Utilisation conjointe de B et TLA+ pour la modélisation et la vérification des systèmes réactifs

Olfa MOSBAHI
Faculté des Sciences de Tunis, LORIA-INPL
olfa.mosbahi@fst.rnu.tn

Leila JEMNI BEN AYED
Faculté des sciences de Tunis
leila.jemni@fsegt.rnu.tn

Résumé

La méthode B fournit un cadre rigoureux de développement de systèmes mais sa limitation concerne le type des propriétés exprimées car seuls les invariants sont considérés. Notre travail a pour objectif d'utiliser le B événementiel pour exprimer des propriétés temporelles de fatalité et d'équité. La logique temporelle des actions TLA+ a prouvé son efficacité dans l'expression et la vérification de propriétés d'équité. Elle se base sur le concept de raffinement, d'action et de transition qui exprime une compatibilité avec une modélisation B événementiel. Notre contribution consiste à proposer une méthode de spécification et de vérification utilisant conjointement B et TLA+ et leur outils de vérification l'AtelierB et le prouveur de théorèmes Isabelle.

Mots-clés : Modélisation, Preuve, Propriétés d'équité, Propriété de fatalité, Méthode B, Raffinement, Systèmes réactifs, Vérification.

1 Introduction

Dans le cadre de notre étude, nous nous intéressons aux systèmes de contrôle commande sûrs de fonctionnement et à leur modélisation au moyen de l'approche par événements de B (B événementiel)[2]. Comme un des aspects les plus importants à traiter dans le cas de la modélisation des systèmes réactifs est le respect des contraintes temporelles, plusieurs travaux de recherche se sont intéressés à étendre le B événementiel pour traiter les propriétés temporelles. Jean Raymond Abrial et Louis Mussat ont proposé dans [3] d'introduire dans les spécifications la description de contraintes dynamiques permettant d'exprimer des propriétés qui ne peuvent être décrites en terme d'invariants. Le travail de Jacques Julliand [6] a consisté en l'ajout de notations PLTL dans un module et en la coopération vérification par preuve de théorèmes et par model checking. Dans notre travail, on s'intéresse à l'utilisation de la méthode B événementiel pour traiter le temps et pour assurer l'équité dans les systèmes réactifs. Les invariants de la machine B expriment les propriétés de sûreté ou d'invariance mais les propriétés de fatalité et d'équité ne peuvent pas être exprimées au niveau de ces invariants. La méthode B concerne des transitions et son extension pour prendre en compte la notion d'équité requiert l'intégration d'une sémantique fondée sur les traces d'exécution. Dans ce contexte, nous nous intéressons à la description, avec le B événementiel, de propriétés de fatalité et d'équité en étendant la modélisation B par l'ajout de clauses contenant une description de propriétés en utilisant la notation de la logique temporelle des actions TLA+ disposant d'un prouveur de théorèmes qu'on peut utiliser pour prouver des propriétés temporelles à partir d'une spécification transformée de B vers TLA+. Dans la section 2, nous allons commencer par donner quelques notions sur la méthode B événementiel. Dans la section 3, nous présentons la logique temporelle des actions TLA+. La section 4 présentera l'utilisation conjointe du B événementiel avec la notation de TLA+ [8, 9]. Nous illustrons par la suite notre méthode sur le cas d'un système réalisant un $time\_out$.

2 Apercu sur la méthode B événementiel

La méthode B évenementiel, permet la spécification de systèmes réactifs [1, 2, 3]. Cette notion d'événements est proche des actions de Back [5, 4] ou des commandes de Dijkstra [7]. Ces systèmes abstraits peuvent être vus comme des systèmes fermés, qui modélisent le système d'intérêt et son environnement, et dont l'état peut évoluer par l'application des événements. Ceux-ci sont décrits en terme d'actions gardées, et ils sont susceptibles d'être déclenchés quand leur garde devient vraie. L'un des événements déclenchables peut alors être appliqué et l'état du

système change en fonction de l'action associée à l'événement.

3 Apercu sur la logique temporelle des actions TLA+

La logique temporelle des actions est due à Lamport [8, 9] et définit un cadre pour exprimer des propriétés temporelles de systèmes comme les propriétés de sûreté et les propriétés de fatalité avec des hypothèses d'équité. Une spécification TLA est une formule TLA de la forme :

$$\begin{array}{ll} \bigwedge & Init \\ \bigwedge & \Box[Next]_x \\ \bigwedge\bigwedge & SF_x(A)\ avecA\ \in A_{SF} \\ \bigwedge\bigwedge & WF_x(A)\ avecA\ \in A_{WF} \end{array}$$

Où Init spécifie les états initiaux des traces, $\Box[Next]_x$ décrit les traces possibles d'exécution, A_{SF} spécifie les actions exécutées sous hypothèse d'équité forte pour les actions de Next concernées et A_{WF} spécifie les actions exécutées sous hypothèse d'équité faible pour les actions de Next concernées. Le raffinement s'identifie à l'implication logique.

