 *Si T est une théorie dont tout sous-ensemble fini est cohérent, alors T est cohérente.*

Preuve : La propriété se démontre par contraposition. Si T est incohérente, alors il existe une formule F telle que $T \vdash F$ and $T \vdash \neg F$. On en déduit $T \vdash (F \wedge \neg F)$ à l'aide de la règle d'introduction de la conjunction. Il existe donc un sous-ensemble fini T_0 de T tel que $T_0 \vdash (F \wedge \neg F)$, c'est-à-dire que T_0 est incohérent.

Définition 11 *Une formule F est* **conséquence** *de la théorie T si tout modèle de T est modèle de F.*

Proposition 4 *Une formule F est conséquence de la théorie T si et seulement si la théorie $T \cup \{\neg F\}$ n'a pas de modèle.*

Preuve : 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.*

4 Logique temporelle linéaire et model checking

La vérification par model checking s'intéresse aux systèmes réactifs, c'est-à-dire à des systèmes qui interagissent avec un environnement, comme par exemple les systèmes embarqués ou les protocoles de communication. Les propriétés que l'on veut vérifier sur ces systèmes sont des propriétés temporelles du type :

- 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).

Le model checking est un ensemble de techniques de vérification automatique de propriétés temporelles sur des systèmes réactifs. Les systèmes en question sont modélisés par des systèmes de transition, appelés aussi modèles de Kripke.

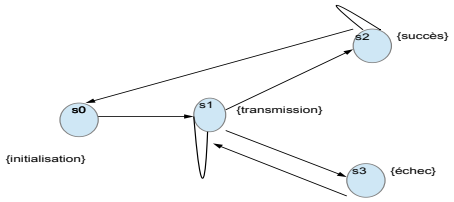
4.1 Systèmes de transition

Un système de transition $\mathcal{M} = \{S, R, s_0, l\}$ est défini par :

- 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 $l : S \longrightarrow 2^{AP}$.

En général, l'ensemble S est fini et on peut alors représenter le système de transition par un graphe orienté et étiqueté. On suppose dans la suite que pour tout état s, il existe un état t tel que $R(s,t)$.

Exemple 10 *Exemple de système de transition*



L'ensemble des exécutions du système à partir de l'état initial s_0 peut être représenté par un arbre, l'arbre des chemins d'origine s_0.

Définition 12 *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, $(s_i, s_{i+1}) \in R$.*

Etant donné un chemin σ, on note σ^i le chemin extrait de σ à partir de la position i et $\sigma(i)$ le i-ième état de σ.
On remarque que si $\mathcal{M}$ est un système à branchement fini, i.e. que pour tout état $s \in S$, il n'existe qu'un nombre fini d'états successeurs de s, l'ensemble des exécutions du système à partir de l'état initial s_0 peut être représenté par un arbre finitaire.

4.2 Logique temporelle linéaire (LTL)

L'ensemble des formules de chemin est défini par induction : c'est le plus petit ensemble

- contenant l'ensemble AP des variables propositionnelles, ainsi qu'une formule $vrai$,
- clos par application des connecteurs propositionnels $\neg, \wedge, \vee$,
- clos par application des opérateurs temporels X et U,

L'interprétation d'une formule de chemin φ sur un chemin σ d'un système de transition $\mathcal{M}$ est définie par induction :

- si $\varphi = vrai$, $(\mathcal{M}, \sigma) \models vrai$,
- si φ est une variable propositionnelle p, $(\mathcal{M}, \sigma) \models p$ ssi $p \in l(s_0)$,
- si φ est de la forme $\neg\psi$, $(\mathcal{M}, \sigma) \models \neg\psi$ ssi $(\mathcal{M}, \sigma) \not\models \psi$,
- si φ est de la forme $\varphi_1 \wedge \varphi_2$, $(\mathcal{M}, \sigma) \models \varphi_1 \wedge \varphi_2$ ssi $(\mathcal{M}, \sigma) \models \varphi_1$ et $(\mathcal{M}, \sigma) \models \varphi_2$,

- si φ est de la forme $\varphi_1 \vee \varphi_2$, $(\mathcal{M}, \sigma) \models \varphi_1 \vee \varphi_2$ ssi $(\mathcal{M}, \sigma) \models \varphi_1$ ou $(\mathcal{M}, \sigma) \models \varphi_2$,
- si φ est de la forme $X\psi$, $(\mathcal{M}, \sigma) \models X\psi$ ssi $(\mathcal{M}, \sigma^1) \models \psi$,
- si φ est de la forme $\varphi_1 U \varphi_2$, $(\mathcal{M}, \sigma) \models \varphi_1 U \varphi_2$ ssi il existe $i \geq 0$ tel que $(\mathcal{M}, \sigma^i) \models \varphi_2$ et que pour tout $0 \leq j < i$, $(\mathcal{M}, \sigma^j) \models \varphi_1$.

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 $\mathcal{M} \models \forall\ \varphi$.

Très souvent, on utilise les formules $F\varphi$ et $G\varphi$ comme abréviations pour les formules suivantes : $F\varphi \equiv vraiU\varphi$ et $G\varphi \equiv \neg F \neg \varphi$. On peut remarquer que les significations de ces deux formules correspondent bien aux propriétés d'accessibilité et de sûreté :

- la formule φ sera vraie dans un état futur (au sens large) pour $F\varphi$,
- la formule φ sera vraie dans tous les états futurs (au sens large) pour $G\varphi$.

