L3 informatique, 2015-2016, Projet de LFC

Langage de script pour l'utilisation d'une plateforme de gestion de systèmes multi-agents

1 Description générale

Nous voulons créer une plateforme permettant de tester des modèles de systèmes multi-agents. Pour cela, nous voulons créer un langage de script permettant de configurer un système multi-agents et d'agir sur celui-ci au cours de son évolution. Ce langage doit permettre, en particulier, de définir différents types d'agents, de créer, supprimer ou modifier des agents, d'arrêter ou faire reprendre l'évolution du système, d'enregistrer l'état du système, etc.
Vous devez écrire un compilateur qui va vérifier la lexicographie, la syntaxe et la sémantique d'un fichier source écrit dans le langage de script et, éventuellement, générer le programme java associé.

2 Modalités

Vous travaillerez en groupes de 3 ou 4 personnes.
Vous devrez rendre le projet par étapes (travaux à rendre toutes les 1 à 3 semaines selon difficulté).
Après chaque étape (sauf la dernière), il vous sera communiqué un corrigé de celle-ci afin que vous puissiez réaliser l'étape suivante sur une base correcte.
Lors des étapes où il faudra rendre un programme, vous devrez fournir un makefile et ne pas oublier de remettre tous les modules nécessaires à la compilation et l'exécution de votre programme (.h, .c, etc.)
Attention ! Afin de ne pas multiplier les plateformes et compilateurs nécessaires à l'évaluation de votre projet, il vous est demandé de faire en sorte que votre programme puisse être compilé et exécuté sur les machines des salles d'enseignement (système linux), en privilégiant les outils lex, yacc et gcc.

3 Description du langage de scripts

3.1 Définition des types d'agents

Dans un système multi-agents, on peut trouver plusieurs types d'agents, chacun étant caractérisé par un certain nombre d'attributs. Le langage de script doit donc permettre de définir les différents types d'agents qui vont constituer le système.

L'instruction permettant de définir un type d'agent est la suivante :

DEFAGENT(*nom_type ;* **liste_d_attributs)**

où

- DEFAGENT est un mot clé ;
- *nom_type* est un identificateur (une chaîne de caractères composée de lettres non accentuées, minuscules ou majuscules, de chiffres et de underscores, commençant obligatoirement par une lettre) ;
- **liste_d_attributs** est une suite de déclarations d'attributs (au moins une), séparées par des point-virgules, chaque déclaration étant de la forme

  nom_attribut : *type_attribut*

  où
  - *nom_attribut* est un identificateur
  - *type_attribut* est un type d'attribut parmi ceux autorisés par le langage (voir ci-dessous).

Les types d'attributs possibles sont :

- les types simples classiques que l'on trouve dans un programme java, soient INT, FLOAT, BOOLEAN, STRING, CHAR ;
- des types spécifiques au langage de script tels que les intervalles de nombres que l'on écrira sous la forme
    - INT(*min*,*max*) pour un intervalle de nombres entiers, *min* et *max* devant être des entiers avec *min* < *max* ;
    - FLOAT(*min*,*max*) pour un intervalle de nombres réels, *min* et *max* devant être des réels avec *min* < *max* ;

3.2 Création des agents

Une fois les types d'agents définis, il faut créer les agents qui vont évoluer dans le système.

L'instruction permettant de créer des agents est la suivante :

CREERAGENT(*type_agent* ; *nombre_agents* ; *type_création*)

où

- CREERAGENT est un mot clé ;
- *type_agent* est le type d'agent que l'on veut créer, ce type d'agent devant avoir été défini au préalable ;
- *nombre_agents* est un entier positif indiquant le nombre d'agents de ce type que l'on veut créer ;
- *type_création* est un mot clé parmi les 3 suivants indiquant la façon dont les valeurs d'attributs des nouveaux agents sont données. On a 3 types de création possibles qui sont :
    - RANDOM : dans ce cas, les valeurs d'attributs sont générées aléatoirement en tenant compte des types d'attributs dans la définition du type d'agent associé ;
    - MANUAL : dans ce cas, les valeurs d'attributs doivent être données dans l'instruction. Le mot MANUAL doit être suivi de plusieurs listes de valeurs (autant que d'agents à créer), chacune donnée entre parenthèses. Les valeurs de chaque liste doivent être séparées par des virgules.
      Exemple, pour 2 agents à créer avec 3 attributs entiers par agent :
      MANUAL(2, 56, 85)(5, 45, 82).
      Si on veut créer les nouveaux agents avec les mêmes valeurs d'attribut, on donnera une seule liste de valeurs entre parenthèses après le mot MANUAL et on insérera un # entre le mot MANUAL et la liste.
      Exemple, pour 2 agents à créer avec les mêmes 3 attributs entiers :
      MANUAL#(2, 56, 85).
    - TABLE : dans ce cas, le mot TABLE doit être suivi d'un nom de fichier, entre parenthèses, censé comporter les valeurs d'attributs des agents à créer.