Les deux langages B et TLA+ sont par nature différents mais ils se basent sur la notion des systèmes de transition. Un module TLA+ et une machine B comportent des éléments faciles à traduire de B vers TLA+ ou de TLA+ vers B à l'exception des aspects liés aux propriétés sur les traces qui ne peuvent pas être prises en compte dans B.

4 Utilisation conjointe de B et TLA+ pour la modélisation et la vérification des propriétés de fatalité et d'équité

Les propriétés exprimées dans B sont celles d'invariance, alors que TLA+ permet l'expression de l'invariance, la fatalité et l'équité. D'un autre coté, B fournit un cadre rigoureux de développement mais seuls les invariants sont considérés. L'objectif de cette section est d'intégrer l'aspect temporel dans la méthode B, sans changer sa notation. Nous proposons une approche regroupant le B événementiel et la logique temporelle TLA+ permettant de spécifier et de vérifier différentes propriétés. Une spécification TLA+ est compatible avec une spécification B puisque les deux méthodes se basent sur un système de transition et sur le raffinement. La méthode que nous proposons consiste à :

1. Modéliser le système avec le B événementiel en ne considérant que les propriétés de sûreté et d'invariance,
2. Valider le modèle avec l'Atelier B,

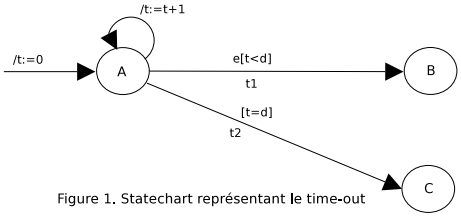

Figure 1. Statechart représentant le time-out

3. Ajouter dans le modèle raffiné des propriétés d'équité dans la clause FAIRNESS et de fatalité dans la clause EVENTUALITY au niveau des machines abstraites pour obtenir des machines abstraites temporelles,
4. Transformer le modèle obtenu (dernier raffinement) en une spécification TLA+.
5. Vérification des propriétés exprimées au niveau des deux clauses FAIRNESS et EVENTUALITY en utilisant le prouveur de théorèmes ISABELLE.

Le système obtenu est finalement validé et toutes les propriétés sont vérifiées conjointement par l'AtelierB et le prouveur Isabelle.

5 Etude de cas : le time_out

Le timeout est un système réactif qui peut être dans un des trois états A,B ou C. Une fois le système est à l'état A, si un signal de l'environnement apparaît dans un intervalle de dd unités de temps alors le système passe dans l'état B. Si rien ne s'est passé à l'issue de la période dd, le système passe à l'état C. Le système est représenté par un statechart à la figure 1.

5.1 Etape 1 : Modèle du système en B

Nous appelons Etats l'ensemble des trois états du système AA, BB, CC. dd est une constante qui désigne la durée maximale à passer dans l'état AA. Les variables $etat_sys, tt, evt$ désignent respectivement l'état du système, le compte du temps, l'occurrence d'un événement et l'occurrence d'un événement evt.

MACHINE *time_out*
SETS *Etat_sys* = {*AA,BB, CC*} ;
CONSANTS *dd*
PROPERTIES *dd*= 10
VARIABLES *etat_sys, tt, ee*
INVARIANT *etat_sys* ∈ *Etat_sys* ∧ *tt*∈ NATURAL ∧ *ee*∈ BOOL∧ *tt* ≤ *dd*
INITIALISATION
etat_sys :=*AA* || *tt* :=*0* || *ee* :=*FALSE*
OPERATIONS
AtoA ≙ **SELECT** *etat_sys=AA* ∧ *ee*= FLASE ∧ tt<dd
THEN *etat_sys* :=*AA* || *tt* := *tt+1* **END** ;
AtoB ≙ **SELECT** *etat_sys=AA* ∧ *ee*= TRUE ∧ tt<dd
THEN *etat_sys* :=*BB* **END** ;
AtoC ≙ **SELECT** *etat_sys=AA* ∧ *tt<dd*
THEN *etat_sys* :=*CC* **END** ;
Change ≙ **SELECT** *ee=FALSE* ∧ *etat_sys = AA*
THEN *ee* :=*TRUE* **END** ;
END

L'événement "AtoA" exprime les conditions sous lesquelles le système reste dans l'état AA (pas de réception de l'événement "evt" et la valeur de tt n' a pas atteint dd unités de temps). L'événement "AtoB" exprime la condition de transit de l'état AA vers BB sous l'occurrence de l'événement "evt". L'événement "AtoC" exprime la condition de transit de l'état AA vers CC. L'événement "*occur*$_e$*vt*" permet d'avoir une valeur aléatoire pour l'événement "evt".

