IGL502/752 – Techniques de vérification et de validation

Université de Sherbrooke

Examen final

Enseignant: Michael Blondin
Date: jeudi 14 décembre 2023
Durée: 3 heures

Directives:

— Vous devez répondre aux questions dans le **cahier de réponses**, et non sur ce questionnaire;

— **Une seule feuille** de notes au format 8½" $\times$ 11" est permise;

— Les **fiches récapitulatives** se trouvent à la fin du questionnaire;

— **Aucun matériel additionnel** (notes de cours, fiches récapitulatives, etc.) n'est permis;

— **Aucun appareil électronique** (calculatrice, téléphone, montre intelligente, etc.) n'est permis;

— Vous devez donner **une seule réponse** par sous-question;

— L'examen comporte **6 questions** sur **6 pages** valant un total de **50 points**;

— La correction se base notamment sur la **clarté**, l'**exactitude** et la **concision** de vos réponses, ainsi que sur la **justification** pour les questions qui en requièrent une.

Question 1: logique temporelle linéaire (LTL)

Soit $AP \coloneqq \{p, q, r\}$ et les formules LTL suivantes sur AP:

$$\varphi_1 \coloneqq p \,\mathsf{U}\, ((\neg q) \,\mathsf{U}\, r) \qquad \varphi_2 \coloneqq (\mathsf{FG}p) \vee (\mathsf{XX}r) \qquad \varphi_3 \coloneqq \mathsf{G}(p \vee \mathsf{F}r)$$

(a) Pour chaque formule φ_i, donnez un mot σ_i qui la satisfait et qui ne satisfait pas les deux autres, c.-à-d. 6 pts

$$\begin{array}{lll} \sigma_1 \models \varphi_1 & \sigma_1 \not\models \varphi_2 & \sigma_1 \not\models \varphi_3, \\ \sigma_2 \not\models \varphi_1 & \sigma_2 \models \varphi_2 & \sigma_2 \not\models \varphi_3, \\ \sigma_3 \not\models \varphi_1 & \sigma_3 \not\models \varphi_2 & \sigma_3 \models \varphi_3. \end{array}$$

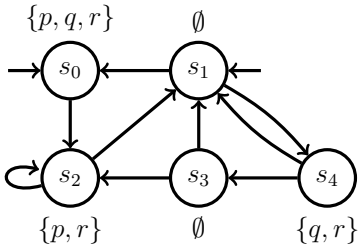
Une solution parmi d'autres:

1. $\{r\}\emptyset^\omega$
2. $\emptyset\{p\}^\omega$
3. $(\{q\}\{r\})^\omega$

(b) Donnez un mot σ qui enfreint à la fois φ_1, φ_2 *et* φ_3, c.-à-d. tel que $\sigma \not\models \varphi_1$, $\sigma \not\models \varphi_2$ et $\sigma \not\models \varphi_3$. 2 pts

Une solution parmi d'autres: $\emptyset^\omega$.

(c) Pour chaque i, dites si cette structure de Kripke $\mathcal{T}$ satisfait φ_i. Justifiez. 3 pts

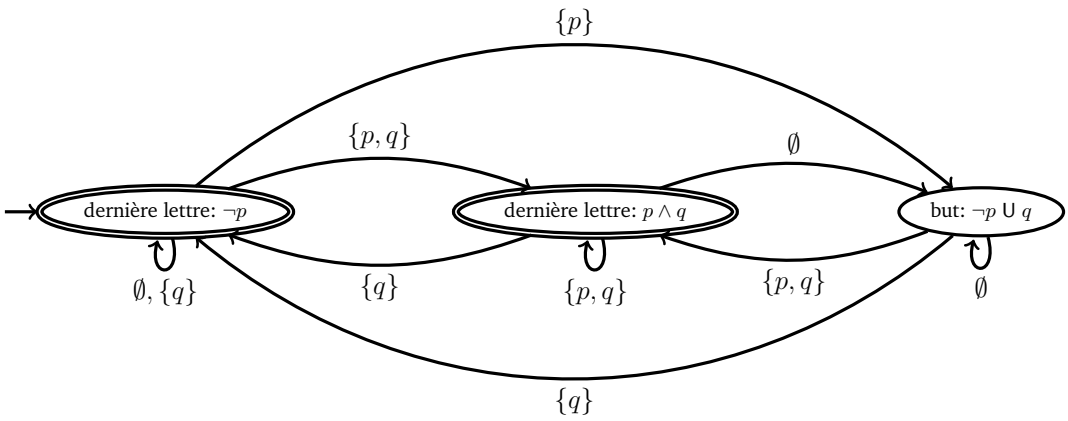


1. Oui. En s_0, on satisfait immédiatement r. En s_1, on satisfait $\neg q$, puis on satisfait forcément r au second moment car on doit se déplacer en s_0 ou s_4.
2. Non, trace$((s_1\, s_4)^\omega)$ enfreint la formule car p n'est jamais satisfaite et r n'est pas satisfaite au troisième moment.
3. Oui. Il n'y a aucun cycle dans le sous-graphe induit par $\{s_1, s_3\}$. Comme les autres états satisfont tous r, on satisfait r infiniment souvent peu importe l'exécution.

Question 2: automates de Büchi

(a) Donnez un automate de Büchi $\mathcal{B}$ tel que $\mathcal{L}(\mathcal{B}) = [\![\mathsf{G}(p \to \mathsf{X}(\neg p\ \mathsf{U}\ q))]\!]$ sur alphabet $\Sigma \coloneqq \{\emptyset, \{p\}, \{q\}, \{p, q\}\}$. 4 pts

La formule est équivalente à $\mathsf{G}(\neg p \vee \mathsf{X}(\neg p\ \mathsf{U}\ q))$, pour laquelle nous donnons cet automate:



(b) Rappelons que selon la construction vue en classe, les états de l'automate, résultant de l'intersection de deux automates de Büchi $\mathcal{A}$ et $\mathcal{B}$, sont de la forme (p, q, I) où p est un état de $\mathcal{A}$, q est un état de $\mathcal{B}$, et $I \in \{\mathcal{A}, \mathcal{B}\}$. Il y a plusieurs façons de choisir l'ensemble des états acceptants. Lesquels de ces choix sont corrects? 2 pts

(i) les états où p est acceptant dans $\mathcal{A}$ et $I = \mathcal{A}$;

