1 Introduction

Une logique devient temporelle quand on ajoute à chacune de ses fonctions (ou chacun de ses prédicats) un paramètre supplémentaire: le temps. Nous nous concentrerons ici sur les logiques temporelles propositionnelles; celles-ci se plongent naturellement dans des logiques du premier ordre (ou du second ordre) monadiques (c'est à dire chaque symbole ayant pour seul argument le temps).

Les logiques monadiques de second ordre ont été bien étudiées dans les années 60: à la suite des travaux, entre autres, de Büchi, Elgott et Rabin, les modèles de ces formules sont reconnus par des automates appropriés. C'est ainsi que les automates entrent dans le paysage et fournissent des techniques de décision pour divers problèmes concernant les logiques temporelles tels la satisfaisabilité et la vérification (model-checking).

Nous montrons donc, après les avoir introduites, diverses logiques temporelles et leurs traduction en logique du premier (resp. second) ordre ainsi que le lien entre les logiques monadiques de second ordre et les automates.

Malheureusement, les techniques générales de Büchi ou Rabin donnent des algorithmes de complexité énorme, qui sont bien moins bons que les algorithmes spécialisés dans les logiques temporelles et qui ont été proposés dans les années 80. L'objectif du cours est de montrer comment rétablir la situation en expliquant des techniques optimales pour le model-checking et la satisfaisabilité des logiques temporelles. Ces techniques se fondent sur deux outils (mathématiques): les *automates alternants* et les *jeux*; pour chaque logique temporelle, l'ensemble des modèles est un langage reconnu par un automate alternant approprié, et les propriétés de ces automates s'expriment à l'aide de jeux à deux joueurs sur un graphe fini. La nature de ce graphe et les conditions de victoire déterminent la complexité du calcul d'une stratégie gagnante, ou, de manière équivalente, la satisfaction de la formule.

Un plan un peu plus détaillé est donné en fin de section.

1.1 Les systèmes réactifs

Un exemple est donné dans la figure 1; il s'agit d'un modèle ultra simplifié d'une machine à café. Cet automatisme est supposé communiquer avec un environnement par le biais de signaux émis ou reçus. Les étiquettes de la forme ?l sont des actions de réception de signaux, alors que les actions !l sont des émissions. Les envois/réceptions de tels messages par l'automatisme provoquent des changements d'état. On peut imaginer, par exemple, une synchronisation basée sur le rendez-vous (les changements d'états ne peuvent s'effectuer que si l'émission et la réception simultanée d'un même message est possible).

On s'intéresse à la question de savoir si de tels automates satisfont des propriétés comme:

> Chaque fois que j'ai mis deux francs, alors, plus tard, j'obtiendrai un café.

formule qui reste à préciser, bien sûr.

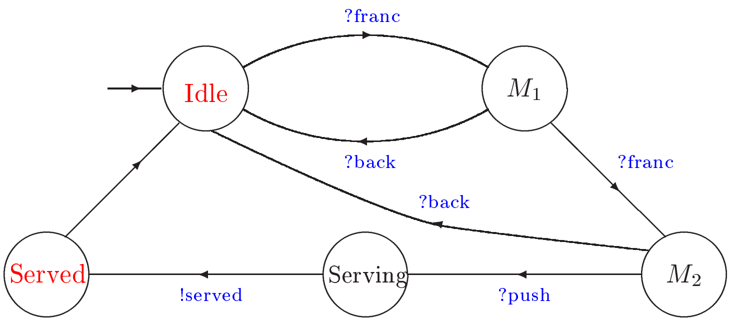

Figure 1: Une machine à café très simple

Si l'on compare ce problème à la vérfication de programmes, on peut noter que:

- On s'intéresse essentiellement aux comportements infinis (et donc pas aux fonctions dont le calcul se termine!)
- On s'intéresse aux états successifs et pas au "résultat" d'un calcul

1.2 Les logiques temporelles

Logique temporelle = logique + temps.

Ce qui correspond à l'ajout d'un paramètre "temps" aux symboles de fonction et de prédicat.

Ici, on s'interessera uniquement aux logiques temporelles propositionnelles.

