GénéSyst : *Génération d'un Système de transitions étiquetées à partir d'une spécification* B *événementiel*

Xavier Morselli (xavier.morselli@imag.fr) Marie-Laure Potet (marie-laure.potet@imag.fr)
Nicolas Stouls (nicolas.stouls@imag.fr)*
LSR-IMAG - 681, rue de la Passerelle, BP72, 38402 St Martin d'Hères Cedex

Le 9 mars 2006

Résumé

La source d'erreur la plus couteuse et la plus délicate à détecter dans un développement formel est l'erreur de spécification. Ainsi la première phase d'un développement formel consiste généralement à représenter l'ensemble des comportements possibles sous la forme d'un automate. Partant de cette constatation, de nombreuses recherches portent sur la génération d'une machine B à partir d'une spécification telle qu'**UML**.

Cependant, il faut ensuite être capable de vérifier que la spécification respecte bien les comportements décrits. C'est pourquoi, nous avons réalisé un outil, GénéSyst, permettant d'extraire le contrôle d'un système B événementiel et de le représenter sous forme de système de transitions étiquetées. Le cas du raffinement est pris en compte et apparait sur l'automate produit sous la forme d'états hiérarchisés.

Cet outil est une implémentation des théories développées dans [SP04].

Mots clefs : Méthode B, spécification, raffinement, systèmes de transitions.

1 Fonctionnalités de GénéSyst

L'approche B événementiel permet de modéliser à la fois les données, leur traitement et la dynamique d'un système. Cette approche est basée sur la notion d'événements. Ceux-ci sont caractérisés par une garde (une condition de déclenchabilité) et une action. La modélisation événementielle introduit une difficulté supplémentaire : le raisonnement sur la dynamique du système. L'outil GénéSyst a pour but de permettre de visualiser cet aspect sous la forme de systèmes de transitions symboliques. Il permet de :

– représenter par un état un ensemble de valeurs (potentiellement infinis).
– calculer des transitions conditionnées représentant les hypothèses sous lesquelles un événement peut être déclenché (condition de déclenchabilité) dans un état donné et les hypothèses sous lesquelles un événement permet d'atteindre un état (condition d'atteignabilité).
– représenter le processus de raffinement par des systèmes de transitions hiérarchisés.

Le premier point permet de visualiser de manière lisible des systèmes manipulant des données appartenant à des domaines infinis ou grands.

---

[0]Article publié dans les actes de la conférence AFADL'04 (http ://lifc.univ-fcomte.fr/afadl2004/)

[*]Ce travail est supporté par une bourse BDI confinancée par le CNRS et ST Microelectronics.

Le second point permet de préciser au mieux les conditions sous lesquelles une transition peut être activée. En particulier si elle n'est pas possible, toujours possible ou conditionnée.

Enfin, la représentation des raffinements par des systèmes de transitions hiérarchisés permet de visualiser le lien entre deux niveaux de raffinement, en précisant la structuration du système de transitions abstrait. Le système de transitions obtenu est complet et correct vis-à-vis de la spécification : il admet exactement les mêmes traces que le système B événementiel [SP04].

Ces travaux ont été initialement introduits par D. Bert et F. Cave [BC00]. Ils ont ensuite été étendus par S. Hamdane [Ham02], avant d'être complétés par N. Stouls et M-L Potet [SP04].

La figure 1 décrit un parking ayant un nombre de place limité ($NbPlaces$) et dans lequel les véhicules peuvent $entrer$ ou $sortir$. Un contrôleur, pour le moment inactif, réagit à chacun de ces 2 stimulis externes. Sur les systèmes de transitions étiquetées produits, une condition est notée [ ] si elle est réductible à true et est notée $[G]$ sinon. Ainsi, Les transitions $controler\_entree$ et $controler\_sortie$ ne sont pas conditionnées car cc est instanciée. En revanche, $entrer$ et $sortir$ dépendent de $NbVoit$ et sont donc conditionnées.