(ii) les états où p est acceptant dans $\mathcal{A}$ et $I = \mathcal{B}$;

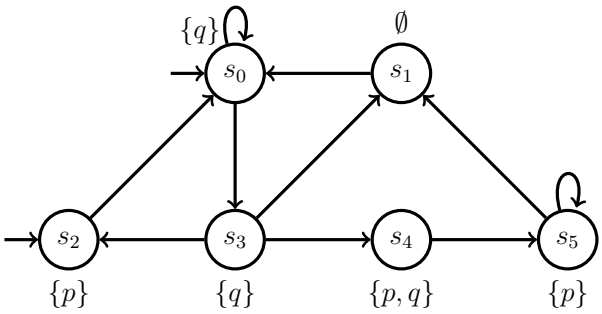
(iii) les états où p est acceptant dans $\mathcal{A}$ et $I = \mathcal{A}$, ainsi que ceux où q est acceptant dans $\mathcal{B}$ et $I = \mathcal{B}$.

(i) et (iii).

Question 3: logique temporelle arborescente (CTL) et vérification symbolique

Rappel: l'abbréviation « BDD » réfère à « diagramme de décision binaire (ordonné et réduit) ».

Supposons que chaque état de la structure de Kripke $\mathcal{T}$ ci-dessous soit codé par la représentation binaire de son indice: $s_0 = 000$, $s_1 = 001$, $s_2 = 010$, $s_3 = 011$, $s_4 = 100$ et $s_5 = 101$.



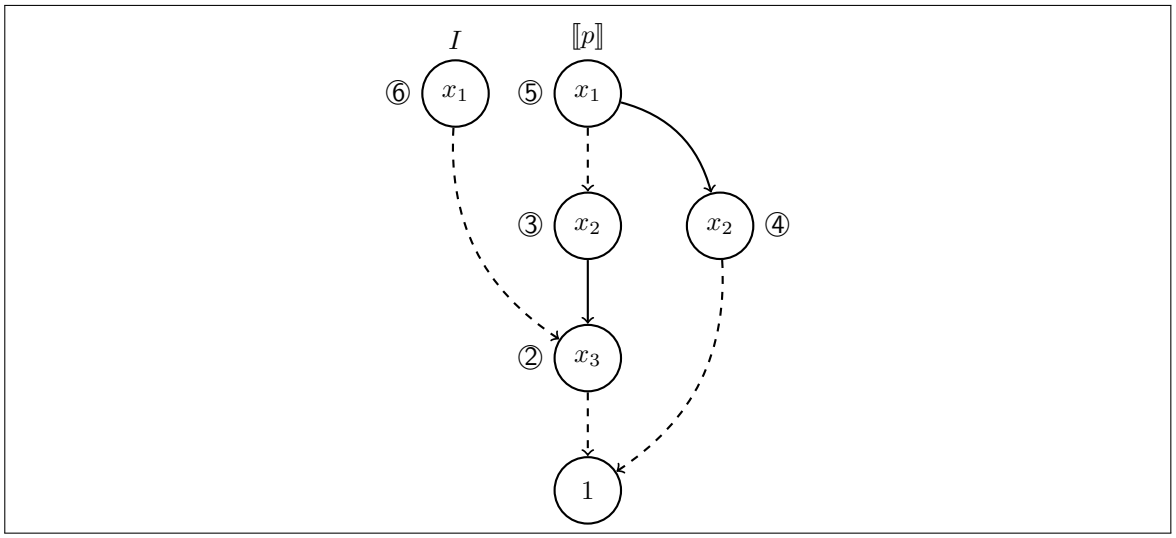
(a) Pour chaque formule Φ ci-dessous, donnez l'ensemble $[\![\Phi]\!]$ des états de $\mathcal{T}$ qui satisfont Φ. 6 pts

(i) $\exists \mathsf{G}(\neg p \vee q)$ (ii) $\exists(q \,\mathsf{U}\, (p \wedge \neg q))$ (iii) $\forall \mathsf{X}\, \exists \mathsf{G}(\neg p \vee q)$

(i) $\{s_0, s_1, s_3\}$ (ii) $\{s_0, s_2, s_3, s_4, s_5\}$ (iii) $\{s_0, s_1, s_2\}$

(b) Construisez un BDD qui représente l'ensemble des états initiaux I. 2 pts

(c) Construisez un BDD qui représente l'ensemble $[\![p]\!]$. 2 pts



(d) Expliquez comment vérifier algorithmiquement si $\mathcal{T} \models p$ à partir des BDDs construits en (b) et (c). 2 pts

On doit vérifier que $I \subseteq [\![p]\!]$, ce qui est équivalent au test $\texttt{apply}_{\rightarrow}(⑥, ⑤) = ①$.

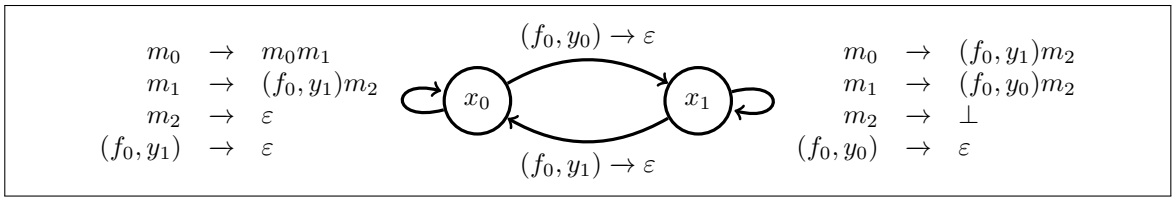
Question 4: systèmes à pile

Considérons ce programme constitué de deux fonctions et d'une variable booléenne globale x:

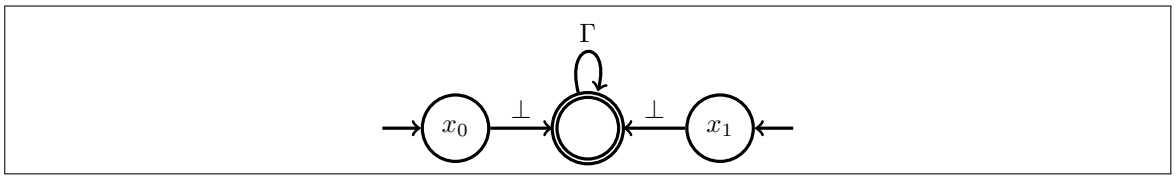
```
                bool x ∈ {faux, vrai}

                main():                    foo(bool y):
m0:               si x:                f0:   x = ¬y
                    foo(x)
                  sinon:
                    main()
m1:               foo(¬x)
m2:             assert(¬x)
```

