10

DM d'introduction à la vérification: CTL

Master 1 Informatique 2020/2021

Pierre-Antoine Rouby

28 mars 2021

Exercice 1

$$\Phi = AF(b \wedge \neg a) \rightarrow AFA(b\ W\ (\neg a \wedge \neg b))$$

1. On sépare la formule en sous formules Ψ_i.

$$\Psi = ENF(\Phi) \qquad\qquad \Psi = \Psi_1 \rightarrow \Psi_2$$

$$\begin{aligned}\Psi_1 &= AF(b \wedge \neg a)\\ &= A(true \cup (b \wedge \neg a))\\ &= \neg EG\neg(b \wedge \neg a)\end{aligned}$$

$$\begin{aligned}\Psi_2 &= AFA(b\ W\ (\neg a \wedge \neg b))\\ &= \neg EG\neg A(b\ W\ (\neg a \wedge \neg b))\\ &= \neg EG\neg\Psi_3\\ &= \neg EG\neg\neg E(b \cup (\neg b \wedge a))\\ &= \neg EGE(b \cup (\neg b \wedge a))\end{aligned}$$

$$\begin{aligned}\Psi_3 &= A(b\ W\ (\neg a \wedge \neg b))\\ &= \neg E((b \wedge \neg(\neg a \wedge \neg b)) \cup (\neg b \wedge \neg(\neg a \wedge \neg b)))\\ &= \neg E(b \cup (\neg b \wedge a))\end{aligned}$$

$$\begin{aligned}\Psi &= \Psi_1 \rightarrow \Psi_2\\ &= (\neg\Psi_1) \vee \Psi_2\\ &= (\neg\neg EG\neg(b \wedge \neg a)) \vee (\neg EGE(b \cup (\neg b \wedge a)))\\ &= EG\neg(b \wedge \neg a) \vee (\neg EGE(b \cup (\neg b \wedge a)))\\ &= \neg(\neg EG\neg(b \wedge \neg a) \wedge EGE(b \cup (\neg b \wedge a)))\end{aligned}$$

✓

2. Pour plus de simplicité nous allons découper Ψ de la question précédente en sous formule.

$$\begin{aligned}Sat(\Psi) &= Sat(\neg(\neg EG\neg(b \wedge \neg a) \wedge EGE(b \cup (\neg b \wedge a))))\\ &= S \backslash Sat(\neg EG\Psi_1 \wedge EG\Psi_2)\\ &= S \backslash (S \backslash Sat(EG\Psi_1)) \cup (EG\Psi_2)\end{aligned}$$

$$
\begin{aligned}
Sat(\Psi_1) &= Sat(\neg(b \wedge \neg a)) \\
&= S \backslash Sat(b \wedge \neg a) \\
&= S \backslash \{s_3, s_2\} \\
&= \{s_0, s_1, s_4, s_5\}
\end{aligned}
\qquad
\begin{aligned}
Sat(\Psi_2) &= Sat(E(b \cup (\neg b \wedge a))) \\
&= Sat(E(\Psi_3 \cup \Psi_4))
\end{aligned}
$$

+ application de l'algo pour EU

$$
\begin{aligned}
Sat(\Psi_3) &= Sat(b) \\
&= \{s_0, s_2, s_3\}
\end{aligned}
\qquad
\begin{aligned}
Sat(\Psi_4) &= Sat(\neg b \wedge a) \\
&= \{s_5\}
\end{aligned}
$$

Maintenant que nous avons découpé en sous formule et résolue les formules trivial, nous allons pouvoir résoudre l'équation $Sat(\Psi)$ en utilisant les algorithmes vu en cours.
Les algorithme nous donne les états suivant :

$$
\begin{aligned}
Sat(\Psi_2) &= \{s_0, s_2, s_5\} \\
Sat(EG\Psi_2) &= \{s_0, s_2, s_5\} \\
Sat(\Psi) &= \{s_0, s_1, s_3, s_4\}
\end{aligned}
\quad \rightarrow \quad
\text{+ application algo}
$$

Exercice 2

1. $AFAXa \not\equiv FXa$.
   — $s \models AFAXa$: Pour tous successeur d'un états accesible s, on a dans à un moment un état s', ou pour tous successer issue de s' on satisfait a.
   — $\pi \models FXa$: On a dans le future de l'execution, un état s' pour le quelle Xa est satisfait.

