GB5 BIMB **IA symb. pour les réseaux biol. complexes** 6 décembre 2023

TD n^o 3 : Implémentation de l'algorithme naïf de Model-checking `CTL`

Le but de ce TD est d'implémenter l'algorithme naif de model checking d'une formule CTL en python, sur une structure de Kripke donnée. La fonction principale devra renvoyer la liste des états qui satisfont la formule.

Exercice 1 : (**Structures de données**)

1. Proposez une représentation des propositions atomiques des formules. Pour simplifier, on pourra considérer que les formules atomiques sont uniquement de la forme `a=1`.
2. Proposez une structure de données pour représenter une structure de Kripke. *Rappel :* une structure de Kripke est un triplet $(V, \rightarrow, L)$ où V est un ensemble d'états, $\rightarrow \subseteq S \times S$ et L est une fonction qui associe à chaque état l'ensemble des propositions atomiques qui sont satisfaites dans cet état.
3. Proposez une structure de données pour représenter une formule. Comme l'algorithme vu en cours étiquette chaque états des sous-formules de la formule d'intérêt, il est nécessaire de proposer une structure qui permettent d'accéder rapidement accéder aux sous-formules directes des sous formules.
   Ce sera la représentation interne de la formule. (On ne s'intéresse pas dans ce TD à proposer des outils permettant de *parser* une expression donnée au clavier sous forme de texte et d'en donner la représentation interne.)

Exercice 2 : (**Les fonctions manipulant les formules**)

1. Écrire la fonction `buildAtom` construisant la forme interne d'une proposition atomique : à partir de `"a"` et `1`, la fonction renverra la forme interne pour les propositions atomiques choisies à l'exercice précédent.
   De même écrire les fonctions `buildTrue()` et `buildFalse()`.
2. Écrire ensuite les constructeurs de formules non atomiques qui à partir d'une (resp. deux) formule φ (resp. deux formules φ et ψ), construisent la représentation interne des formules : $\neg\varphi$, `EX`φ, `AX`φ, `EF`φ, `AF`φ, `EG`φ, `AG`φ (resp. $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, `E[`φ`U`ψ`]`, `A[`φ`U`ψ`]`).
3. Écrire la fonction qui renvoie le "type" d'une formule : cette fonction doit dire si la formule passée en argument est la formule `True`, `False`, une proposition atomique, une formule donc le connecteur principal est un $\neg$, $\wedge$, $\vee$, $\Rightarrow$, `EX`, `AX`, `EF`, `AF`, `EG`, `AG`, `EU`, `AU`.
4. Écrire une fonction qui prend en argument une formule non atomique unaire (le connecteur principal est unaire) et qui renvoie la sous-formule principale.
5. De même écrire une fonction qui prend en argument une formule non atomique binaire (le connecteur principal est binaire) et qui renvoie les deux sous-formules filles.

Exercice 3 : (**Traitement du connecteur `EX`**)

Le but de cet exercice est d'écrire l'algorithme qui traite le cas du connecteur temporel `EX`.

1. Avant de se lancer dans la programmation, il faut réfléchir à la manière dont on va étiqueter chacun des états par les sous-formules de la formule donnée en argument.
2. Ecrire une fonction qui prend en entrée la structure de Kripke, un état `s` et une formule φ, et qui renvoie `True` si et seulement si l'état `s` satisfait `EX`φ, autrement dit s'il existe un successeur de `s` qui satisfait φ.
3. Dans cette question, on considère la formule `EX`φ, et on considère les labels L_φ et $L_{\texttt{EX}\varphi}$ de φ et `EX`φ. Ecrire une fonction qui prend en paramètre une structure de Kripke (dans laquelle on suppose que l'étiquetage de L_φ a déjà été fait), ainsi que les labels L_φ et $L_{\texttt{EX}\varphi}$ de φ et `EX`φ, et qui renvoie la structure de Kripke, dans laquelle les labels $L_{\texttt{EX}\varphi}$ ont été rajoutés.
4. En supposant que la fonction `SAT` est déjà écrite, écrire une fonction qui prend en entrée un structure de Kripke et une formule CTL, et qui renvoie l'ensemble des états où la formule `EX`φ est vraie. L'idée est toute simple : on appelle la fonction `SAT` pour étiqueter les états avec la formule φ, on calcule ensuite le label pour la formule `EX`φ, et enfin on appelle la fonction de la question précédente.

Exercice 4 : (**Traitement du connecteur temporel `AF`**)

Le but de cet exercice est d'écrire l'algorithme qui traite le cas du connecteur temporel `AF`.