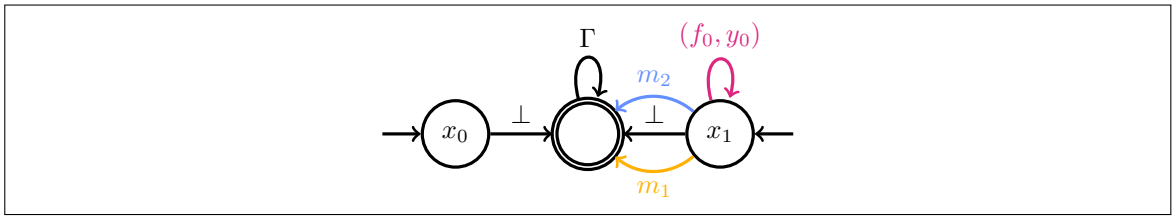
(a) Modélisez le programme avec un système à pile $\mathcal{P}$. 2,5 pts



(b) Donnez un $\mathcal{P}$-automate $\mathcal{A}$ tel que *Conf*$(\mathcal{A})$ est l'ensemble des configurations où l'assertion est enfreinte. 1 pt



(c) Construisez partiellement un $\mathcal{P}$-automate $\mathcal{B}$ qui accepte Pre*(*Conf*$(\mathcal{A})$). Plus précisément, donnez au moins trois nouvelles transitions obtenues à partir de $\mathcal{A}$ en exécutant l'algorithme de saturation vu en classe. Au moins deux de ces transitions doivent être obtenues *sans* utiliser une transition de $\mathcal{P}$ étiquetée par une règle de la forme « *lettre* $\to \varepsilon$ ». 2,5 pts

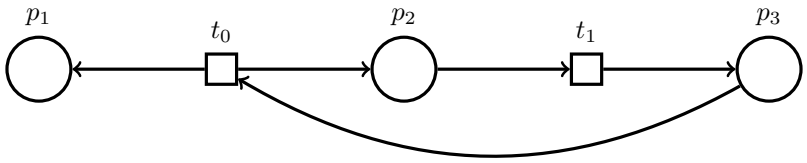


(d) Si vous aviez entièrement calculé le $\mathcal{P}$-automate $\mathcal{B}$ en (c), comment auriez-vous pu déterminer si l'assertion du programme est toujours satisfaite? 1 pt

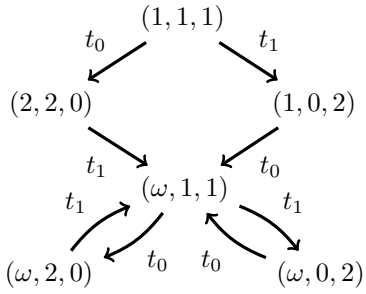
Elle est toujours satisfaite ssi $\langle x_0, m_0 \rangle \notin$ *Conf*$(\mathcal{B})$ et $\langle x_1, m_0 \rangle \notin$ *Conf*$(\mathcal{B})$. Afin de tester $\langle x_i, m_0 \rangle \notin$ *Conf*$(\mathcal{B})$, on vérifie qu'il n'y a pas de transition de l'état x_i vers l'état acceptant étiquetée par m_0.

Question 5: systèmes à compteurs

Soit le réseau de Petri $\mathcal{N} = (P, T, F)$ suivant:



(a) Dessinez un graphe de couverture qui débute en $\boldsymbol{m} \coloneqq (1,1,1)$. 3 pts



(b) Dites lesquels de ces marquages peuvent être couverts à partir de $\boldsymbol{m} \coloneqq (1,1,1)$. Justifiez brièvement. 1,5 pts

$$\boldsymbol{m}_0 \coloneqq (1,1,2), \qquad \boldsymbol{m}_1 \coloneqq (0,3,2), \qquad \boldsymbol{m}_2 \coloneqq (4,2,0).$$

— $\boldsymbol{m}_0$ n'est pas couvrable car tous les sommets sont incomparables avec $\boldsymbol{m}_0$ sur les deux dernières composantes.
— $\boldsymbol{m}_1$ n'est pas couvrable car tous les sommets sont incomparables avec $\boldsymbol{m}_1$ sur les deux dernières composantes.
— $\boldsymbol{m}_2$ est couvrable car $(\omega,2,0) \geq \boldsymbol{m}_2$.

(c) L'ensemble des marquages qui peuvent couvrir $\boldsymbol{m}' \coloneqq (1,0,0)$ est $\uparrow\{(1,0,0),(0,1,0),(0,0,1)\}$. Pourquoi? 2,5 pts

Sol. 1. Remarquons que:
— À partir de $(0,0,0)$ rien n'est déclenchable, donc clairement $(0,0,0)$ ne peut pas couvrir $\boldsymbol{m}'$;
— Le marquage $(1,0,0)$ couvre trivialement $\boldsymbol{m}'$;
— Nous avons $(0,1,0) \xrightarrow{t_1} (0,0,1) \xrightarrow{t_2} (1,1,0)$, donc $(0,1,0)$ et $(0,0,1)$ peuvent tous les deux couvrir $\boldsymbol{m}'$.

Comme $\mathrm{Pre}^*(\uparrow\boldsymbol{m}')$ est clos par le haut, nous avons bien $\mathrm{Pre}^*(\uparrow\boldsymbol{m}') = \uparrow\{(1,0,0),(0,1,0),(0,0,1)\}$.