Exemple 1 Reprenons l'exemple précédent (figure 1)

$$\text{Money} \quad \Rightarrow \qquad \text{Served}$$

est une formule du calcul propositionnel sur les variables propositionnelles Money et Served.

$$\forall t.\text{Money}(t) \Rightarrow \exists t' > t.\text{Served}(t')$$

est un énoncé de logique temporelle propositionnelle.

Exemple 2

$$\exists P. \quad (\qquad P \quad \Rightarrow \text{Idle} \quad) \wedge (\qquad \text{Idle} \quad \Rightarrow P \quad)$$

est une formule propositionnelle de second ordre et

$$\exists P.\forall t.(\exists t' > t.P(t) \Rightarrow \text{Idle}(t')) \wedge (\exists t' > t.\text{Idle}(t) \Rightarrow P(t'))$$

est une formule temporelle de second ordre.

1.3 Le temps

Linéaire

vs. Arborescent $\left\{\begin{array}{l}\text{branchement fini}\\ \text{branchement infini}\end{array}\right.$

Discret

vs. Réel

Infini

vs. Transfini

Avec origine du temps

vs. Sans origine du temps

1111 $\mathbb{N}$

2111 $\{0,1\}^*$ avec l'ordre préfixe

1211 $\mathbb{R}_+$

1121 ϵ_0

1112 $\mathbb{Z}$

3221 $\{[0,t] \to \mathbb{R}_+ ; \mid t \in \mathbb{R}_+\}$ (ordre= $t < t'$ et f coïncide avec f' sur $[0,t]$).

Exercice 1 On considère la formule du premier ordre suivante:

$$\begin{array}{ll} & \exists t.P(t) \\ \wedge & \exists t.\neg P(t) \\ \wedge & \forall t.P(t) \Rightarrow ((\exists t' > t.P(t')) \wedge \forall t' < t.P(t')) \end{array} \qquad ds$$

Dire, suivant les modèles du temps, si cette formule est satisfaisable ou non (et justifier sa réponse soit par une preuve d'insatisfaisabilité, soit en exhibant un modèle).

1.4 Les problèmes

	Donnée	Question
Vérification (modèles "plats")	M, ϕ	$M \models \phi$?
Vérification (modèles "concurrents")	$M_1, \ldots, M_n, \phi$	$M_1 \times \ldots \times M_n \models \phi$?
Satisfaisabilité	ϕ	existe-t-il M tel que $M \models \phi$?
Synthèse	ϕ	Donner M tel que $M \models \phi$

$\times$ étant un produit synchrone qui reste à préciser, comme beaucoup d'autres choses d'ailleurs.

Les formules ϕ considérées seront des formules de logique temporelle propositionnelle (ou de second ordre propositionel) avec divers modèles du temps. Nous nous limiterons dans le cours aux modèles suivants du temps: $\mathbb{N}$, $\{1, \ldots, n\}^*$, $\mathbb{N}^*$. On parlera alors de *temps arborescent* dans le premier cas (sans autre précision) et de *temps arborescent* dans les deux derniers cas (soit à branchement borné, soit à branchement non borné).

Par ailleurs, nous considérerons les logiques suivantes (qui seront définies dans le cours) et qui sont les plus couramment utilisées:

Temps linéaire: LTL, S1S

Temps arborescent: CTL, CTL*, μ-calcul, SkS

On se préoccupera d'*efficacité*.

1.5 Complexité de la vérification

1.5.1 Rappels des classes de complexité

Type de machine :

D = déterministe

N = non-déterministe

A = alternante

Quantité mesurée :

TIME : le temps de calcul

SPACE: l'espace de calcul

La classe de fonctions :

P Polynômial

EXP Exponentiel

2-EXP Doublement exponentiel

Rappelons quelques résultats (cf. [Joh90] pour une synthèse):

Si $f(n) \geq n$, ATIME$f(n)) \subseteq$ DSPACE$(f(n)) \subseteq \bigcup_{c>0}$ ATIME$(c \times f(n)^2)$.

En particulier, APTIME = DPSPACE.

NSPACE$(f(n)) \subseteq$ DSPACE$(f(n)^2)$

En particulier NPSPACE =DP SPACEsi bien qu'on omet Nou D.