   Figure 1, on peut voir un exemple de système de transition qui satisfait FXa mais pas $AFAXa$. En effet, il n'y à aucun état pour le quelle AXa est vrai.

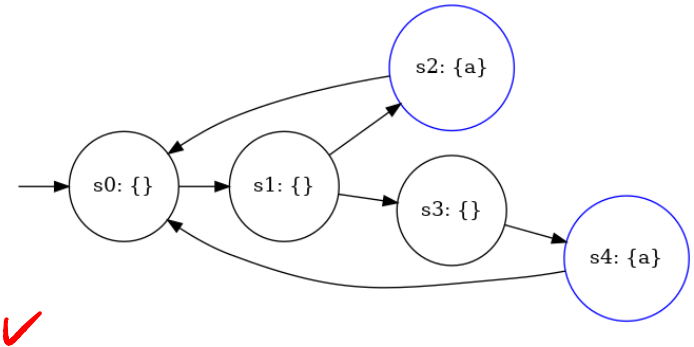

FIGURE 1 – ST_1

2. $AF(a \wedge AFb) \equiv F(a \wedge Fb)$.
   — $AF(a \wedge AFb)$: Pour tous sommet de s, un sommet atteignable, tous les chemins satisfont dans le future a et AFb. AFb signifie que tous les chemins à partire d'un s', qui satisfait $(a \wedge AFb)$, satisferons b dans le future.
   — $F(a \wedge Fb)$: π une execution partant de s, satisfait à un état $s' > s$, $(a \wedge Fb)$ et à un $s'' > s'$, b.

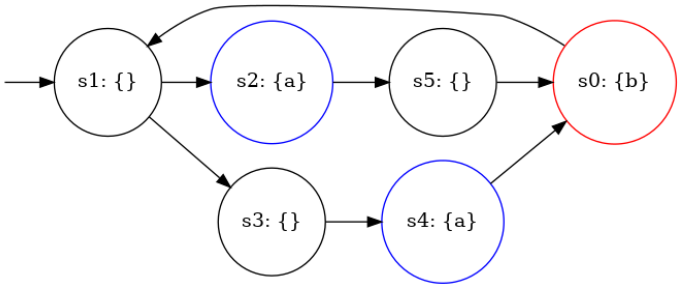

FIGURE 2 – ST_2

$$ST_2 \models AF(a \wedge AFb)$$

(=>)

Soit un système de transition $ST \models AF(a \wedge AFb)$ et π une execution quelconque de ST.

$$\pi = \underset{\emptyset}{s} \to \cdots \to \underset{a}{s'} \to \cdots \to \underset{b}{s''} \to \dots$$

Ici nous avons s'' qui satisfait b, donc par définition on peut dire que tous les prédécesseur de s'' satisfont Fb. s' satisfait a et comme il est prédécesseur de s'' il satisfait donc $(a \wedge Fb)$. Enfin comme s et prédécesseur de s', il satisfait aussi $F(a \wedge Fb)$. On peut donc en conclure que pour tous exécution π' de ST :

$$\pi' \models F(a \wedge Fb)$$

(<=)

Supposons un système de transition $ST \models AF(a \wedge AFb)$ et π un chemin quelconque de ST.