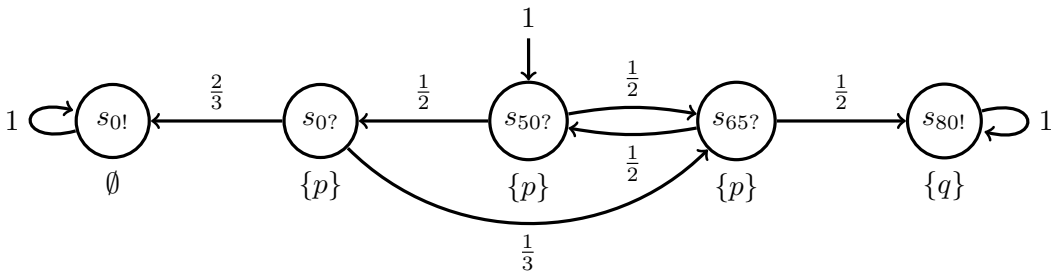
Sol. 2. Cela se vérifie avec l'algorithme arrière:

Itér.	**Base**	**Prédecesseurs**
0	$\{(1,0,0)\}$	$(1,0,0)_{t_0} = (0,0,1)$ $(1,0,0)_{t_1} = (1,1,0)$
1	$\{(1,0,0),(0,0,1)\}$	$(0,0,1)_{t_0} = (0,0,2)$ $(0,0,1)_{t_1} = (0,1,0)$
2	$\{(1,0,0),(0,0,1),(0,1,0)\}$	$(0,1,0)_{t_0} = (0,0,1)$ $(0,1,0)_{t_1} = (0,2,0)$

Question 6: systèmes probabilistes

Considérons un casino où l'on peut entrer avec 50$ en poche. On est forcé d'y jouer à répétition jusqu'à gagner le montant maximal ou tout perdre. Lorsqu'on perd, le casino nous rend parfois un montant d'argent et nous force à rejouer.

Le casino est modélisé par la chaîne de Markov $\mathcal{M}$ ci-dessous. Un état de la forme $s_{x?}$ indique qu'on détient x\$, mais qu'on ne sait pas si c'est notre montant final. Un état de la forme de $s_{x!}$ indique qu'on détient x\$ et qu'on doit quitter le casino. Les propositions atomiques p et q indiquent respectivement que nous sommes en train de jouer et que nous avons gagné.



Remarque: il n'est pas obligatoire d'appliquer des algorithmes pour répondre aux questions.

(a) Donnez $\mathbb{P}(s_{50?} \models \mathsf{G}\, p)$, c.-à-d. la probabilité de ne jamais quitter le casino. Justifiez. 2 pts

Remarquons que les seuls états qui satisfont $\neg p$ appartiennent aux composantes fortement connexes terminales. Ainsi, nous avons

$$\begin{aligned}\mathbb{P}(s_{50?} \models \mathsf{G}\, p) &= 1 - \mathbb{P}(s_{50?} \models \mathsf{F}\, \neg p) \\ &= 1 - 1 && \text{(par le théorème des comportements limites)} \\ &= 0.\end{aligned}$$

(b) Donnez $\mathbb{P}(s_{50?} \models \mathsf{G}^{\leq 3}\, p)$, c.-à-d. la probabilité d'être encore dans le casino après trois joutes. Justifiez. 2 pts

$$\mathbb{P}(s_{50?} \models \mathsf{G}^{\leq 3}\, p) = \underbrace{(1/2)\cdot(1/3)\cdot(1/2)}_{s_{50?}\to s_{0?}\to s_{65?}\to s_{50?}} + \overbrace{(1/2)^3}^{s_{50?}\to s_{65?}\to s_{50?}\to s_{65?}} + \underbrace{(1/2)^3}_{s_{50?}\to s_{65?}\to s_{50?}\to s_{0?}} = (1/12) + (2/8) = 1/3.$$

(c) Le propriétaire du casino prétend que son établissement est « juste » car $\mathcal{M}$ satisfait $\mathcal{P}_{=\frac{1}{2}}(\mathsf{F}\, q)$. Montrez que $\mathcal{M}$ satisfait en effet cette propriété PCTL. *(le gain espéré du casino est donc de 10$, ce qui n'est pas si juste!)* 3 pts

Sol. 1. Posons $S_0 \coloneqq \{s_{0!}\}$, $S_1 \coloneqq \{s_{80!}\}$, $S_? \coloneqq \{s_{0?}, s_{50?}, s_{65?}\}$,

$$\mathbf{A} \coloneqq \begin{pmatrix} 0 & 0 & 1/3 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{pmatrix} \text{ et } \boldsymbol{b} \coloneqq \begin{pmatrix} 0 \\ 0 \\ 1/2 \end{pmatrix}.$$

Nous devons vérifier que l'unique solution du système $(\mathbf{I} - \mathbf{A}) \cdot (x, y, z)^{\mathsf{T}} = \boldsymbol{b}$ satisfait $y = 1/2$. En substituant $y = 1/2$ dans les trois équations, on obtient:

$$\begin{aligned} x \qquad - z/3 &= 0, \\ -x/2 + 1/2 - z/2 &= 0, \\ -1/4 + \quad z &= 1/2. \end{aligned}$$

Par la dernière équation, $z = 3/4$. Par la première équation, $x = 1/4$. La deuxième équation est satisfaite puisque $-1/8 + 1/2 - 3/8 = 0$. Nous avons donc $y = 1/2$ comme attendu.

Sol. 2. Par le théorème des comportements limites, il suffit de vérifier que $\mathbb{P}(s_{50?} \models \mathsf{F}\, s_{0!}) = 1/2$. Remarquons qu'il y a deux cycles, autour de l'état initial, de probabilité $1/2 \cdot 1/2 = 1/4$ et $1/2 \cdot 1/2 \cdot 1/3 = 1/12$ respectivement. Ils peuvent être combinés de façon arbitraire. Nous avons ainsi:

$$\begin{aligned} \mathbb{P}(s_{50?} \models \mathsf{F}\, s_{0!}) &= \left[\sum_{k=0}^{\infty} \sum_{i=0}^{k} \binom{k}{i} \cdot \left(\frac{1}{4}\right)^{i} \cdot \left(\frac{1}{12}\right)^{k-i}\right] \cdot \frac{1}{2} \cdot \frac{2}{3} \\ &= \frac{1}{3} \cdot \sum_{k=0}^{\infty} \sum_{i=0}^{k} \binom{k}{i} \cdot \left(\frac{1}{4}\right)^{i} \cdot \left(\frac{1}{12}\right)^{k-i} \\ &= \frac{1}{3} \cdot \sum_{k=0}^{\infty} \left(\frac{1}{4} + \frac{1}{12}\right)^{k} \\ &= \frac{1}{3} \cdot \sum_{k=0}^{\infty} \left(\frac{1}{3}\right)^{k} \\ &= \frac{1}{3} \cdot \frac{1}{1 - 1/3} \\ &= \frac{1}{3} \cdot \frac{3}{2} \\ &= \frac{1}{2}. \end{aligned}$$

Remarque: une personne a une probabilité de $1/2$ de gagner 30$ et une probabilité de $1/2$ de perdre 50$. Ainsi, le gain espéré d'une personne est de $(30 - 50)/2 = -10$$!

