HAL
archives-ouvertes.fr

Spécification et vérification des propriétés de vivacité en B événementiel

Olfa Mosbahi, Jacques Jaray, Samir Ben Ahmed

▶ **To cite this version:**

Olfa Mosbahi, Jacques Jaray, Samir Ben Ahmed. Spécification et vérification des propriétés de vivacité en B événementiel. 6ème Colloque Francophone sur la Modélisation des Systèmes Réactifs - MSR 2007, Oct 2007, Lyon, France. 16 p., 2007. <inria-00172417>

HAL Id: inria-00172417
https://hal.inria.fr/inria-00172417

Submitted on 17 Sep 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Spécification et vérification des propriétés de vivacité en B événementiel

Olfa Mosbahi[*,**] — **Jacques Jaray**[*] — **Samir Ben Ahmed**[**]

[*] *Laboratoire LORIA INRIA Lorraine*
Campus Scientifique - BP 239 - 54506 Vandoeuvre-lès-Nancy Cedex
{mosbahi, jaray}@loria.fr

[**] *Faculté des Sciences, Campus universitaire - B.P. n˚244, 1060 Tunis El Manar II*
{benahmed}@fst.rnu.tn

RÉSUMÉ. *Dans ce papier, nous nous intéressons à la vérification de propriétés de vivacité sur des systèmes réactifs. Nous nous basons sur le B événementiel pour la spécification et la validation de tels systèmes. En considérant la limitation du B aux propriétés d'invariance, nous proposons d'appliquer le langage TLA*$^+$ *pour la vérification de telles propriétés. Nous étendons, en particulier, l'expressivité et la sémantique d'un modèle B pour obtenir un modèle B temporel. En transformant le modèle B étendu en un module TLA*$^+$*, nous pouvons vérifier ces propriétés grâce au prouveur de théorèmes Isabelle.*

ABSTRACT. *This paper deals with the verification of liveness properties on reactive systems. We are based on the event B method to specify and validate such systems. By considering the limitation of the B to invariance properties, we propose to apply the language TLA*$^+$ *to verify liveness properties on a software behaviour. We extend, in particular, the expressivity and the semantics of the event B method to deal with the specification of fairness and eventuality properties. By transforming an extended B model into TLA*$^+$ *module, we can verify these properties thanks to the theorem proving Isabelle.*

MOTS-CLÉS : *Spécification, Propriété d'équité, Propriété de fatalité, Raffinement, B événementiel, TLA*$^+$*, Vérification.*

KEYWORDS: *Specification, Fairness properties, Eventuality properties, Refinement, Event B method, TLA*$^+$*, Verification.*

1. Introduction

Le développement de systèmes sûrs de fonctionnement est l'une des préoccupations actuelles en ingénierie des systèmes. Il s'agit le plus souvent des systèmes réactifs événementiels interagissant avec l'environnement. Ces systèmes sont "orientés contrôle" et devant respecter le plus souvent des contraintes temporelles. Les méthodes formelles permettent la vérification et la validation de la spécification au moyen de techniques s'appuyant sur la preuve. Dans le cadre de notre étude, nous nous intéressons au développement formel de systèmes de contrôle commande où deux types de propriétés peuvent être considérés : des propriétés d'invariance et des propriétés de vivacité telles que la fatalité et l'équité. Nous nous basons sur le B événementiel pour la spécification et la validation de tels systèmes. Cette méthode fournit un ensemble de techniques et d'outils pour la spécification, le raffinement, la validation et l'implémentation de tels systèmes. Néanmoins, sa limitation concerne le type de propriétés qu'il vérifie car seuls les invariants sont considérés. Pour cela, nous proposons d'appliquer le langage TLA$^+$ [LAM 02] pour la vérification des propriétés de vivacité. Le langage de modélisation TLA$^+$ se base sur le concept de raffinement, d'action et de transition qui exprime une compatibilité avec une modélisation B événementiel.