RANDOM, MANUAL et TABLE sont des mots clés.

3.3 Suppression et modification des agents

L'instruction permettant de supprimer un ou plusieurs agents est la suivante :

SUPPRAGENT(*type_agent* ; *critère*)

où

- SUPPRAGENT est un mot clé ;
- *type_agent* est le type des agents que l'on veut supprimer, ce type d'agent devant avoir été défini au préalable ;
- *critère* est une expression booléenne (même écriture qu'en java) indiquant les tests à faire sur les attributs de l'agent pour savoir lesquels doivent être supprimés.

L'instruction permettant de modifier la valeur d'un attribut d'un ensemble d'agents est la suivante :
MODIFAGENT(*type_agent* ; *critère* ; modifications)
où

- MODIFAGENT est un mot clé ;
- *type_agent* est le type des agents que l'on veut modifier, ce type d'agent devant avoir été défini au préalable ;
- *critère* est une expression booléenne (même écriture qu'en java) permettant de sélectionner les agents dont les attributs vont être modifiés ;
- **modifications** est une liste d'opérations (au moins une) à effectuer sur les attributs des agents concernés. S'il y a plusieurs opérations à effectuer, celles-ci sont séparées les unes des autres par un point-virgule. Les opérations sont des affectations de la forme
  nom_attribut = **expression**
  où **expression** est une valeur ou une expression arithmétique faisant intervenir des noms d'attributs, des valeurs, éventuellement des parenthèses, et les opérateurs arithmétiques classiques (+ - * / ^).

3.4 Informations complémentaires

- On supposera que les mots clés doivent être écrits en majuscules et qu'un mot clé (en majuscules) ne peut pas être utilisé comme identificateur.
- Les espaces, tabulations, et retours à la ligne doivent être ignorés (acceptés n'importe où dans le programme source (sauf au milieu des mots) mais n'appartenant pas à une unité lexicale).
  Par exemple :
  DEFAGENT(passant ; nom : string ; age : INT(0,150))
  est aussi bien accepté que

```
DEFAGENT
        (passant        ;
        nom  :  string  ;
        age  :  INT (0 , 150 )    )
```

4 Etapes du projet

4.1 Analyse lexicale

La première chose à faire sera de recenser toutes les unités lexicales du langage de script (les catégories de mots que l'on peut trouver dans un programme source), ainsi que les règles d'écriture de ces unités lexicales (exemple : un nombre entier est une suite de chiffres).
Une fois que vous aurez établi la liste des unités lexicales et les expressions régulières permettant de caractériser les mots appartenant à chacune d'elles, vous pourrez générez un analyseur lexical permettant de vérifier que le programme source ne contient que des mots autorisés par le langage.

4.2 Analyse syntaxique

Il faudra ensuite écrire la grammaire générant les instructions du langage de script : définir les symboles terminaux et non terminaux, l'axiome, et les règles d'écriture des phrases du langage.
Grâce à la grammaire, vous pourrez générer un analyseur syntaxique qui permettra de vérifier qu'un programme écrit dans le langage de script est syntaxiquement correct.

4.3 Analyse sémantique et table des symboles

L'étape suivante sera la vérification sémantique des instructions (le nombre de valeurs d'attributs données lors de la création d'un agent correspond au nombre d'attributs de ce type d'agent, l'expression affectée à un attribut d'agent lors de sa modification est du bon type, un type d'agents a été défini avant la création d'agents de ce type, etc.).

Pour exécuter l'analyse sémantique, il faudra créer une structure (une table des symboles) et y stocker toutes les informations utiles (noms des types d'agents, nombre d'attributs de chaque type d'agents, noms et types de ceux-ci, etc.)

4.4 Génération du programme cible

Une fois la vérification lexicale, syntaxique et sémantique effectuées, la traduction du programme source en programme cible pourra commencer. Le programme java associé à un programme en langage de script sera composé de plusieurs fichiers (une classe par type d'agent). On devrait trouver la déclaration d'une classe par type d'agents avec les méthodes get et set pour chaque attribut et les méthodes de création et de modification des agents, etc.

5 1ère étape – A rendre au plus tard lundi 1er février à 13h

Faire la liste des unités lexicales du langage source et donner une description, la plus précise possible, de chacune.
Cette liste peut être donnée sous forme de tableau de la forme :

Unité lexicale	Nom	Description
Nombre entier	ENTIER	chaîne d'au moins 1 caractère constituée de chiffres entre 0 et 9

Votre travail devra être déposé sur plubel (un seul fichier par groupe) au plus tard le lundi 1er février à 13h.

- cours « L3 - Langages formels et compilation (I. Foucherot) »
- section « projet »
- lien « dépôt étape 1»