Si $f(n) \geq \log n$, alors ASPACE$(f(n)) = \bigcup_{c>0}$ DTIME$(2^{c \times f(n)})$

En particulier APSPACE= DEXPTIME.

1.5.2 Résumé des résultats connus

Les résultats ci-dessous ont été obtenus dans les années 80 (par ex. [SC85, QS82, CES86, ?, VW94, Cle93, EJ88]). Nous nous proposons ici de les établir tous d'une manière plus uniforme.

Il faudrait aussi ajouter d'autres résultats pour d'autres modèles du temps. (ex. temps réel).

	Complexité du model-checking	Complexité de la satisfaisabilité	Complexité de la vérification de systèmes concurrents
LTL	PSPACE-complet	PSPACE-complet	PSPACE-complet
CTL	PTIME-complet	DEXPTIME-complete	PSPACE-complet
CTL*	PSPACE-complet	D2-EXPTIME-complet	PSPACE-complet
Alternation-free μ-calculus	Polynomial	DEXPTIME-complet	DEXPTIME-complet
μ-calculus	NPTIME $\cap$ co-NPTIME	DEXPTIME-complet	DEXPTIME-complet

Ces complexités sont toutes optimales, sauf le model-checking pour le μ-calcul: on ne sait pas actuellement s'il existe un algorithme polynômial.

1.6 Plan du cours

Le cours commencera par introduire les concepts de base dans le cas du temps linéaire: logique du temps linéaire (LTL), logique monadique de second ordre à un successeur (S1S), automates de Büchi et correspondance entre S1S et automates de Büchi, en laissant de côté le théorème de complémentation. Ensuite, nous introduirons la notion clef d'automate alternant. En découlent des algorithmes simples et optimaux pour les problèmes de décision de LTL. Nous reviendrons alors sur le théorème de complémentation: nous introduirons les jeux à parité en montrant le lien avec les automates alternants. La complémentation des automates alternants à parité découle d'un théorème simple de détermination des jeux.

Nous nous consacrerons ensuite au temps arborescent, en commençant cette fois par la logique monadique de second ordre à k successeur et les automates d'arbres, sans toutefois prouver le théorème de complémentation (voir [Tho97] pour une preuve accessible basée sur un théorème de détermination des jeux). Puis nous introduisons les logiques CTL et CTL*, les automates alternants d'arbres, et leur lien avec les logiques temporelles. Nous en déduisons des algorithmes optimaux pour les divers prblèmes de décision. Enfin, nous terminons par le μ-calcul, reprenant les techniques déjà mises en place, en particulier les jeux à parité.

Les sources bibliographiques sont essentiellement:

- Thomas (1997) pour les logiques monadiques de second ordre [Tho97]
- Vardi (1994) pour la logique du temps linéaire [Var96]
- Kupferman, Vardi, Wolper (2000) pour les logiques du temps arborescent [KVW00]
- Thomas (1997–1999) [Tho99, Tho97] Pour les liens logique-automates alternants-jeux.

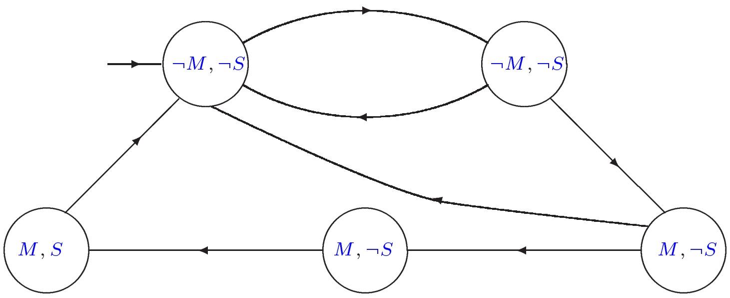

Figure 2: Une structure de Kripke finie

2 Structures de Kripke

2.1 Définition

$\mathcal{K} = (S, R, s^0, f, \mathcal{P})$ où $R \subseteq S \times S$, $s^0 \in S$ et $f : S \to \mathcal{P}$. On suppose que, pour tout $s \in S$, il existe au moins un $s' \in S$ tel que $R(s, s')$.