5.2 Etape 2 : Preuves

Le modèle a été prouvé par le prouveur automatique de l'AtelierB et les invariants sont préservés par les événements.

5.3 Etape 3 : Adjonction de clauses

Une spécification B décrit un système par un ensemble d'événements et par un invariant qui doit être préservé par les événements. Une extension de B qui intégrerait des aspects liés à la fatalité et à l'équité pourrait être obtenue sur le plan syntaxique par l'ajout de clauses spécifiques. Deux clauses ont été ajoutées :

1. Une clause FAIRNESS qui décrit l'équité : les événements AtoB et AtoC sont exécutées sous hypothèse d'équité faible.

2. Une clause EVENTUALITY qui décrit les propriétés de fatalité.

MACHINE *time_out*
SETS *Etat_sys* = {*AA,BB, CC*} ;
CONSANTS *dd*
PROPERTIES *dd*= 10
VARIABLES *etat_sys, tt, ee*
INVARIANT *etat_sys* ∈ *Etat_sys* ∧ *tt*∈ NATURAL ∧ *ee*∈ BOOL∧ *tt* ≤ *dd*
INITIALISATION
etat_sys :=*AA* || *tt* :=*0* || *ee* :=*FALSE*
OPERATIONS
AtoA ≙ **SELECT** *etat_sys=AA* ∧ *ee*= FLASE ∧ tt<dd
THEN *etat_sys* :=*AA* || *tt* := *tt+1* **END** ;
AtoB ≙ **SELECT** *etat_sys=AA* ∧ *ee*= TRUE ∧ tt<dd
THEN *etat_sys* :=*BB* **END** ;
AtoC ≙ **SELECT** *etat_sys=AA* ∧ *tt<dd*
THEN *etat_sys* :=*CC* **END** ;
Change ≙ **SELECT** *ee=FALSE* ∧ *etat_sys = AA*
THEN *ee* :=*TRUE* **END** ;
FAIRNESS
WF(AtoB)
WF(AtoC)
EVENTUALITY
etat_sys = AA ∧ *ee = TRUE* ∧ *tt < dd* ⇝ *etat_sys = BB*
etat_sys = AA ∧ *tt = dd* ⇝ *etat_sys = CC*
END

5.4 Etape 4 : Transformation du modèle B en un modèle TLA+

MODULE *time_out*

PARAMETERS
dd : CONSTANTS
etat_sys , tt, evt : VARIABLES
TypeInvariant = ∧ **tt** ∈ **NATURAL** ∧ **dd = 10**
∧ **etat_sys** ∈ **{AA, BB, CC}** ∧ **ee** ∈ **{TRUE, FALSE}**
∧ **tt** ≤ **dd**
PREDICATE
Init = ∧ **etat_sys = AA** ∧ **tt = 0** ∧ **ee = FALSE**
ACTIONS
AtoA = ∧ **etat_sys = AA** ∧ **tt < dd** ∧ **ee = FALSE**
∧ $UNCHANGED_{<etat\_sys>}$ ∧ **tt' = tt + 1**
AtoB = ∧ **etat_sys = AA** ∧ **tt < dd** ∧ **ee = TRUE**
∧ **etat_sys' = BB**
AtoC = ∧ **etat_sys = AA** ∧ **tt = dd** ∧ **etat_sys' = CC**
change = ∧ **etat_sys = AA** ∧ **ee = FALSE** ∧ **ee' = TRUE**
Next = AtoA ∨ **AtoB** ∨ **AtoC** ∨ **change**
TEMPORAL
Specification General = ∧ **Init**
∧ □ $[Next]_{<etat\_sys,tt,ee>}$ ∧ $WF_{<etat\_sys,ee,tt>}$**(Next)**

Pour pouvoir vérifier les propriétés d'équité et de fatalité, nous devons transformer la spécification B en une spécification TLA+. Un module TLA+ décrit les paramètres, les prédicats

et les actions de la spécification. La traduction d'une spécification B en une spécification TLA+ est triviale, les événements B sont des actions TLA+ et les invariants B sont des invariants en TLA+.