Supposons que $\pi \not\models F(a \wedge Fb)$, avec π une execution infini de la forme :

$$\pi = \underset{\emptyset}{s} \to \cdots \to \underset{a}{s'} \to \underset{a}{s'} \to \ldots$$

Ici l'exécution π ne satisfait jamais b, et ne satisfait donc pas $(a \wedge Fb)$. Or si un telle chemin existe dans ST, cela signifie qu'il existe un sommet satisfaisant $(a \wedge \neg Fb)$, or dans la formule CTL, il faut que tous les chemins $\models (a \wedge AFb)$ à un moment, mais si il existe un chemin qui ne satisfait pas Fb alors AFb est faut.

Il y a contradiction avec $ST \models AF(a \wedge AFb)$. Il ne peut donc pas exister d'execution qui ne satisfont pas $F(a \wedge Fb)$.

Les deux formules sont donc équivalente.

3. $EGAXa \not\equiv GXa$.
   - $EGAXa$: Il existe toujours une execution dont tous les ~~successeurs~~ satisfont a.
   - GXa : Pour toutes executions partant de l'états $i \in I$ on satisfait toujours Xa.

   Les deux formule ne sont pas équivalente, en effet on peut voir sur le système de transition Figure 3 avec le chemin $s_0 \to s_1^\omega$ satisfait la formule $EGAXa$ mais le chemin $s_0 \to s_2^\omega$ pas la formule GXa, donc toutes les executions de ne satisfont pas GXa.

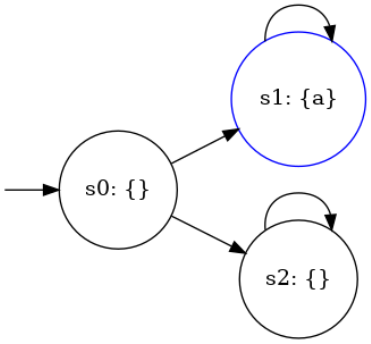

FIGURE 3 – ST_3

Ce n'est pas la réciproque que vous montrez ici. Vous devez supposer que $ST \not\models F(a \wedge Fb)$ et montrer que $ST \models AF(a \wedge AFb)$.

Exercice 3

1. On cherche à montré que $Sat(A(\Phi \cup \Psi))$ et équivalent à l'ensemble T.

$$T' = Sat(A(\Phi \cup \Psi)) \qquad T \supseteq Sat(\Psi) \qquad T = \{s \in Sat(\Phi) : post(s) \subseteq T\} \subseteq T$$

— $Sat(\Psi) \subseteq Sat(A(\Phi \cup \Psi))$ car tous les états qui satisfont Ψ, satisont aussi $A(\Phi \cup \Psi)$
— $Sat(\Phi)$ l'ensemble des états qui satisfont Φ
— $post(s) \subseteq T$ Vrai seulement si tous les successeurs de s sont dans T.
— $\{s \in Sat(\Phi) : post(s) \subseteq T\}$ Si on itère cette ensemble sur T (algorithme de point fix sur T), nous allons obtenir l'ensemble des états qui satisfont $A(\Phi \cup \Psi)$.

On cherche maintenant à montré que l'ensemble T est le plus petits ensemble qui satisfait $A(\Phi \cup Psi)$.

Supposons un système de transition ST, et un ensemble T qui satisfait $A(\Phi \cup \Psi)$, où $\Phi = a$ et $\Psi = b$.

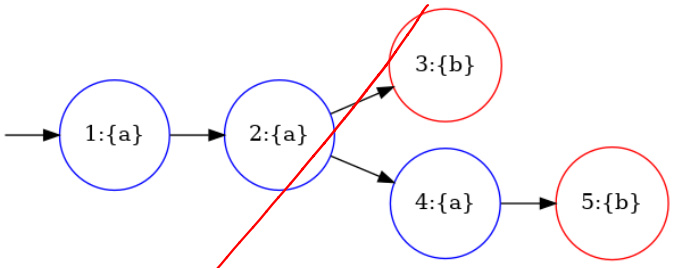

FIGURE 4 – ST_{31} : Exemple de systèmes qui satisfait $A(\Phi \cup \Psi)$

Ici, $Sat(A(\Phi \cup \Psi)) = T = \{1, 2, 3, 4, 5\}$, l'état 1 étant l'unique états initial on peut dire que $ST_{31} \models A(\Phi \cup \Psi)$.