MACHINE $parking$
CONSTANTS $NbPlaces$
PROPERTIES $NbPlaces \in \mathsf{NAT} \wedge NbPlaces > 0$
VARIABLES $NbVoit, cc$
INVARIANT $(NbVoit \in 0..NbPlaces)$
$\wedge (cc \in -1..1)$
$\wedge (cc = -1 \Rightarrow NbVoit < NbPlaces)$
$\wedge (cc = 1 \Rightarrow NbVoit > 0)$
INITIALISATION $NbVoit := 0 \parallel cc := 0$
OPERATIONS
entrer $\widehat{=}$ SELECT $NbVoit < NbPlaces \wedge (cc = 0)$ THEN $NbVoit := NbVoit + 1 \parallel cc := 1$ END ;
controler_entree $\widehat{=}$ SELECT $cc = 1$ THEN $cc := 0$ END ;
sortir $\widehat{=}$ SELECT $NbVoit > 0 \wedge (cc = 0)$ THEN $NbVoit := NbVoit - 1 \parallel cc := -1$ END ;
controler_sortie $\widehat{=}$ SELECT $cc = -1$ THEN $cc := 0$ END
END

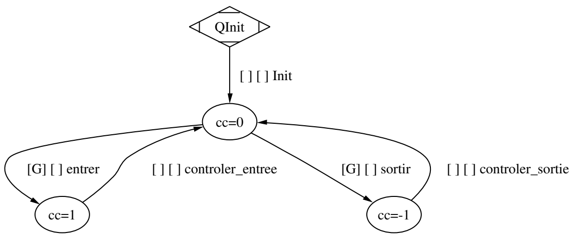

Figure 1: Exemple d'un parking avec contrôleur (inactif) et son système de transitions associé.

La figure 2, décrit un raffinement du parking de la figure 1, dans lequel un feu d'entrée a été introduit. La gestion de celui-ci est effectuée par le contrôleur.

REFINEMENT $parking\_r1$
REFINES $parking$
SETS $Couleur\_feu = \{vert, rouge\}$
VARIABLES $feu, NbVoit, cc$
INVARIANT $feu \in Couleur\_feu$
$\wedge ((cc = 0 \wedge feu = vert) \Rightarrow NbVoit < NbPlaces)$
$\wedge ((cc = 0 \wedge feu = rouge) \Rightarrow NbVoit = NbPlaces)$
$\wedge (cc = 1 \Rightarrow feu = vert)$
$\wedge (cc = -1 \Rightarrow NbVoit < NbPlaces)$
$\wedge ((cc = -1 \wedge feu = rouge) \Rightarrow NbVoit = NbPlaces - 1)$
$\wedge ((cc = -1 \wedge feu = vert) \Rightarrow NbVoit < NbPlaces - 1)$
INITIALISATION $NbVoit := 0 \| feu := vert \| cc := 0$
OPERATIONS
entrer $\widehat{=}$ SELECT $cc = 0 \wedge feu = vert$ THEN $NbVoit := NbVoit + 1 \parallel cc := 1$ END ;
controler_entree $\widehat{=}$ SELECT $cc = 1$ THEN IF $NbVoit = NbPlaces$ THEN $feu := rouge$ END $\parallel cc := 0$ END ;
sortir $\widehat{=}$ SELECT $NbVoit > 0 \wedge (cc = 0)$ THEN $NbVoit := NbVoit - 1 \parallel cc := -1$ END ;
controler_sortie $\widehat{=}$ SELECT $cc = -1$ THEN IF $feu = rouge$ THEN $feu := vert$ END $\parallel cc := 0$ END
END

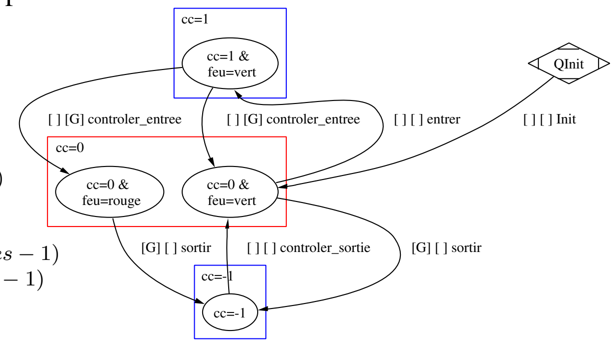

Figure 2: Exemple du parking raffiné et son système de transitions associé

2 Réalisation

GénéSyst prend en entrée un système B décrit par une machine ou un raffinement. L'utilisateur fournit les états du système de transitions sous la forme d'une disjonction donnant les prédicats associés à ces états. Cette disjonction est fournie par l'intermédiaire de la clause ASSERTIONS. L'obligation de preuve associée à cette clause va garantir que les états représentent toutes les valeurs possibles de l'invariant.