Nous proposons dans ce papier, une approche de spécification et de vérification utilisant conjointement le B événementiel et le langage TLA$^+$. Cette combinaison nous permet de profiter de l'environnement de spécification du B ainsi que de l'outil de preuve associé pour vérifier des propriétés de sûreté et aussi de pouvoir vérifier en TLA$^+$ des propriétés de fatalité et d'équité qui ne peuvent pas être vérifiées naturellement avec le prouveur de B. Nous proposons dans un premier temps, d'étendre l'expressivité et la sémantique du B événementiel pour obtenir un modèle B temporel spécifiant des propriétés de fatalité et d'équité. Par la suite, nous proposons des règles de preuve pour la vérification de telles propriétés dans l'axiomatique de B. Dans un second temps, nous proposons de vérifier ces propriétés en utilisant le prouveur de théorèmes Isabelle, après avoir transformé le modèle B temporel vers un module TLA$^+$. Le papier est organisé comme suit : dans la section 2, nous présentons les travaux existants autour du B événementiel. Dans la section 3, nous donnons un aperçu sur le B événementiel et le langage TLA$^+$. La section 4 présente la méthode proposée utilisant conjointement le B événementiel et TLA$^+$ [LAM 02, LAM 93]. Nous illustrons par la suite notre méthode sur le cas d'un système réalisant un time_out.

2. Etat de l'art

Plusieurs travaux de recherche se sont intéressés à étendre le B événementiel pour traiter les propriétés de vivacité. Jean Raymond Abrial et Louis Mussat ont proposé dans [ABR 98a, ABR 98b] d'introduire dans les spécifications la description de contraintes dynamiques permettant d'exprimer des propriétés qui ne peuvent être décrites en terme d'invariants. Afin de vérifier ces propriétés dynamiques, il faut exhiber, en plus de la propriété un *invariant de boucle* et une fonction décroissante *VARIANT*. Le *VARIANT* décrit la raison pour laquelle les événements du système conduisent inévitablement à un état qui satisfait un prédicat donné. L'invariant de boucle est une propriété complétant l'invariant du système qui est nécessaire à la démonstration de la terminaison des portions de chemins conduisant à un état où le prédicat donné est vérifié. Chouali et Julliand [BEL 00] se sont intéressés à la vérification par model checking des propriétés dynamiques exprimables en PLTL sur des systèmes réactifs spécifiés en B étendus par des hypothèses d'équité. Ils ont aussi apporté des solutions au problème de la vérification par

model checking qui est l'explosion combinatoire en utilisant la démarche de spécification par raffinement. Leur objectif était de simplifier la vérification de propriétés du raffinement qui est vérifié séparément.

Didier Bert et Barradas [BAR 02, BAR 05, BER 01] dans leur travaux ont proposé une méthode de spécification et de vérification des propriétés de vivacité sous hypothèses d'équité dans les systèmes d'événements B. Ils se sont inspirés de la méthode UNITY. L'équité prise en compte par ce travail est l'équité faible et la "minimal progress" : à chaque fois que le système arrive dans un état, s'il a un moyen d'avancer en activant une transition alors la transition sera activée. Les propriétés de vivacité étudiées sont de la forme "$P\ LeadsTo\ Q$" . Les propriétés de vivacité sont divisées en deux classes : propriétés de base et propriétés générales. Les auteurs ont donné des obligations de preuve fondées sur le calcul des plus faibles préconditions et pas sur les traces pour prouver les propriétés de base. Les propriétés de vivacité générales sont prouvées par les définitions et théorèmes de la logique UNITY. Ils ont donné les obligations de preuve de la préservation des propriétés de vivacité dans le raffinement. Dans le cadre de notre travail, nous proposons d'étudier la sémantique ainsi que les règles de preuve des propriétés de vivacité en se basant sur les traces d'exécution.

3. Aperçu sur les méthodes et outils utilisés

Pour présenter la contribution de ce papier, nous présentons dans cette section le B événementiel et le langage TLA$^+$ et leurs outils associés l'Atelier B et le prouveur Isabelle.

3.1. *Le B événementiel*

Le B événementiel est une extension de B [ABR 96] qui permet la spécification de systèmes réactifs [ABR 98a, ABR 98b]. Dans cette extension, les concepts de machine abstraite et d'opérations sont respectivement remplacés par ceux de système abstrait et d'événements. Ces systèmes abstraits peuvent être vus comme des systèmes fermés, qui modélisent le système 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.

Événements. Un événement B modélise une transition discrète et peut être défini par une relation *"avant-après"* notée *BA(x,x')*, où x et x' désignent respectivement la valeur des variables avant et après l'exécution de la substitution. Les événements ont trois formes possibles : le premier cas correspond à un événement simple, le deuxième correspond à un événement gardé et le troisième à un événement non déterministe.