Exemple 11 *Exemple de propriétés temporelles pour le système précédent :*

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

A titre d'exercice, on peut vérifier l'équivalence des deux formules suivantes :

$$\varphi_1 U \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge X(\varphi_1 U \varphi_2))$$

4.3 Model checking de LTL et automates

Il est possible de voir un système de transition comme un automate dont tous les états sont acceptants. Cependant, comme les chemins d'exécution sont infinis, il est nécessaire de considérer des automates sur des mots infinis.

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

Une exécution r de $\mathcal{A}$ sur un mot infini $w = a_0 \dots a_i \cdots \in \Sigma^\omega$ est une 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 13

- *Soit* $inf(r) = \{q \ / \ q = q_i$ *pour une infinité de* $i\}$. *Une exécution* r *de* $\mathcal{A}$ *est acceptante si* $inf(r) \cap F \neq \emptyset$, *i.e. 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.*

Il est possible de transformer un système de transition $\mathcal{M}$ en un automate $\mathcal{A}_\mathcal{M}$ en rajoutant un état initial ι, en considérant tous les états du système comme acceptants et en mettant les étiquettes sur les transitions et non plus sur les états. Il est un peu plus compliqué d'associer un automate à une formule de chemin de LTL. Il est facile de voir que les transformations associées aux connecteurs propositionnels $\wedge, \vee$ et $\neg$ sont respectivement l'intersection, la réunion et le passage au complémentaire sur les automates. Cependant, pour les opérateurs temporels, il est nécessaire de faire des constructions *ad hoc*. Le principe du model checking de LTL est le suivant :

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 l'algorithme 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. On peut remarquer que les formules exprimant des propriétés temporelles usuelles, comme par exemple l'accessibilité ou la sûreté, sont de petite taille.

5 Exercices

Exercice 1. Dans le langage $\mathcal{L} = \{R, c\}$, donner deux formules closes qui expriment respectivement :

- qu'il n'existe pas de sommet avec une boucle sur lui-même,
- qu'il existe un chemin de longueur 3 dans le graphe.

Exercice 2. Dans le langage $\mathcal{L} = \{R, c\}$, donner une formule $F(x, y)$ à 2 variables libres exprimant qu'il existe un chemin de longueur n entre deux sommets.

Exercice 3. Dans le langage $\mathcal{L} = \{R, c\}$, déterminer trois formules closes qui expriment respectivement :

- que tout sommet a exactement une arête sortante,
- que tout sommet a exactement une arête entrante,
- que le graphe est un cycle.

Exercice 4. Un graphe est dit *complet* si pour tout couple de sommets, il existe une arête les reliant. Donner une formule exprimant l'existence d'un sous-graphe complet de taille k.

Exercice 5. Dans un graphe, un *recouvrement* de sommets est un sous-ensemble de sommets tel que toute arête ait l'une de ses extrémités dans ce sous-ensemble. Dans le langage $\mathcal{L}_1 = \{R, U\}$, où R est un symbole de relation binaire et U un symbole de relation unaire, donner une formule exprimant que l'interprétation de U est un recouvrement.

Exercice 6. Un 3-*coloriage* d'un graphe est une partition de l'ensemble des sommets en trois couleurs. Dans le langage $\mathcal{L}_2 = \{R, U_1, U_2, U_3\}$, où R est un symbole de relation binaire et U_1, U_2, U_3 des symboles de relation unaire, donner deux formules closes exprimant respectivement que :

- l'interprétation de U_1, U_2, U_3 est un 3-coloriage,
- l'interprétation de U_1, U_2, U_3 est un *bon* 3-coloriage, i.e. que pour tout couple de sommets reliés par une arête, les couleurs des 2 sommets sont différentes.

Exercice 7. Quelques affirmations concernant les connecteurs temporels :

- Fp est-il vrai si p est vrai dans l'état courant ?
- Gp est-il vrai si p est faux dans l'état courant et vrai partout ailleurs ?
- pUq est-il vrai si p est faux et q est vrai dans l'état courant ?
- pUq est-il vrai si q est toujours faux et p toujours vrai ?

Exercice 8. On désire modéliser de manière temporelle le fonctionnement d'un système de feux : on choisit 3 variables propositionnelles r, o, v pour représenter *rouge,orange,vert.* Déterminer des formules temporelles exprimant les propriétés suivantes :

- le système ne reste pas tout le temps au *rouge*,
- un état *orange* est toujours immédiatement suivi d'un état *rouge*,
- le système passe infiniment souvent par un état *vert.*

Exercice 9.

1. Définir formellement l'interprétation de deux connecteurs supplémentaires :
   - $F^{\infty}p$ (*infiniment souvent*) : *p est infiniment vrai au long du chemin d'exécution*,
   - $G^{\infty}p$ (*presque toujours*) : *à partir d'un moment donné, p est toujours vrai.*
2. Exprimer ces deux connecteurs à l'aide des connecteurs F et G.

Exercice 10. Pour un système de transition concernant l'émission et la réception de messages, exprimer en langage naturel les propriétés suivantes :

- $G(emission \rightarrow Freception)$,
- $F^{\infty}ok \rightarrow G(emission \rightarrow Freception)$.

Exercice 11. Pour le système de transition de l'exemple 10, déterminer, pour chacune des formules temporelles suivantes, un chemin d'exécution la satisfaisant :

- $initialisation \rightarrow X(transmissionUsucces)$,
- $F(succes \wedge X(initialisationUtransmission)$,
- $G\neg echec$.