On peut maintenant distinguer deux cas :

(a) On modifie un états s qui satisfait Φ en $s' \models \neg\Phi$.
Dans ce cas, nous nous retrouvons avec un chemin qui ne satisfait plus la formule, or la formule est de la forme AU et portant donc sur l'ensemble des executions. Alors le système de transition ne satisfait plus la formule.

(b) On modifie un états s qui satisfait Ψ en $s' \models \neg\Psi$.
Dans ce cas, l'on se retrouve avec un chemin qui ne se termine plus pas Ψ se qui rend $(\Phi \cup \Psi)$ non satisfait, or comme pour le cas précedent si un chemin ne satisfait plus la formule, le système de transition ne satisfait plus la formule.

Pour toutes modification d'états sur les sommets de l'ensemble T, nous nous changeons les propriétés du sysèteme, T est donc l'ensemble minimal pour $Sat(A(\Phi \cup \Psi))$.

2. On cherche à montré que $Sat(AG\Phi)$ est équivalent à l'ensemble T

$$T' = Sat(AG\Phi) \qquad T \subseteq \{s \in Sat(\Phi) : post(s) \subseteq T\}$$

— $AG\Phi$: Pour toute execution π, on doit toujours avoir Φ, on peut dire par definition de A :

$$\forall \pi \in ST, \pi \models G\Phi$$

Ce qui revient pas induction sur G, à :

$$\forall \pi \in ST, \forall s \in \pi, s \models \Phi \wedge XG\Phi$$

Ce que l'on peut écrire par :

$$\forall s \in V, s \in T \rightarrow (s \models \Phi \wedge post(s) \in T)$$

— $s \in Sat(\Phi)$: L'ensemble des états accessible qui satisfont Φ.
— $post(s) \subseteq T$: Vrai seulement si tous les successeurs de s sont dans T, or, pour qu'un s' soit dans T, il est faut que $s' \in Sat(\Phi) \wedge post(s) \in T$. On peut donc déduire par induction que :

$$\forall s \in V, s \in T \rightarrow (s \models \Phi \wedge post(s) \in T)$$

Les deux ensembles sont donc équivalent.
On cherche à montré maintenant que T est plus plus grand ensemble.
Supposons ST un système de transition qui satisfait $AG\Phi$, et T l'ensemble des états qui satisfont $AG\Phi$.
On va montré par contradiction que l'on ne peut pas ajouté d'état à T.
Soit s un états atteignable depuis un état initial, qui satisfait Φ, et supposons s' un succeseurs de s qui n'est pas dans T.

(a) Si $s' \not\models \Phi$, il y à contradiction avec $s \in T$ car s appartient à T seulement si tous ces successeur le sont aussi.
(b) Si $s' \models \Phi$ comme s' n'est pas dans T, cela signifie qu'il possède un succeseurs qui ne vérifie pas $AG\Phi$. Il y a donc contradiction avec la supposisons $s \in T$.

On ne peut donc pas rajouter de sommet dans T, il est dons le plus grand ensemble.

3. On cherche à montré que $Sat(A(\Phi W \Psi))$ est équivalent l'ensemble T

$$T' = Sat(A(\Phi\ W\ \Psi)) \qquad T \subseteq Sat(\Psi) \cup \{s \in Sat(\Phi) : post(s) \subseteq T\}$$

$Sat(A(\Phi W \Psi))$: Équivalent à $Sat(A(\Phi \cup \Psi))$, à la difference qu'ici, nous cherchons aussi à reconnaître les executions infinis qui satisfont $G\Phi$. Nous n'avons donc pas de garantis que Ψ soit un jour satisfait.
— $T \subseteq Sat(\Psi)$: L'ensemble des sommets satisfaisant Ψ.
— $\{s \in Sat(\Phi) : post(s) \subseteq T\}$: L'ensemble des sommets satisfaisant Φ et dont tous les sucesseurs sont dans T.
— L'union des deux ensemble suivant est bien équivalent au *weak until* car il comport tous les états satisfaisant Ψ et tous les cycles satisfaisant toujours Φ.