Dans le tableau de la figure 1, x_0 désigne la valeur initiale de x avant d'effectuer la substitution, G désigne un prédicat d'état, t un terme et P, R des substitutions généralisées. Les variables x doivent respecter l'invariant $I(x)$ et la vérification de cet aspect est réalisée par la preuve de conditions appelées *obligations de preuve*. L'événement d'initialisation est une substitution généralisée. Sa forme normale est $x : Init(x)$ et il faut que l'initialisation établisse l'invariant $I(x)$. L'obligation de preuve est la suivante : $Init(x) \Rightarrow I(x)$. Chaque événement doit préserver l'invariant. Si $BA(x, x')$ est le prédicat *"avant-après"* de l'événement alors l'obligation de preuve est la suivante : $I(x) \wedge BA(x, x') \Rightarrow I(x')$.

Evénement : E	*Prédicat"avant-après"*
evt = BEGIN $x : P(x_0, x)$ END	$P(x, x')$
evt = SELECT $G(x)$ THEN $x : P(x_0, x)$ END	$G(x) \wedge P(x, x')$
evt = ANY t WHERE $G(t, x)$ THEN $x : P(x_0, x, t)$ END	$\exists t.(G(t, x) \wedge P(x, x', t))$

Figure 1. *Formes d'événements*

3.2. *Le langage TLA*$^+$

TLA$^+$ est un langage de spécification due à Lamport [LAM 02], [LAM 91] étendant la logique temporelle des actions TLA à l'aide de la structure de module et la théorie des ensembles. TLA reprend les notations de langages de spécification comme Z ou VDM 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. La sémantique d'une spécification TLA est basée sur le comportement des variables à travers une séquence d'états. Une spécification TLA est une formule TLA de la forme :

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

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. La logique temporelle des actions possède des propriétés intéressantes liées au bégaiement. Le bégaiement (*stuttering*) désigne une suite d'états où certaines variables conservent la même valeur. Le bégaiement apparaît naturellement si l'on cache certaines variables et il joue par conséquent un rôle essentiel dans le raffinement. Deux traces qui ne diffèrent que par le bégaiement satisfont les mêmes formules TLA$^+$. Le raffinement s'identifie à l'implication logique. En TLA, il n'y a pas de distinction entre une spécification et une propriété : toutes les deux sont des formules TLA. Les propriétés du système qui nous intéressent le plus souvent sont les propriétés d'invariance, d'équité et de fatalité. L'invariance est une propriété de sûreté, l'équité et la fatalité sont des propriétés de vivacité.

Sûreté. Les formules TLA permettent d'exprimer les propriétés de sûreté. $\Box P$ exprime que P est toujours vrai. Généralement, prouver une propriété de sûreté revient à prouver que quelque chose de mauvais n'arrivera pas.

Vivacité. Les formules TLA permettent d'exprimer que les comportements infinis doivent satisfaire certaines propriétés. On exprime la vivacité dans TLA en fonction de la *fatalité (Eventuality)* et *l'équité (Fairness)*. On distingue *l'équité faible* et *l'équité forte*.

La fatalité, exprime que quelque chose de souhaitable arrivera et sa sémantique nécessite les traces d'exécution.

L'équité faible est définie par : $WF_f(A) \triangleq \Diamond\Box\neg Enabled\ \langle \mathrm{A} \rangle_f \Rightarrow \Box\Diamond\ \langle \mathrm{A} \rangle_f$. Elle exprime le fait que si une action A est autorisée indéfiniment alors elle va être exécutée infiniment

souvent. *L'équité forte* est définie par : $SF_f(A) \triangleq \Box\Diamond\neg Enabled\ \langle \mathrm{A} \rangle_f \Rightarrow \Box\Diamond\ \langle \mathrm{A} \rangle_f$. Elle exprime le fait que si une action A est autorisée infiniment souvent alors elle va être exécutée infiniment souvent.

4. Approche proposée

Dans cette section, nous proposons une approche pour la vérification des propriétés de vivacité à partir d'un modèle B temporel. B permet essentiellement d'exprimer et de vérifier des propriétés d'invariance. B concerne des transitions et son extension pour prendre en compte la notion d'équité et de fatalité requiert l'intégration d'une sémantique fondée sur les traces d'exécution. Pour ce faire, nous proposons :

– un cadre syntaxique des modèles B engendré par l'ajout des clauses décrivant des propriétés d'équité et de fatalité,

– un cadre sémantique basé sur les traces d'exécution,

– des règles de preuve pour la vérification de telles propriétés dans l'axiomatique de B,

– la transformation du modèle B temporel vers une spécification TLA$^+$, en utilisant le prototype B2TLA$^+$ que nous avons développé,

– la vérification de ces propriétés par le prouveur de théorèmes Isabelle.