5.5 Etape 5 : Vérification du modèle TLA+ obtenu par le prouveur de théorèmes Isabelle

Avant d'utiliser le prouveur de théorèmes général Isabelle pour vérifier les propriétés d'équité et de fatalité, nous devons transformer le modèle TLA vers un modèle compréhensible par Isababelle utilisant le codage de TLA en Isabelle. Le modèle ci-dessous est modèle TLA en Isabelle.

time_out.thy

Timeout = TLA+
datatype *etat_sys = AA || BB || CC*
constdefs *dd : : nat "dd==10"*
constdefs *etat_sys : : Etat_sys stfun*
tt : : nat stfun ee : : stpred
rules *base "basevars (etat_sys, tt, ee)"*
constdefs *InitTimeout : : stpred*
"InitTimeout == PRED etat_sys = #AA & tt=#0 & ee = #False"
AtoA : : action
"AtoA == ACT $ etat_sys = #AA & $ tt < #dd & $ ee = #False & etat_sys $= # AA & tt $ = $ tt + # 1 & unchanged ee "
AtoB : : action
"AtoB == ACT $ etat_sys = #AA & $ tt < #dd & $ ee = #TRUE & etat_sys $= # BB & unchanged (tt, ee) "
AtoC : : action
"AtoC == ACT $ etat_sys = #AA & $ tt = #dd & etat_sys $= # CC & unchanged (tt, ee) "
change : : action
"change == ACT $ etat_sys = #AA & $ ee = #False & ee $= # True & unchanged (tt, etat_sys) "
Next : : action
"Next == ACT $ (AtoA | AtoB | AtoC | change)"
Timeout : : temporal
"Timeout == TEMP (Init InitTimeout
$\& \Box[Next]_{(-etat\_sys,tt,ee)}$
$\& \Box WF(AtoB)_{-(etat\_sys,tt,ee)}$
$\& \Box WF(AtoC)_{-(etat\_sys,tt,ee)}$*)"*
Live1 : : temporal
"Live1 == TEMP ((etat_sys = #AA & ee = #True & tt < # dd) $\rightsquigarrow$ etat_sys = #BB)"
Live2 : : temporal
"Live2 == TEMP ((etat_sys = #AA & tt = # dd) $\rightsquigarrow$ etat_sys = #CC)"

Les propriétés d'équité et de fatalité ont été prouvées par le prouveur de théorèmes Isabelle.

6 Conclusion

Dans ce papier, nous avons présenté une méthode de spécification et de vérification des systèmes réactifs et sûrs de fonctionnement en utilisant conjointement la méthode B événementiel et la logique temporelle des actions TLA+. L'apport de regroupement de ces deux notations s'explique par le traitement des propriétés d'équité qui à notre connaissance constitue une contribution originale par rapport aux extensions apportées à la méthode B. Nous avons présenté une démarche regroupant les deux méthodes et utilisant les deux outils de vérification associés l'AtelierB et le prouveur Isabelle. Comme perspectives, nous tenons à valider la préservation des propriétés exprimées dans les nouvelles clauses FAIRNESS et EVENTUALITY par le raffinement et à automatiser le processus de transformation d'un modèle B vers une spécification équivalente en TLA+.

Références

[1] J-R. Abrial. *The B-Book : Assigning Programs to Meanings*. Cambridge University Press, 1996.

[2] J-R. Abrial. Extending B without changing it (for developing distributed systems). In Henri Habrias, editor, *Proceedings of the 1st Conference on the B method*, pages 169–191, November 1996.

[3] J-R. Abrial and L. Mussat. Introducing dynamic constraints in B. In Didier Bert, editor, *B'98 : The 2nd International B Conference*, volume 1393 of *Lecture Notes in Computer Science (Springer-Verlag)*, pages 83–128, Montpellier, April 1998. Springer Verlag.

[4] R-J. Back. Refinement calculus, part II : Parallel and reactive programs. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[5] R-J. Back and K-Sere. Stepwise refinement of action systems. In *Mathematics of Program Construction.*, pages 115–138, Berlin - Heidelberg - New York, June 1989. Springer.

[6] D. Berry, S. Moisan, and J. Rigault. Esterel : Towards a synchronous and semantically sound high level language for real-time applications. In *Proceedings of the 1983 IEEE Real-Time Systems Symposium*, pages 30–37, December 1983.

[7] G. Berry and L. Cosserat. The esterel synchronous programming language and its mathematical semantics. In A. W. Roscoe S. D. Brookes and G. Winskel, editors, *Proceedings of the Seminar on Concurrency*, volume 197 of *LNCS*, pages 389–448. Springer, July 1984.

[8] M. Büchi and E. Sekerinski. Formal methods for component software : The refinement calculus perspective. *Lecture Notes in Computer Science*, 1357 :332–350, 1998.

[9] K-M. Chandy and J. Misra. *Parallel Program Design*. Addison-Wesley, Austin, Texas, May 1989.