1. Ecrire une fonction qui extrait la liste de tous les états d'une structure de Kripke.
2. Ecrire une fonction qui extrait la liste de tous les états d'une structure de Kripke qui sont étiqueté par un label passé en paramètre.
3. Ecrire une fonction qui teste si deux listes d'états sont différentes.
4. Ecrire une fonction qui prend en argument une structure de Kripke, un état `s` et un ensemble d'états `Y`, et qui renvoie `True` uniquement si tous les états successeurs de `s` sont dans `Y`.
5. Ecrire une fonction qui prend en entrée la structure de Kripke, un ensemble d'états `X` et le label $L_{\texttt{AF}\varphi}$ de $\texttt{AF}\varphi$ et qui ajoute aux labels des états de l'ensemble `X`, le label $L_{\texttt{AF}\varphi}$ (uniquement s'il n'y est pas déjà).
6. En supposant que la fonction `SAT` est déjà écrite, écrire une fonction qui prend en entrée un structure de Kripke et une formule CTL φ, et qui renvoie l'ensemble des états où la formule $\texttt{AF}\varphi$ est vraie. L'idée est de suivre l'algorithme vu en cours : on construit d'abord l'ensemble `X` de tous les états, on appelle la fonction `SAT` pour étiqueter les états avec la formule φ, on calcule ensuite l'ensemble des états satisfaisant φ. On implémente ensuite la boucle "Tant que" vue en cours. On calcul le label pour la formule $\texttt{AF}\varphi$, et enfin on appelle la fonction de la question précédente.

Exercice 5 : (**Traitement du connecteur temporel** `EU`)

Le but de cet exercice est d'écrire l'algorithme qui traite le cas du connecteur temporel `EU`.

1. Ecrire une fonction qui prend en argument une structure de Kripke, un état `s` et un ensemble d'états `Y`, et qui renvoie `True` uniquement s'il existe un état successeur de `s` qui est dans `Y`.
2. Ecrire une fonction qui prend en entrée la structure de Kripke, un ensemble d'états `X` et le label $L_{\texttt{E}[\varphi\texttt{U}\psi]}$ de $\texttt{E}[\varphi\texttt{U}\psi]$ et qui ajoute aux labels des états de l'ensemble `X`, le label $L_{\texttt{E}[\varphi\texttt{U}\psi]}$ (uniquement s'il n'y est pas déjà).
3. En supposant que la fonction `SAT` est déjà écrite, écrire une fonction qui prend en entrée un structure de Kripke et deux formules CTL φ et ψ, et qui renvoie la structure de Kripke complétée par les étiquetage par $L_{\texttt{E}[\varphi\texttt{U}\psi]}$. L'idée est de suivre l'algorithme vu en cours : on appelle la fonction `SAT` pour étiqueter les états avec la formule φ, on extrait ensuite l'ensemble `W` de tous les états satisfaisant φ, on construit l'ensemble `X` de tous les états, on appelle la fonction `SAT` pour étiqueter les états avec la formule ψ, on extrait ensuite l'ensemble `Y` de tous les états satisfaisant ψ. On implémente ensuite la boucle "Tant que" vue en cours. On calcule le label pour la formule $\texttt{E}[\varphi\texttt{U}\psi]$, et enfin on appelle la fonction de la question précédente.

Exercice 6 : (**Traitement général**)

1. Dans cette question, on considère la formule $\neg\varphi$, et on considère les labels L_φ et $L_{\neg\varphi}$ de φ et $\neg\varphi$. Ecrire une fonction qui prend en argument une structure de Kripke (dans laquelle on suppose que les lables L_φ ont déjà été insérés), et le label $L_{\neg\varphi}$ d'une formule $L_{\neg\varphi}$, et qui renvoie la structure de Kripke, dans laquelle les labels $L_{\neg\varphi}$ ont été rajoutés.
2. Dans cette question, on considère la formule $\varphi \wedge \psi$, et on considère les labels L_φ, L_ψ et $L_{\varphi\neg\psi}$ de φ, ψ et $\varphi \wedge \psi$. Ecrire une fonction qui prend en argument une structure de Kripke (dans laquelle on suppose que les labels L_φ et L_ψ ont déjà été insérés), et le label $L_{\varphi\wedge\psi}$ d'une formule $L_{\varphi\wedge\psi}$, et qui renvoie la structure de Kripke, dans laquelle les labels $L_{\varphi\wedge\psi}$ ont été rajoutés.
3. Même question pour la formule $\varphi \vee \psi$.
4. Il est maintenant possible d'implémenter l'algorithme global.

Exercice 7 : (**Application**)

Pour les deux graphes d'états ci-contre, faites tourner votre algorithme de model-checking pour les formules CTL suivantes :

1. $\texttt{AG}(a = 2)$
2. $\texttt{AG}(\texttt{EF}(a = 2))$
3. $((a = 0) \vee (a = 1)) \Rightarrow \texttt{AG}(\neg(a = 2))$
4. $\texttt{E}[((a = 0) \vee (a = 1))\ \texttt{U}\ (a = 2)]$