Dans la section 4.1 nous présentons notre approche, ensuite nous proposons un cadre syntaxique et sémantique engendré par l'ajout des clauses d'équité et de fatalité. Enfin, nous proposons des règles de preuve des propriétés de vivacité dans l'axiomatique de B.

4.1. *Présentation de l'approche*

Dans cette section, nous proposons d'intégrer l'aspect temporel dans le B événementiel sans changer sa notation. Pour ce faire, nous proposons de regrouper le B événementiel et le langage TLA$^+$ dans une approche qui permet de spécifier et de vérifier des propriétés de sûreté, de fatalité et d'équité. Le B événementiel et la logique temporelle TLA$^+$ sont par nature différents mais ils se basent sur la notion des systèmes de transition et le raffinement ce qui font d'elles deux méthodes compatibles. Un modèle B comporte des éléments faciles à traduire vers TLA$^+$ à l'exception des aspects liés aux propriétés sur les traces qui ne peuvent pas être prises en compte dans B mais nous proposons de les exprimer dans la notation TLA$^+$ et de donner au modèle B étendu par ces nouvelles propriétés une sémantique basée sur les traces. Ainsi, nous pourrons transformer le modèle résultant vers un module TLA$^+$ et vérifier les propriétés ajoutées en utilisant le prouveur Isabelle. La méthode que nous proposons (Figure 2) 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,

3) Ajouter dans le modèle des propriétés de vivacité telles que des propriétés d'équité dans la clause FAIRNESS et de fatalité dans la clause EVENTUALITY au niveau des systèmes abstraits pour obtenir des systèmes abstraits temporels,

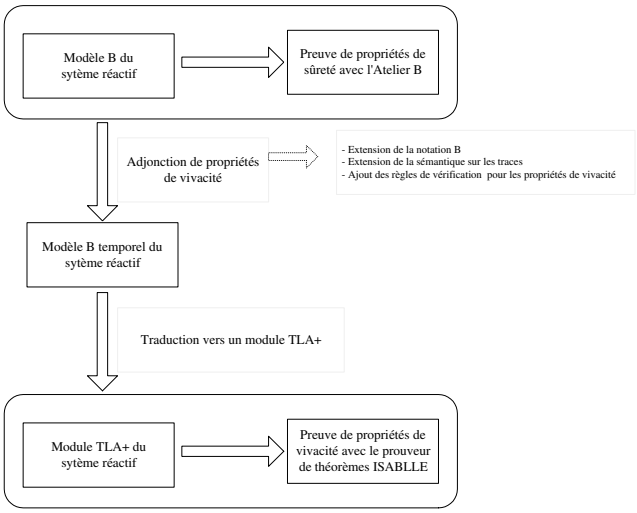

Figure 2. *Translation d'un modèle B vers un module TLA*$^+$

MODEL ⟨*nom*⟩
SETS ⟨*ensembles*⟩
CONSTANTS ⟨*constantes*⟩
VARIABLES ⟨*variables*⟩
INVARIANT ⟨*invariants*⟩
INITIALISATION ⟨*initialization de variables*⟩
EVENTS ⟨*événements*⟩
FAIRENESS ⟨*propriétés d'équité*⟩
EVENTUALITY ⟨*propriétés de fatalité*⟩
END

4) Transformer le modèle obtenu en une spécification TLA$^+$.

5) Vérifier les 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.

4.2. *Cadre syntaxique de l'extension*

Dans cette section, nous proposons une méthode pour étendre l'expressivité du B événementiel. Nous proposons l'utilisation de l'opérateur de la logique temporelle TLA$^+$ *"Leads to"* pour spécifier les propriétés de fatalité et $WF(e)$, $SF(e)$ pour spécifier respectivement les équités faible et forte. De ce fait, nous intégrons la logique TLA$^+$ dans les spécifications des systèmes d'événements B. Soit V l'ensemble dénombrable des variables flexibles ou d'états et X l'ensemble dénombrable de variables rigides (des variables qui ne sont pas modifiées par

des transitions du programme et gardent la valeur initiale choisie durant toute l'exécution). Soit $SP_{V \cup X}$, l'ensemble des prédicats d'états.