Exemple 3 Si l'on ne tient pas compte des actions et que l'on étiquette chacun des états de l'automate de la figure 1 par un sous-ensemble de $\{M, S\}$, on obtient l'automate et le marquages comme indiqués sur la figure 2; M est supposé indiquer que l'on a déjà mis deux francs dans la machine et S que le café est servi.

Exemple 4 Si l'on complique un peu le modèle de la machine à café en acceptant de pouvoir mettre un nombre arbitraire de pièces, on obtient le modèle suivant: $S = \{s^i \mid i \in \mathbb{N}\} \cup \{t^i \mid i \geq 2\} \cup \{u^i \mid i \geq 2\}$,

$$\begin{array}{ccll} s^i & \to & s^{i+1} & \text{pour } i \in \mathbb{N} \\ s^i & \to & s^0 & \text{pour } i \in \mathbb{N} \\ s^i & \to & t^i & \text{pour } i \geq 2 \\ t^i & \to & u^i & \text{pour } i \geq 2 \\ u^i & \to & s^{i-2} & \text{pour } i \geq 2 \end{array}$$

et $f(t^i) = \{M\}$, $f(u^i) = \{S, M\}$ et $f(s^i) = \emptyset$.

2.2 Plusieurs modalités

Si l'on veut tenir compte des émissions/réceptions de messages, on peut considérer le modèle plus général (plusieurs modalités): on considère un nombre fini d'actions $A = \{a_1, \ldots, a_n\}$ et une relation binaire par action. Par exemple, si on reprend l'automate de la figure 1, on peut considérer les relations binaires $R_{?franc}, R_{?back}, \ldots$.

Elargir le modèle à plusieurs modalités ne présente pas beaucoup d'intérêt car on peut "traduire" le cas multimodal en une seule modalité en remplaçant S par $S \times A$ (soit en devinant à chaque étape l'action qui va suivre, soit en se rappelant de l'action qui précède).

2.3 Executions et dépliage

Une *exécution* de la structure de Kripke est une séquence infinie $w \in S^\omega$ telle que pour tout $i \in \mathbb{N}$, $w(i)Rw(i+1)$ et $w(0) = s^0$

Un sous-ensemble D de $\mathbb{N}_+^*$ est un *domaine d'arbre* s' il est

- clos par préfixe
- clos lexicographiquement: $p \cdot i \in D$ entraine $\forall j, 1 \leq j \leq i, p \cdot j \in D$.

Un *arbre* est une application T d'un domaine d'arbre D dans un ensemble d'étiquettes A.

Si $\mathcal{K}$ est une structure de Kripke de branchement fini (i.e. pour tout $s \in S$, au plus un nombre fini d'éléments de S sont en relation avec s), l'ensemble des branches issues de chaque sommet est totalement ordonné de façon arbitraire. Le *dépliage* de la structure de Kripke est un arbre (éventuellement infini) étiqueté par S défini comme suit: $\epsilon \in D$ et $T(\epsilon) = s^0$. Si $p \in D$ et $T(p) = s$ et $\{s_1, \ldots, s_n\}$ sont les éléments de S tels que sRs_i pour tout i, alors $p \cdot 1, \ldots p \cdot n \in D$ et, pour tout i, $T(p \cdot i) = s_i$.

Le dépliage à partir de $s \in S$ s'obtient de la même façon en remplaçant s^0 par s.

3 Logique du temps linéaire (LTL)

3.1 Syntaxe

Etant donné un ensemble de variables propositionnelles $\mathcal{P}$, l'ensemble des formules de LTL est le plus petit ensemble tel que, si ϕ, ψ sont des formules, alors

true, **false**

P $(P \in \mathcal{P})$

$\mathbf{X}\phi$

$\phi \wedge \psi$, $\phi \vee \psi$

$\neg\phi$

$\phi\, \mathcal{U} \psi$

sont des formules.