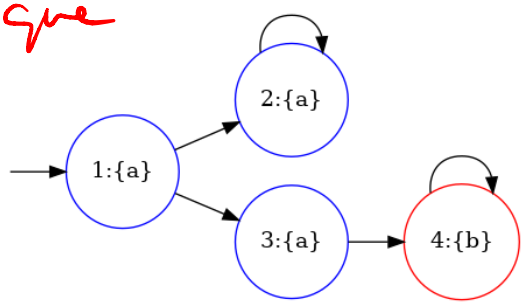

FIGURE 5 – ST_{33} : Exemple de systèmes qui satisfait $A(aWb)$

$$ENF(A(\Phi W \Psi)) = \neg E((\Phi \wedge \neg\Psi) \cup (\neg\Phi \wedge \neg\Psi))$$

L'algorithme pours calculer l'ensemble satisfesant EU nous garantie le plus petit ensemble T se sommets de S satisfaisant la formule. La négations de cette formule $S \backslash T$ et donc le plus grand ensemble qui ne satisfait pas la sous formule. Donc le plus grand ensemble qui satisfait $A(\Phi W \Psi)$.

Exercice 4

$$\begin{aligned}\Phi :=& a \\ |& \neg a \\ |& \Phi \wedge \Phi \\ |& E\varphi\end{aligned} \qquad \begin{aligned}\varphi :=& X\Phi \\ |& G\Phi \\ |& \Phi \cup \Phi\end{aligned}$$

Soit $\mathcal{S}_1, \mathcal{S}_2$:

$$\mathcal{S}_i = (S_i, Act, \rightarrow_i, I_i, AP, L_i)$$

manque : $a, \neg a, \wedge$

$\mathcal{S}_1 \subseteq \mathcal{S}_2$ si $S_1 \subseteq S_2, \rightarrow_1 \subseteq \rightarrow_2, I_1 = I_2$ et $L_1(s) = L_2(s), \forall s \in S_1$

1. L'ensemble des formules de $ECTL\Phi$ porte sur la possibilité d'existance d'un chemin, en effet il est interdit avec la grammaire de $ECTL$ les formules de la forme :
   — $\neg EGa$: Il n'existe pas d'execution satisfaisant toujours a.

   Ainsi les formules utilisant E sont formcement de la forme : Il existe une execution ...

   À partir de cette constatation, on peut montré que si $\mathcal{S}_1 \subseteq \mathcal{S}_2$, alors $\mathcal{S} \models \Phi$ implique que $\mathcal{S}_2 \models \Phi$.

   (✓) En effet, étant donnée que $\mathcal{S}_1$ et un sous ensemble de $\mathcal{S}_2$, tous le sommets de $\mathcal{S}_1$ sont aussi des sommets de $\mathcal{S}_2$, ainsi que les transitions. On peut donc dire que toutes execution de $\mathcal{S}_1$ est aussi une execution de $\mathcal{S}_2$. ✓

   $$\forall \pi_1 \in \mathcal{S}_1, \exists \pi_2 \in \mathcal{S}_2, \pi_1 = \pi_2$$

   Donc si il existe une execution $\pi_1 \models \Phi$, il existe focement $\pi_2 \models \Phi$, donc $\mathcal{S}_1 \models \Phi$ et $\mathcal{S}_2 \models \Phi$. → ok pour $E\varphi$
2. Le langage ECTL n'accepte pas les négations sur autre chose que des proposition atomique, à partir de la il est impossible d'exprimer tous une parti des formules CTL. Pour le montré nous allons prendre l'exemple de la formule CTL : AGa.
   — AGa : La sémentique de cette formule est : Pour toutes executions, on a toujours a.
   — $\neg EF\neg a$: La sémentique de cette formule est la même que précédemment, mais exprimé par ça négation : Il n'existe pas d'execution pour la quelle nous avons $\neg a$.

