Modélisation et Vérification Probabiliste

Projet de TP

L'objectif du projet est de créer un model-checker probabiliste et statistique pour les chaînes de Markov et les Processus de Décision Markoviens, en Python.

Les objets d'entrée (MC et MDP) seront représentés dans un fichier texte, utilisant la grammaire suivante :

```
   Gram = States Actions Trans
States = ‘‘States’’ ID (‘‘,’’ ID)* ‘‘;’’
Actions = ‘‘Actions’’ IDact (‘‘,’’ IDact)* ‘‘;’’
Trans = (Transact | Transnoact) (Trans)*
Transact = ID ‘‘[‘‘ IDact ‘‘]’’ ‘‘->’’ Int ‘‘:’’ ID (‘‘+’’ Int ‘‘:’’ ID)* ‘‘;’’
Transnoact = ID ‘‘->’’ Int ‘‘:’’ (‘‘+’’ Int ‘‘:’’ ID)* ‘‘;’’
```

Dans cette grammaire, après un préambule déclarant les noms des états et des actions utilisés dans le modèle, on déclare les transitions une par une, en donnant pour chaque transition l'action qu'elle utilise (si elle en utilise une), et la liste des états cibles et les poids des arrêtes associées. Pour faciliter l'écriture et l'interprétation des modèles, on utilise ici des poids entiers permettant de définir les probabilités de chaque arrête (probabilité = poids de l'arrête divisé par la somme des poids).

1 Partie 1 : Lecture du modèle et simulation

La première étape consiste en la création de la structure de données associée aux modèles que nous allons interpréter.

Pour vous faciliter la tâche, une grammaire a été écrite et compilée en Python à l'aide de l'outil antlr4. Les fichiers sources sont disponibles à l'adresse suivante :

`https://uncloud.univ-nantes.fr/index.php/s/9Ec2cXfKxoxCp8y`

Dans le dossier MDP-parser, vous avez accès à un ensemble de fichiers pré-compilés :

— Le fichier `gram.g4` contient la définition de la grammaire. La compilation de cette grammaire se fait à l'aide d'antlr4, en utilisant la commande :
`antlr4 -Dlanguage=Python3 gram.g4`
Cette compilation produit les fichiers `gram.interp`, `gram.tokens`, `gramLexer.interp`, `gramLexer.py`, `gramLexer.tokens`, `gramListener.py` et `gramParser.py`. Il faudra importer ces fichiers pour pouvoir utliser le parser/lexer généré par antlr4.

— Le fichier `mdp.py` est un squelette que vous pouvez utiliser, si vous le souhaitez, pour votre projet. Ce fichier importe le lexer et le parser générés par antlr4 et définit un certain nombre d'actions à effectuer lors du parcours de l'arbre syntaxique généré par le modèle d'entrée. Le modèle d'entrée est passé par l'entrée standard en ligne de commande ou via un fichier texte en utilisant la commande suivante :
`python mdp.py < fichier`

— Le fichier `ex.mdp` est un exemple de fichier d'entrée respectant la grammaire définie ci-dessus.

L'objectif de cette première partie de projet est de produire un programme Python qui permet de parser des modèles de chaînes de Markov et de Processus de décision Markoviens définis à l'aide de la grammaire ci-dessus, de construire une structure de donnée correspondante et de simuler (exécuter) ces modèles. Proposer une sortie textuelle ou graphique, sous la forme que vous souhaitez, sera un plus.

Il sera aussi important de tester la correction des modèles que vous importez, par exemple :

— Les états et actions utilisées sont bien déclarés en préambule,

— Les chaînes de Markov n'utilisent pas d'actions,

— Les MDP ne mélangent pas de transitions sans actions et avec actions,

— ...

2 Partie 2 : Vérification des modèles

Dans cette deuxième partie, vous devrez implémenter des algorithmes de vérification de modèles.

1. Des algorithmes permettant de calculer la probabilité d'atteindre un état ou un ensemble d'états, pour les MC et les MDP. Dans les deux cas, ce calcul peut être fait de manière symbolique ou itérative (cf. cours).
2. Des algorithmes permettant de calculer l'espérance d'une fonction de récompense donnée. Vous devrez pour cela modifier la grammaire pour permettre aux modèles d'entrée de prendre en compte des récompenses d'états. Lors de la déclaration des états en préambule de modèle, on écrira les récompenses (entières) de la façon suivante :
   `States S0 : 1, S1 : 2 ...`
   Ces calculs pourront être faits de manière symbolique et/ou itérative.
3. Des algorithmes permettant de faire du model checking statistique sur les chaînes de Markov (propriétés quantitatives et qualitatives) pour des propriétés d'accessibilité. Vous pourrez étendre ces algorithmes pour calculer l'espérance de fonctions de récompenses.
4. Les différents algorithmes d'apprentissage par renforcement vus en cours, à appliquer sur des MDP afin d'optimiser la récompense obtenue lors d'une exécution.

Le projet sera évalué en séance le 05/01/2026, et un rapport (10p max, format pdf) sur votre projet sera à rendre pour le 16/01/2026 avec les sources du projet.