On utilise couramment les abréviations suivantes:

$\Diamond\phi \stackrel{\text{def}}{=} \mathbf{true}\, \mathcal{U}\phi$

$\Box\phi \stackrel{\text{def}}{=} \neg(\mathbf{true}\, \mathcal{U}\neg\phi)$

3.2 Sémantique

Soit $\mathcal{K}$ une structure de Kripke. Par définition, $\mathcal{K}$ satisfait ϕ si, pour toute exécution w de $\mathcal{K}$, $(w, 0) \models \phi$, où $\models$ est définie par:

$w, t \models \mathbf{true}$ et $w, t \not\models \mathbf{false}$

$w, t \models P$ ssi $P \in f(w(t))$

$w, t \models \mathbf{X}\phi$ ssi $w, t+1 \models \phi$

$w, t \models \phi \wedge \psi$ ssi $w, t \models \phi$ et $w, t \models \psi$

$w, t \models \phi \vee \psi$ ssi $w, t \models \phi$ ou $w, t \models \psi$

$w, t \models \neg\phi$ ssi $w, t \not\models \phi$.

$w, t \models \phi\, \mathcal{U}\psi$ ssi

- il existe $t' \geq t$ tel que $w, t' \models \psi$
- et pour tout $t'' \in [t, t'[$ on a $w, t'' \models \phi$.

Exercice 2 On note $\phi \equiv \psi$ lorsque ϕ et ψ ont les mêmes modèles. Démontrer ou invalider les équivalences suivantes:

1. $\mathbf{X}\neg\phi \equiv \neg\mathbf{X}\phi$
2. $\mathbf{X}\Diamond\phi \equiv \Diamond\mathbf{X}\phi$
3. $\Box\mathbf{X}\phi \equiv \mathbf{X}\Box\phi$
4. $\mathbf{X}\phi\wedge\mathbf{X}\psi \equiv \mathbf{X}(\phi\wedge\psi)$
5. $\mathbf{X}\phi\vee\mathbf{X}\psi \equiv \mathbf{X}(\phi\vee\psi)$
6. $(\mathbf{X}\phi)\,\mathcal{U}\psi \equiv \psi\vee\mathbf{X}(\phi\,\mathcal{U}(\phi\wedge\psi))$
7. $\neg\Diamond\phi \equiv \Box\neg\phi$
8. $\neg\Box\phi \equiv \Diamond\neg\phi$

Exercice 3 Montrer que $\mathcal{U}$ n'est pas associatif:

$$(\phi_1\,\mathcal{U}\phi_2)\,\mathcal{U}\phi_3 \not\equiv \phi_1\,\mathcal{U}(\phi_2\,\mathcal{U}\phi_3)$$

Il suffira d' exhiber des formules ϕ_1, ϕ_2 et ϕ_3 bien choisies.

3.3 Langage défini par une formule de LTL

Un *mot infini* sur l'alphabet A est une application de $\mathbb{N}$ dans A. L'ensemble des mots infinis construits sur A est noté A^ω. Si $w \in A^\omega$, $w(i)$ pour $i \in \mathbb{N}$ est la $i$ème lettre de w.

L'ensemble des variables propositionnelles étant ordonné de manière arbitraire, une affectation de ces variables est identifiée à un mot de $\{0,1\}^n$. La notion de satisfaction d'une formule ϕ de LTL pour un mot $w \in (\{0,1\}^k)^\omega$ est définie comme dans la section 3.2, la fonction f étant définie par $f(\alpha) = \{P_i \mid \alpha(i) = 1\}$.

Le *langage* L_ϕ défini par une formule LTL ϕ est l'ensemble des mots $w \in (\{0,1\}^k)^\omega$ qui satisfont ϕ.

Exemple 5 Supposant qu'il y a une seule proposition P, si ϕ est la formule $\Box\Diamond P$, $L_\phi = (0^*1)^\omega$, c'est à dire l'ensemble des mots contenant une infinité de 1.

De même, si $\mathcal{K}$ est une structure de Kripke, on note $L_\mathcal{K}$ le langage qu'elle définit sur l'alphabet $\{0,1\}^k$: $L_\mathcal{K}$ est l'ensemble des images par f des exécutions de $\mathcal{K}$.

