

Modélisation et spécification – Master 2 LC

TP 3 : CADP

Mihaela Sighireanu

(www.liafa.jussieu.fr/~sighirea/cours/modspec/)

CADP (www.inrialpes.fr/vasy/cadp) est un environnement de modélisation et vérification qui fournit, parmi d'autres outils, les moyens pour modéliser, spécifier et vérifier en utilisant les systèmes de transitions étiquetés (STE).

L'outil CADP est installé sur le serveur `cunillet` dans le répertoire `/usr/local/cadp`. Pour l'utiliser, il faut affecter des variables d'environnement suivantes (on utilise la syntaxe `bash`) :

```
# .bashrc
CADP=/usr/local/cadp
export CADP

PATH=$PATH:/usr/local/bin:$CADP/bin.macOS:$CADP/com
export PATH
```

1 Formats de description de STE

CADP offre plusieurs formats de description des STE : AUT et DOT sont des formats textuels, BCG est un format binaire.

Format aut La première ligne du fichier `.aut` spécifie le numéro de l'état initial du STE, le nombre de transitions et le nombre d'états. Les états sont numérotés à partir de 0, sans discontinuité. Après cette ligne sont énumérées toutes les transitions ; le format utilisé est état source, étiquette (chaîne de caractères en majuscules ou "i" pour l'action interne τ) et état cible.

Par exemple, un tampon à une place avec acquittement est décrit comme suit en format AUT :

```
des(0,4,4)
(0, "IN", 1)
(1, "OUT", 2)
(2, "ACKOUT", 3)
(3, "ACKIN", 0)
```

L'outil `bcg_io` permet de transformer ce format vers les autres formats pour sa visualisation ou son analyse. La visualisation des STE peut être faite :

- en utilisant le format BCG et l'outil `bcg_draw` ou
- en utilisant le format DOT et l'outil `GRAPHVIZ`.

Exercice 1 : Décrire puis visualiser les STE des exercices faits au TD précédent.

2 Composition des STE et SVL

Le langage de script SVL (Script Verification Language) permet de composer les STE décrits en différents formats avec les opérateurs de composition parallèle, de renommage ou d'abstraction.

Renommage des actions Il est effectué en utilisant la construction `rename`. On doit spécifier la liste des actions à renommer et leur nouveaux noms, ainsi que le STE sur lequel l'opération est faite. Le résultat peut être stocké dans un fichier au format souhaité (le format est indiqué par l'extension du fichier). Par exemple :

```
"bufack_1.aut" = rename OUT -> A, ACKOUT -> B in "bufack.aut" ;
```

Abstraction ou renommage des actions vers l'action interne τ ("i" pour CADP) est faite en utilisant l'opérateur `hide`. Par exemple, la construction ci-dessous permet de abstraire (rendre interne) les actions A et B du STE `bufack2.aut` :

```
"bufack2_i.aut" = hide A, B in "bufack2.aut" ;
```

Composition parallèle est un opérateur *binnaire* ayant différentes variantes en fonction du nombre des actions à synchroniser :

- Si on veut synchroniser sur toutes les actions (non-internes) des deux STE opérands, alors on utilise l'opérateur `||`.
- Si aucune synchronisation doit être faite, on utilise l'opérateur `|||`.
- Si on doit synchroniser les deux STE sur les actions d'une liste L , on utilise l'opérateur `|[L]|`.

Afin de obtenir le STE résultant de cette composition, il faut utiliser la construction `generation of`.

Par exemple, pour composer deux STE représentant chacun un tampon à une place avec acquittement afin d'obtenir un tampon à deux places, on utilise le script SVL suivant :

```
"bufack2.aut" = generation of ("bufack_1.aut" |[A,B]| "bufack_2.aut");
```

La composition parallèle est un opérateur binaire, mais elle peut se transformer en un opérateur n -aire en enchaînant les compositions parallèles. Par exemple, pour que l'action A soit effectué en même temps par trois STE, on écrira :

```
("ste1.aut" |[A]| "ste2.aut") |[A]| "ste3.aut"
```

Toutefois, si on souhaite avoir que des synchronisation binaires sur la même action, il faut bien séparer les STE qui sont synchronisés des STE qui le sont pas. Par exemple, la synchronisation entre les travailleurs de l'atelier de Milner et les outils est binaire :

```
("perso1.aut" ||| "perso2.aut") | [PUTM,GETM] | "marteau.aut"
```

Exercice 2 : Décrire la composition parallèle en présence de la tenaille.

Exercice 3 : Décrire la composition parallèle dans l'exemple de l'algorithme d'exclusion mutuelle de Dekker.

3 Minimisation et comparaison et de STE

CADP fournit plusieurs outils pour la minimisation et la comparaison des STE par des relations d'équivalence et de pré-ordre. Les relations vues en cours ont la syntaxe suivante :

- **trace** pour équivalence ou inclusion de traces,
- **weaktrace** pour équivalence ou inclusion de traces avec actions internes,
- **strong** pour bisimulation ou simulation forte,
- **observational** pour bisimulation ou simulation faibles.

La minimisation est faite par la construction **reduction** et elle doit spécifier la relation d'équivalence utilisée. Par exemple :

```
"bufack2_min.aut" = strong reduction of "bufack2.aut";
```

La comparaison est faite de façon similaire par la construction **comparison** qui doit spécifier le type de comparaison (équivalence, inclusion) et la relation à utiliser. Par exemple :

```
"diag_bufack.seq" = comparison using strong  
"bufack.aut" == "bufack2.aut";
```

teste l'équivalence des deux STE en utilisant la bisimulation forte; en cas de non satisfaction de la relation proposée (ici l'égalité), une séquence de contre-exemple est donnée dans le fichier `.seq`.

Exercice 4 : Montrer, en utilisant CADP, les différentes relation d'équivalence demandées dans le TD précédent.