Fiches récapitulatives

1. Systèmes de transition

Systèmes de transition

- Modélise formellement un système concret
- *États*: ensemble S (sommets)
- *Relation de transition*: $\to \subseteq S \times S$ (arcs)
- *États initiaux*: $I \subseteq S$ (où S peut débuter)

Prédécesseurs et successeurs

- *Successeurs immédiats*: $\text{Post}(s_1) = \{s_2\}$
- *Prédécesseurs immédiats*: $\text{Pre}(s_1) = \{s_0, s_2\}$
- *Successeurs*: $\text{Post}^*(s_1) = \{s_1, s_2, s_3\}$
- *Prédécesseurs*: $\text{Pre}^*(s_1) = \{s_0, s_1, s_2\}$
- *États terminaux*: s_3 car $\text{Post}(s_3) = \emptyset$

Chemins et exécutions

- *Chemin fini*: $s_0 \to s_1 \to s_2 \to s_1$
- *Chemin infini*: $s_1 \to s_2 \to s_1 \to s_2 \to \cdots$
- *Ch. maximal*: ne peut être étendu, par ex. $s_0 \to s_1 \to s_2 \to s_3$
- *Exécution*: chemin maximal qui débute par un état initial

Structures de Kripke

- Décrit les propriétés des états d'un système
- *Système de transition* $(S, \to, I)$
- *Propositions atomiques AP*
- *Fonction d'étiquetage* $L\colon S \to 2^{AP}$
- *Exemple*: si $AP = \{p, q, r\}$ et $L(s_1) = \{p, q\}$, alors s_1 satisfait p et q, mais pas r

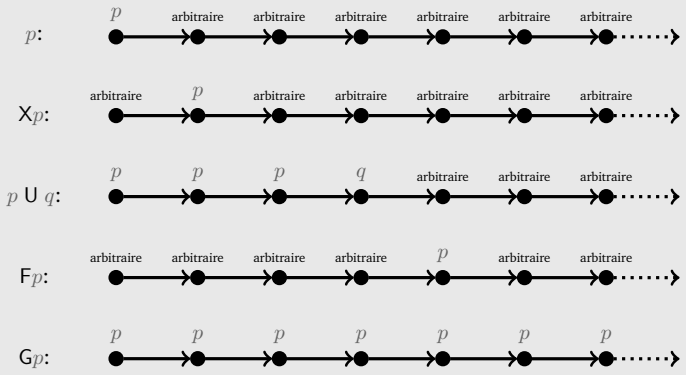
Explosion combinatoire

- Nombre d'états peut croître rapidement, par ex. $\mathcal{O}(2^n)$
- Existe techniques pour surmonter ce problème

2. Logique temporelle linéaire (LTL)

Logique

- *Intérêt*: spécifier formellement des propriétés
- *Syntaxe*: *vrai* | p | $\varphi \wedge \varphi$ | $\varphi \vee \varphi$ | $\neg\varphi$ | $\mathsf{X}\varphi$ | $\varphi\ \mathsf{U}\ \varphi$ | $\mathsf{F}\varphi$ | $\mathsf{G}\varphi$
- *Interprétation*: sur des traces, c.-à-d. mots infinis de $(2^{AP})^\omega$
- *Sémantique*:



Équivalences

- *Distributivité*: X, G, U (gauche) sur $\wedge$; X, F, U (droite) sur $\vee$
- *Dualité*: X dual de lui-même, F dual de G
- *Autre*: seules combinaisons de F et G: $\{\mathsf{F}, \mathsf{G}, \mathsf{FG}, \mathsf{GF}\}$

Types de propriétés

- *Invariant*: propriété toujours vraie: $\mathsf{G}\varphi$
- *Sûreté*: réfutable avec préfixe fini
- *Vivacité*: comportements vers l'infini

Vérification

- *Trace*: états d'une exécution infinie $\mapsto$ leurs étiquettes
- *Satisfaisabilité*: $\mathcal{T} \models \varphi \iff \text{Traces}(\mathcal{T}) \subseteq [\![\varphi]\!]$
- *Équité*: omettre traces triviales où un processus est ignoré
- *En pratique*: Spin (avec Promela), par ex. algorithme de Lamport, protocole de Needham-Schroeder

3. Langages ω-réguliers

Expressions ω-régulières

- *Décrivent*: les langages ω-réguliers de mots infinis
- *Syntaxe*:

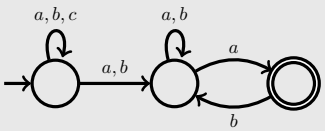
$$s ::= r^\omega \mid (r \cdot s) \mid (s + s)$$
$$r ::= r^* \mid (r \cdot r) \mid (r + r) \mid a \mid \varepsilon$$

- *Exemples*:

$a(a+b)^\omega$: mots qui débutent par a,
$(ab)^\omega$: l'unique mot $ababab\cdots$
$b^*(aa^*bb^*)^\omega$: mots avec une infinité de a *et* de b
$(a+b)^*b^\omega$: mots avec un nombre fini de a

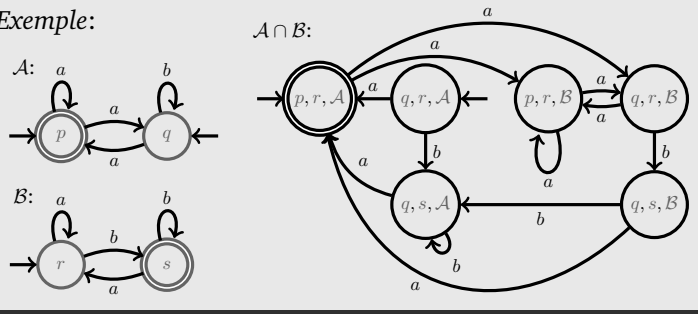
Automates de Büchi

- *Définition*: automates usuels; plusieurs états initiaux
- *Langage*: mots qui visitent états acceptants ∞ souvent
- *Expressivité*: $\equiv$ expressions ω-rég.; déterminisme $\neq$ non dét.
- *Exemple*: mots tels que $\#a = \infty$, $\#b = \infty$ et $\#c \neq \infty$:



Intersection d'automates de Büchi

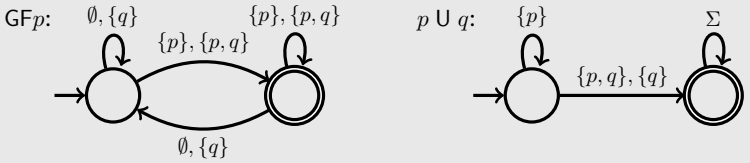
- $1^{ère}$ *idée*: simuler $\mathcal{A}$ et $\mathcal{B}$ en parallèle via produit; pas suffisant
- *Solution*: faire deux copies, alterner aux états acceptants
- *Exemple*:



4. Vérification algorithmique de formules LTL

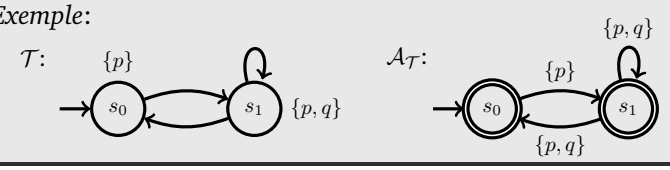
LTL vers automates

- *Alphabet*: $\Sigma := 2^{AP}$
- *Conversion*: $\varphi \to \mathcal{A}_\varphi$ (pire cas: $2^{\mathcal{O}(|\varphi|)}$ états)
- *Exemples*:



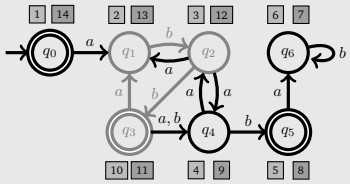
Structures de Kripke vers automates

- *Conversion*: étiquettes ≡ lettres + tout acceptant
- *Exemple*:

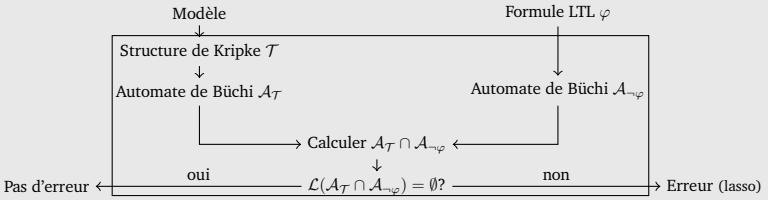


Test du vide

- *Vérification*: $\mathcal{T} \models \varphi \iff \mathcal{L}(\mathcal{A}_\mathcal{T}) \cap \mathcal{L}(\mathcal{A}_{\neg\varphi}) = \emptyset$
- *Lassos*: $\mathcal{L}(\mathcal{B}) \neq \emptyset \iff \exists q_0 \xrightarrow{*} q \xrightarrow{+} q$ où $q_0 \in Q_0$, $q \in F$
- *Détection*: double parcours en profondeur (temps linéaire)



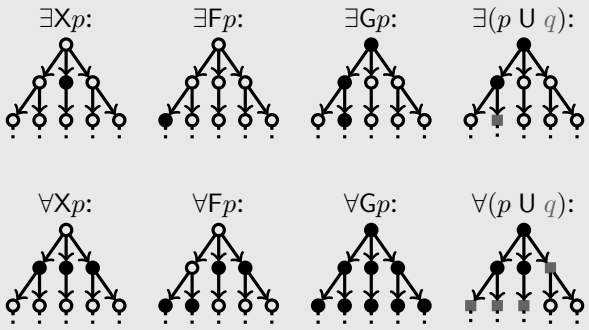
- *Mieux*: identifier les composantes fort. conn. (temps linéaire)
- *Sommaire*:



5. Logique temporelle arborescente (CTL)

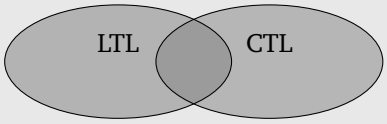
Logique

- *Intérêt*: raisonne sur le temps avec un futur indéterminé
- *Syntaxe*: *vrai* | p | $\Phi \wedge \Phi$ | $\Phi \vee \Phi$ | $\neg\Phi$ | $QT\Phi$ | $Q(\Phi$ U $\Phi)$
  où $Q \in \{\exists, \forall\}$, $T \in \{\mathsf{X}, \mathsf{F}, \mathsf{G}\}$
- *Interprétation*: sur l'arbre de calcul d'une structure de Kripke
- *Sémantique*:



Propriétés d'un système

- *Satisfiabilité dépend des états*: $[\![\Phi]\!] := \{s \in S : s \models \Phi\}$
- *Spécification*: $\mathcal{T} \models \Phi \iff I \subseteq [\![\Phi]\!]$
- *Expressivité*: incomparable à LTL



Équivalences

- *Distributivité*:

$$\exists\mathsf{F}(\Phi_1 \vee \Phi_2) \equiv (\exists\mathsf{F}\Phi_1) \vee (\exists\mathsf{F}\Phi_2)$$
$$\forall\mathsf{G}(\Phi_1 \wedge \Phi_2) \equiv (\forall\mathsf{G}\Phi_1) \wedge (\forall\mathsf{G}\Phi_2)$$

- *Attention*: pas équiv. si on change les quantificateurs
- *Dualité*: effet d'une négation: $\exists \leftrightarrow \forall$, $\mathsf{X} \leftrightarrow \mathsf{X}$ et $\mathsf{F} \leftrightarrow \mathsf{G}$
- *Idempotence*: $QT\,QT\Phi \equiv QT\Phi$ où $Q \in \{\exists, \forall\}$, $T \in \{\mathsf{F}, \mathsf{G}\}$

6. Vérification algorithmique de formules CTL

Algorithme

- *Approche*: calculer $[\![\Phi']\!]$ pour chaque sous-formule Φ' de Φ
- *Vérification*: tester $I \subseteq [\![\Phi]\!]$
- *Forme normale existentielle* plus simple, mais pas nécessaire
- *Complexité*: $\mathcal{O}((|S| + |\to|) \cdot |\Phi|)$ avec bonne implémentation
- *En pratique*: NuSMV + langage de description de haut niveau