On peut alors reformuler les problèmes de satisfaisabilité et de vérification de modèles dans ce contexte:

$$\boxed{\phi \text{ est satisfaisable} \quad \Leftrightarrow \quad L_\phi \neq \emptyset}$$

$$\boxed{\mathcal{K} \models \phi \quad \Leftrightarrow \quad L_\mathcal{K} \subseteq L_\phi}$$

3.4 logique monadique du premier ordre à un successeur

On considère la logique du premier ordre *monadique*, à un successeur construite sur un ensemble fini de prédicats unaires $\mathcal{Q}$: les termes sont construits à partir de $0, S$ et des variables. Les formules atomiques sont de la forme $t \geq t'$ ou bien $t = t'$ ou bien $P(t)$, $P \in \mathcal{Q}$, t, t' étant des termes.

On interprète $0, S, \geq, =$ dans les entiers naturels: S est le successeur. Si $\mathcal{Q} = \{P_1, \ldots, P_k\}$, une interprétation est représentée par k ensembles d'entiers, ou, si l'on superpose leurs fonctions caractéristiques, par des mots de $(\{0,1\}^k)^\omega$.

Par exemple, si $\mathcal{Q} = \{P_1, P_2\}$, et $\phi \stackrel{\text{def}}{=} \forall t. P_1(0) \wedge P_1(t) \Rightarrow P_2(S(t)) \wedge P_2(t) \Rightarrow P_1(S(t))$, alors $w \stackrel{\text{def}}{=} \begin{smallmatrix}1\\0\end{smallmatrix}\begin{smallmatrix}0\\1\end{smallmatrix}\begin{smallmatrix}1\\0\end{smallmatrix}\begin{smallmatrix}0\\1\end{smallmatrix}\ldots = (\begin{smallmatrix}1\\0\end{smallmatrix}\begin{smallmatrix}0\\1\end{smallmatrix})^\omega$ satisfait ϕ.

Exercice 4 Donner d'autres modèles de la formule ϕ ci-dessus.

3.5 Traduction de LTL en logique du premier ordre

Si ϕ est une formule de LTL construite sur les variables propositionnelles $\mathcal{P}$, on construit une formule $\tilde{\phi}(t)$ de la logique monadique de premier ordre sur les prédicats unaires $\{Q_P \mid P \in \mathcal{P}\}$, avec une variable libre t de telle manière que, pour toute structure de Kripke $\mathcal{K}$ (sur l'ensemble de variables propositionnelles $\mathcal{P}$):

$$\mathcal{K} \models \phi \quad \Leftrightarrow \quad L_{\mathcal{K}} = \{w \mid w \models \tilde{\phi}(0)\}$$

$\widetilde{\mathbf{true}} \stackrel{\text{def}}{=} \mathbf{true}$

$\widetilde{\mathbf{false}} \stackrel{\text{def}}{=} \mathbf{false}$

$\widetilde{P}(t) \stackrel{\text{def}}{=} Q_P(t)$.

$\widetilde{\phi \wedge \psi}(t) \stackrel{\text{def}}{=} \widetilde{\phi}(t) \wedge \widetilde{\psi}(t)$

$\widetilde{\phi \vee \psi}(t) \stackrel{\text{def}}{=} \widetilde{\phi}(t) \vee \widetilde{\psi}(t)$

$\widetilde{\neg\phi}(t) \stackrel{\text{def}}{=} \neg\widetilde{\phi}(t)$

$\widetilde{\mathbf{X}\phi}(t) \stackrel{\text{def}}{=} \widetilde{\phi}(t+1)$

$\widetilde{\phi\,\mathcal{U}\psi}(t) \stackrel{\text{def}}{=} \exists t'. t' \geq t \wedge \widetilde{\psi}(t') \wedge \forall t''. (t \leq t'' < t' \Rightarrow \widetilde{\phi}(t''))$

Theorème 6 (Kamp) *L est définissable en logique monadique du premier ordre si et seulement si L est définissable en LTL.*

Plus précisément: on peut traduire les formules de LTL en logique monadique du premier ordre (c'est ce qui est fait dans le paragraphe précédent) et réciproquement (on l'admettra), il est possible de traduire toute formule du premier ordre monadique en une formule de LTL qui définit le même langage. Cette dernière traduction est, en fait, non élémentaire.