   Cette dernière formule n'est pas valide en $ECTL$, car nous avons une négation devant le EF. Plus généraliement, la proposition $A\Psi$, pour tous execution Ψ, peuvent s'exprimer par la négation de l'existance d'une execution inverse $\neg E\neg Psi$. Or avec $ECTL$ il n'est pas possible d'exprimer la négation de l'existance d'une execution.

   Il est donc impossible d'exprimer en $ECTL$ les formules du type "pour tous".

il ne suffit pas de dire que la syntaxe ne permet pas d'exprimer AGa

Annexes

Vérification partiel des résultats de $Sat(ENF(\Phi))$

Pour vérifier mes résultats trouver pour les algorithmes de Sat, j'ai implementer une partie des algorithmes. L'intégralité du code à été réaliser par mes soins. Il est disponible à l'addresse `https://parouby.fr/pub/M1/IV/SatCTL.el`.

Si dessous le code qui ma servie à vérifier mes résultats pour l'exercice 1 :

```
(defvar G-exo1 nil "DM CTL Graph exercice 1")

(setq G-exo1
      (mk-graph '("s_0" "s_1" "s_2" "s_3" "s_4" "s_5")
                '(("s_0" . "s_1")
                  ("s_1" . "s_0")
                  ("s_0" . "s_2")
                  ("s_0" . "s_3")
                  ("s_1" . "s_2")
                  ("s_2" . "s_5")
                  ("s_2" . "s_3")
                  ("s_5" . "s_2")
                  ("s_5" . "s_4")
                  ("s_4" . "s_3")
                  ("s_3" . "s_4")
                  ("s_3" . "s_3"))
                ;; blue == a
                ;; reb == b
                ;; purple == a /\ b
                '(("color" . (("s_0" . "purple")
                              ("s_2" . "red")
                              ("s_5" . "blue")
                              ("s_3" . "red"))))))

(defun png-digraph-exo1 ()
  (digraph-to-png G-exo1 "exo1"))

(defun sat-psi4 ()
  '("s_5"))

(defun sat-psi3 ()
  '("s_0" "s_2" "s_3"))

(defun sat-psi1 ()
  '("s_0" "s_1" "s_4" "s_5"))

(defun sat-psi ()
  (sat-neg (sat-and (sat-neg (sat-EG (sat-psi1)
                                     G-exo1)
                             G-exo1)
                    (sat-EG (sat-EU (sat-psi3) (sat-psi4)
                                    G-exo1)
                            G-exo1)
                    G-exo1)
           G-exo1))

;; Return result as LaTeX string
(format "$$Sat(\\Psi) = \\{ %s \\}$$"
        (string-join (sat-psi) ", "))
```

$$Sat(\Psi) = \{s_0, s_1, s_3, s_4\}$$

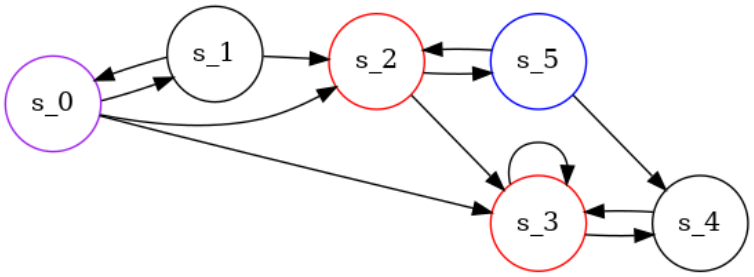

FIGURE 6 – Graphs Exercice 1