Travaux Pratiques : N° 1 - Mars 2008
Modélisation des processus communicants / LOTOS-CADP

1 Présentation de CADP

CADP : *Construction and Analysis of Distributed Processes* (anciennement CAESAR/ALDEBARAN Development Package) voir `www.inrialpes.fr/vasy/cadp/` est une boîte à outils pour la conception et l'analyse de protocoles de communications et plus généralement de systèmes communicants distribués.
CADP permet la compilation, la vérification et la validation de programmes LOTOS. Elle contient de nombreux outils sous forme de modules (indépendants) de compilation ou de vérification.
La vérification dans CADP est essentiellement basée sur le *model checking*. C'est une technique qui consiste à explorer entièrement un modèle fini (un système de transitions étiquettées) en le confrontant avec des propriétés données.

2 Principe général d'utilisation de CADP

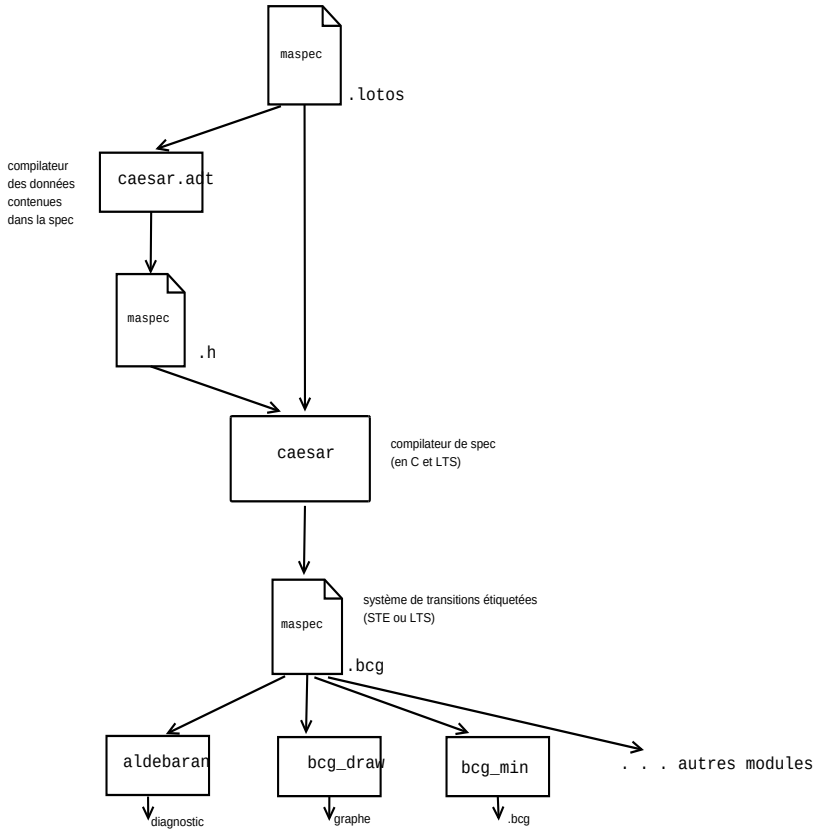

FIG. 1 – Principe d'usage de CADP

3 Aperçu de quelques modules de CADP

3.1 Modules de base

– **caesar.adt** : compile la partie donnée, lorsqu'elle existe dans la specification (library ... endlib), en un fichier .h qui est utilisé ensuite par le compilateur **caesar**

– **caesar** : compile toute la spécification LOTOS dont on donne le nom en paramètre.
Ce module (et commande) admet plusieurs options (telles que `-aldebaran, -bcg, -open`, etc)

3.2 Modules spécifiques

– **ALDEBARAN** : principal module de vérification ; il compare deux graphes, celui du modèle et celui de la propriété ou d'un programme, selon des relations indiquées (bissimulation forte, équivalence observationnelle, etc) ;
– **EXECUTOR** : permet une exécution aléatoire ;
– **GENERATOR, REDUCTOR** : permettent de construire le graphe des états accessibles ;
– **TERMINATOR** : recherche les états de blocage ;
– **BCG_MIN** : minimise le graphe par les techniques de bissimulation ;
– ...

3.3 Modules utilitaires

Tous ces modules prenent en entrée le fichier .bcg
– **BCG_DRAW** : produit une représentation graphique (2D) en postscript du graphe construit après la compilation d'une spécification ;
– **BCG_EDIT** : un éditeur interactif du graphe issu de la compilation d'une spécification ;
– **BCG_INFO** : produit des informations sur le graphe ;
– **BCG_IO** : transforme le .bcg en d'autres formats spécifiés comme options (...)
– **BCG_LABELS** : permet de modifier les étiquettes des transitions du graphe
– **BCG_LIB** : générateur de librairies dynamiques pour bcg
– **BCG_OPEN** : permet d'établir une passerelle entre bcg et l'environnement open/caesar.

4 Vérification de propriétés

On vérifie les propriétés d'un système (ou on analyse formellement un système)
– soit en utilisant ALDEBARAN,
– soit en utilisant le langage et le module XTL (*eXecutable Temporal Language*).

Absence de *deadlock* : L'absence de blocage (*deadlock*) est simplement vérifiée en utilisant le module/commande ALDEBARAN de la façon suivante :

```
aldebaran -dead maspec.bcg
```

ou bien en utilisant le module XTL de la façon suivante :

```
xtl mapropriete.xtl maspec.bcg
```

où `mapropriete.xtl` est un fichier contenant l'expression de la propriété. On utilise les options `-french, -verbose, ...`
En effet pour des propriétés spécifiques, ALDEBARAN peut paraître limité, on exprime alors les propriétés en logique temporelle, puis on les fournit comme paramètre à la procédure de vérification.

4.1 Expression de propriétés en XTL

Absence de deadlock

[true] <true> true

qui exprime que tout état à aumoins un successeur !.
On ne peut faire l'action bb, sans avoir fait l'action aa :

[(not "aa.*") . "bb.*"] false

On arrivera à effectuer l'action "mmm" :

<true*. "mmm"> true