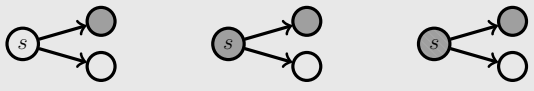
Logique propositionnelle

- *Règles récursives*:

$$[\![\mathit{vrai}]\!] = S,$$
$$[\![p]\!] = \{s \in S : p \in L(s)\},$$
$$[\![\Phi_1 \wedge \Phi_2]\!] = [\![\Phi_1]\!] \cap [\![\Phi_2]\!],$$
$$[\![\neg\Phi]\!] = S \setminus [\![\Phi]\!].$$

Opérateurs temporels existentiels

- *Calcul de* $[\![\exists\mathsf{X}\Phi]\!]$: $\{s \in S : \mathrm{Post}(s) \cap [\![\Phi]\!] \neq \emptyset\}$
- *Calcul de* $[\![\exists\mathsf{G}\Phi]\!]$: $T \subseteq [\![\Phi]\!]$ max. t.q. $\forall s \in T : \mathrm{Post}(s) \cap T \neq \emptyset$
- *Calcul de* $\exists(\Phi_1$ U $\Phi_2)$: $T \supseteq [\![\Phi_2]\!]$ min. t.q.
  $s \in [\![\Phi_1]\!] \wedge \mathrm{Post}(s) \cap T \neq \emptyset) \implies s \in T$



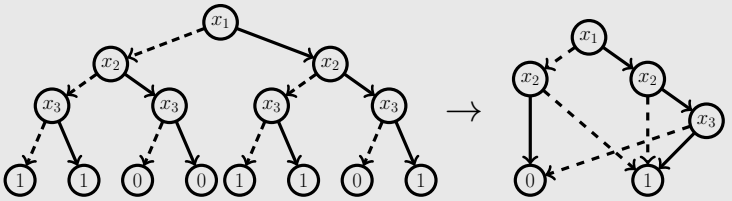
Optimisations

- *Autres opérateurs*: $\forall$ et F s'implémentent directement; nécessaire pour obtenir un algorithme polynomial
- *Points fixes*: temps linéaire si calculs directs sans raffinements itératifs; par ex. calcul de composantes fortement connexes

7. Vérification symbolique : diagrammes de décision binaire

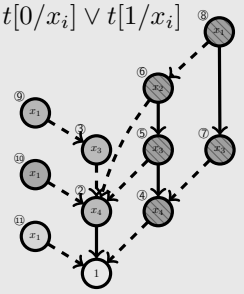
Diagramme de décision binaire

- *But*: représenter des fonctions booléennes de façon compacte
- *Utilité*: manipuler efficacement de grands ensembles d'états
- *Propriétés*:
  - graphe dirigé acyclique
  - sommets étiquetés par variables ordonnées sauf 0 et 1
  - les chemins respectent l'ordre des variables
  - sommets *uniques* et *non redondants* ($lo(u) \neq hi(u)$)
- *Canonicité*: pas deux BDDs pour la même fonction booléenne



Manipulation

- *Représentation*: tableau associatif *sommet* $\leftrightarrow (x_i, lo, hi)$
- *Ajout d'un sommet*: temps constant avec $\texttt{make}(x_i, lo, hi)$
- *Construction*: par substitutions récursives avec $\texttt{build}(\varphi)$
- *Op. bool.*: application récursive « synchronisée » avec $\texttt{apply}_{\bigcirc}$
- *Quantif.*: $\texttt{exists}\, x_i \in \{0,1\} : t$ obtenu via $t[0/x_i] \vee t[1/x_i]$
- *Complexité*: polynomiale sauf pour `build`



Vérification

- *État*: représenté par identifiant binaire
- *Transition*: paire d'identifiants binaires
- *Ensemble*: représenté par sommet de BDD
- *Logique prop.*: manipulation de BDD
- *Opérateurs temporels*: via calculs de Post ou Pre sur BDD
- *Satisfiabilité*: $I \subseteq [\![\Phi]\!] \Leftrightarrow I \cap \overline{[\![\Phi]\!]} = \emptyset \Leftrightarrow \texttt{apply}_{\wedge}(u_I, u_{\overline{[\![\Phi]\!]}}) = 0$

8. Systèmes avec récursion

Contexte

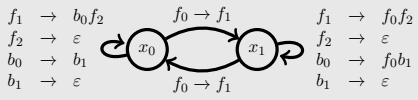
- *Espace d'états*: pile d'appel ou d'éléments (+ valeurs locales)
- *Défi*: gérer un nombre infini ou arbitraire d'états
- *Approche*: construction et analyse de systèmes à pile

Systèmes à pile

- *Définition*: états P, alphabet Γ, transitions $\{p \xrightarrow{a \to u} q, \ldots\}$
- *But*: décrire un ensemble de piles (et non accepter des mots)
- *Configuration*: $\langle p, w \rangle \in P \times \Gamma^* \mapsto (p)$ + pile 
- *Exemple de modélisation*:

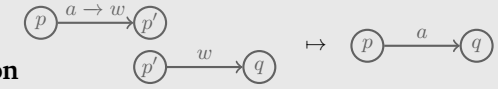
```
bool x ∈ {faux,vrai}

foo():                    bar():
f0:  x = ¬x:              b0:  si x: foo()
f1:  si x: foo()          b1:  retourner
     sinon: bar()
f2:  retourner
```



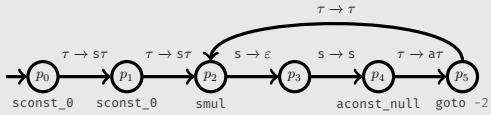
Calcul de prédécesseurs/successeurs

- *Déf.*: $\text{Pre}^*(C) := \bigcup_{i \geq 0} \text{Pre}^i(C)$ et $\text{Post}^*(C) := \bigcup_{i \geq 0} \text{Post}^i(C)$
- *Représentation*: symbolique de C avec un $\mathcal{P}$-automate $\mathcal{A}$
- *Idée*: (états initiaux = états de $\mathcal{P}$) + mots sur alphabet de pile
- *Décrit*: $\mathit{Conf}(\mathcal{A}) := \{\langle p, w \rangle : p \xrightarrow{w}_{\mathcal{A}} ◎\}$
- *Algorithme*: permet de calculer $\text{Pre}^*(\mathit{Conf}(\mathcal{A}))$ par saturation
- *Approche*: init. $\mathcal{B} := \mathcal{A}$ puis enrichir avec cette règle:



Vérification

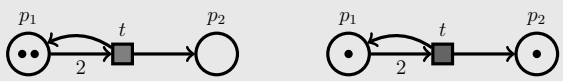
- *Approche*: système $\mapsto$ sys. à pile, spécification $\mapsto$ $\mathcal{P}$-automate, vérification: par Pre^*/Post^*/automate de Büchi (LTL)
- *Applications*: raisonnement sur piles, par ex. « bytecode »



9. Systèmes infinis

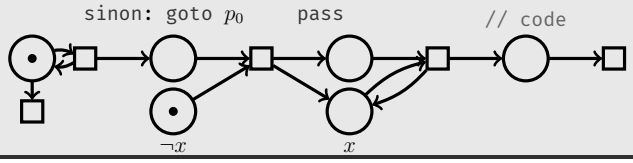
Réseaux de Petri

- *Déf.*: places et transitions reliées par arcs pondérés
- *Marquage*: nombre de jetons par place
- *Déclenchement*: si assez de jetons pour chaque arc entrant, les retirer, et en ajouter de nouveaux selon les arcs sortants
- *Successeurs*: $\text{Post}^*(\boldsymbol{m}) = \{\boldsymbol{m}' \in \mathbb{N}^P : \boldsymbol{m} \xrightarrow{*} \boldsymbol{m}'\}$
- *Prédecesseurs*: $\text{Pre}^*(\boldsymbol{m}') = \{\boldsymbol{m} \in \mathbb{N}^P : \boldsymbol{m} \xrightarrow{*} \boldsymbol{m}'\}$
- *Exemple*:



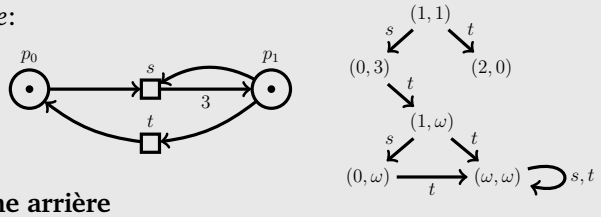
Modélisation

- *Processus*: comptés par les places
- *Exemple*: `si ¬x: x = vrai tant que ¬x: sinon: goto` p_0 `pass // code`



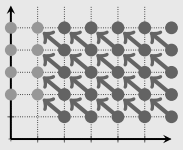
Graphes de couverture

- *Idée*: construire $\text{Post}^*(\boldsymbol{m})$ mais accélérer avec ω si $\boldsymbol{x} < \boldsymbol{x}'$
- *Test*: $\boldsymbol{m}'$ couvrable ssi le graphe contient un $\boldsymbol{m}'' \geq \boldsymbol{m}'$
- *Exemple*:



Algorithme arrière

- *Idée*: construire $\uparrow\text{Pre}^*(\uparrow\boldsymbol{m}')$ en déclenchant vers l'arrière
- *Représentation*: d'ensemble clos par le haut par base finie
- *Test*: $\boldsymbol{m}$ peut couvrir $\boldsymbol{m}'$ ssi découvert



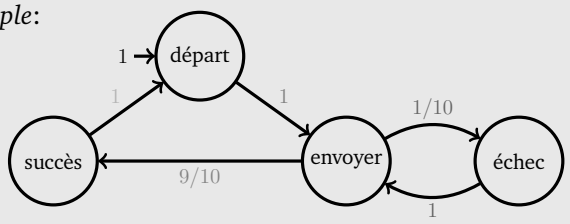
Accessibilité

- *Problème*: tester si $\boldsymbol{m}' \in \text{Post}^*(\boldsymbol{m})$
- *Décidable* mais plus compliqué

10. Systèmes probabilistes

Chaîne de Markov

- *But*: remplacer non-déterminisme par probabilités
- *Déf.*: struct. de Kripke avec proba. sur transitions / états init.
- *Représentation*: probabilités = matrice **P** et vecteur **init**
- *Exemple*:



- *Événements*: exéc. inf. décrites par préfixes finis (cylindres)
- *Probabilité*: somme du produit des transitions de cylindres
- *Exemple*: $\mathbb{P}(\mathsf{F}\,\text{succès}) = \sum_{i=0}^{\infty} 1 \cdot ((1/10) \cdot 1)^i \cdot (9/10) \cdot 1 = 1$
- *Outils*: PRISM/Storm (PCTL, analyse quantitative, etc.)

Accessibilité

- *Accessibilité*: événement de la forme $A \,\mathsf{U}\, B$
- *Partition*: $\underbrace{S_0 \coloneqq [\![\neg\exists(A \,\mathsf{U}\, B)]\!]}_{\text{prob. nulle}}$, $\underbrace{S_1 \coloneqq B}_{\text{prob. = 1}}$, $\underbrace{S_? \coloneqq S \setminus (S_0 \cup S_1)}_{\text{prob. à déterminer}}$
- *Approche*: $\mathbf{A} \coloneqq \mathbf{P}$ sur $S_?$; $\boldsymbol{b}(s) \coloneqq$ proba. d'aller de s vers S_1; $\boldsymbol{x}(s) = \mathbb{P}(s \models A \,\mathsf{U}\, B)$ est la solution de $(\mathbf{I} - \mathbf{A}) \cdot \boldsymbol{x} = \boldsymbol{b}$
- *Approx.*: $A \,\mathsf{U}^{\leq n}\, B$ obtenu par $f^n(\mathbf{0})$ où $f(\boldsymbol{y}) \coloneqq \mathbf{A} \cdot \boldsymbol{y} + \boldsymbol{b}$

Comportements limites

- *CFC terminales*: une est atteinte et parcourue avec proba. 1
- FG *et* GF: se calculent via accessibilité et CFC terminales

CTL probabiliste (PCTL)

- *Syntaxe*: comme CTL, mais $\exists$ / $\forall$ deviennent $\mathcal{P}_I$, et ajout $\mathsf{U}^{\leq n}$
- $\mathcal{P}_I(\varphi)$: proba. de φ dans intervalle I?
- $\mathsf{U}^{\leq n}$: côté droit satisfait en $\leq n$ étapes?
- *Vérification*: calcul récursif + éval. proba. d'accessibilité