Le résultat est fourni sous la forme d'un fichier représentant le système de transitions. Ce fichier, qualifié de *format intermédiaire*, est une représentation textuelle du système de transitions produit. C'est à partir de celui-ci que GénéSyst génère des systèmes de transitions dans différents formats exploitables par d'autres outils (pour l'instant, seuls les formats *DOT* et *BCG* sont supportés).

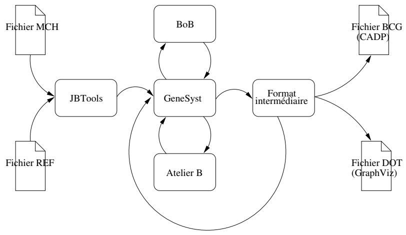

Figure 3: Environnement de GénéSyst

GénéSyst fonctionne en mode automatique. Pour chaque événement il cherche à prouver si l'événement est toujours déclenchable ou jamais déclenchable. Si une de ces preuves abouties, alors on a la réponse, sinon la transition est visualisée par défaut. Ceci ramène donc le problème de la recherche des conditions à un problème de preuve. Le même type d'approche est utilisée pour l'atteignabilité. L'approche proposée et sa correction est décrite dans [SP04].

GénéSyst est réalisé en Java en utilisant les outils suivants :

- le parseur JBTools [VTH02] qui permet d'analyser les spécifications.
- la **BoB** (boite à Outils B du LSR) qui permet de produire les obligations de preuve par calcul de plus faible pré-condition.
- le prouveur de l'*AtelierB* utilisé actuellement en mode automatique.

3 Conclusions et perspective

GénéSyst peut être utilisé en phase de mise au point d'une spécification ou d'un développement. Dans ce cadre, le système de transitions peut être modifié sans être remis en cause totalement. GénéSyst peut donc aussi utiliser en entrée une description (totale ou partielle) d'un système de transitions. Dans ce cas son travail consiste à vérifier et corriger ce système de transitions. L'efficacité de l'outil est améliorée puisque le système de transitions permet de guider les preuves à faire. De plus, l'utilisateur peut ainsi simplifier les conditions sur les transitions et GénéSyst produira les obligations de preuve assurant leur correction. Cette extension est en cours.

Une autre amélioration consiste à découpler l'activité de preuve de l'outil. Ceci permet soit d'affiner les preuves de manière interactive avec le prouveur de l'*AtelierB* soit d'interfacer l'outil avec un autre démonstrateur. La qualité du résultat peut ainsi être améliorée puisque certaines transitions visualisées correspondent à un défaut de preuve automatique.

D'un point de vue méthodologique, l'introduction d'états hiérarchisés permet de décrire les transitions à différents niveaux (entre les sur-états ou entre les sous-états). Ceci nécessite d'élaborer des heuristiques permettant de faire ces choix de visualisation. Enfin, une interface graphique est en cours d'élaboration. Le but est de pouvoir zoomer sur certaines parties du système de transitions, par exemple pour voir le détail d'une transition ou d'un état.

GénéSyst est diffusée sur le site du LSR à l'adresse :

http ://www-lsr.imag.fr/Les.Personnes/Nicolas.Stouls/

References

[BC00] D. Bert and F. Cave. Construction of Finite Labelled Transition Systems from B Abstract Systems. In *Integrated Formal Methods*, volume 1945 of *LNCS*. Springer-Verlag, 2000.

[Ham02] Smaine Hamdane. Génération de systèmes de transition étiquetés à partir de la description d'un système d'évènements décrits avec le langage B. Rapport de maîtrise, Université Joseph Fourrier, Grenoble-1, France, mai 2002.

[SP04] N. Stouls and M.-L. Potet. Explicitation du contrôle de développements B événementiel. In *AFADL'04*, LNCS. Springer-Verlag, 2004.

[VTH02] J.C. Voisinet, B. Tatibouet, and A. Hammad. jBTools : An experimental platform for the formal B method. In *PPPJ'02*, pages 137–140. Trinity College, Dublin, Ireland, Juin 2002.