Formules de Transition. Une formule de transition décrit les transitions d'états. Une formule de transition ac est une formule de la forme :
$ac ::= GS(e) \mid [e]sp \mid \langle e \rangle sp$ où sp est un prédicat d'état, e est un événement et $GS(e)$ est la substitution généralisée de l'événement e.

Propriétés de sûreté. Les propriétés de sûreté sont des formules de la forme F : := $\Box sp$ | $\Box(sp \Rightarrow \Box sp)$, où sp est un prédicat d'état.

Propriétés de vivacité. Les propriétés de vivacité (fatalité et équité) sont des formules définies comme suit :

- Les propriétés de fatalité sont exprimées par des formules de la forme :
  $F \rightsquigarrow G$ (F *leads to* G) définies par $\Box(F \Rightarrow \Diamond G)$ et signifient que chaque F sera suivi par G, où F et G sont des formules de la forme : $F ::= sp \mid \Diamond F \mid \Box F \mid WF(e) | SF(e)$. où sp est un prédicat d'état, et $WF(e)$, $SF(e)$ sont respectivement les équités faible et forte de l'événement e.
- Les propriétés d'équité sont exprimées par des formules de la forme :
  - Pour l'équité faible d'un événement e : $WF(e)$ est définie par $\Diamond\Box grd(e) \Rightarrow \Box\Diamond GS(e)$.
  - Pour l'équité forte d'un événement e : $SF(e)$ est définie par $\Box\Diamond grd(e) \Rightarrow \Box\Diamond GS(e)$.

Où e est un événement, $grd(e)$ est la garde de l'événement e (prédicat d'état) et $GS(e)$ est la substitution généralisée (formule de transition).

4.3. *Cadre Sémantique de l'extension*

Dans cette section nous proposons une sémantique des propriétés de vivacité (équité, fatalité) basée sur les traces d'exécution. Nous présentons les événements, les systèmes, les états et les traces comme une spécification TLA$^+$. Dans notre extension, nous pouvons voir un ensemble d'événements comme une relation entre les variables primées et non primées et nous utilisons ce point pour définir l'extension du B événementiel. Un système S est modélisé par un ensemble d'événements déclenchant des actions quand les gardes sont vraies. Un événement e est défini par une garde grd dénotée par $grd(e)$ et par une relation sur un ensemble de variables flexibles (V) noté par $GS(e)$ (relation indiquant la transformation des variables). Respectant un module TLA$^+$, nous considérons trois types de propriétés :

- *Propriétés_d'états(S)* dénotent les propriétés sur les états de S et sont interprétés sur les états. Ces propriétés sont des prédicats d'états,
- *Propriétés_relationnelles(S)* dénotent les relations entre les paires d'états que nous appelons les formules de transition,
- *Propriétés_temporelles(S)* dénotent des propriétés définies sur les traces et utilisent des propriétés d'états, des propriétés relationnelles et des opérateurs temporels ($\Box$, $\Diamond$, $\rightsquigarrow$, ...).

Dans ce qui suit, nous dénotons par $s_i, \xi \models sp$ la satisfaction du prédicat d'état sp à l'état s_i du système de transition, par $\sigma, \xi \models F$ la satisfaction de la formule temporelle F sur une trace $\sigma \in Traces(S)$ et par I l'invariant du système. Les propriétés sont interprétées sur les traces d'états. Nous introduisons des notations pour caractériser les systèmes :

- V est l'ensemble de variables flexibles du système S ; v est une variable d'état, x est la valeur avant de v et x' est la valeur après de v. $Primed\_Var(S) = \{x' \mid v \in V\}$ et $Unprimed\_Var(S) = \{x \mid v \in V\}$.
- Val est l'ensemble des valeurs des variables du système S.
- $States(S)$ est l'ensemble des états du système S et un état s_i de S est une transformation de V vers Val :
$$s_i \in V \rightarrow Val.$$
- Une relation de satisfaction des prédicats d'états sur les états States(S) est définie comme suit :
$s_i, \xi \models sp$ dénote que le prédicat d'état sp est satisfait à l'état s_i, où ξ est la valuation des variables rigides de S.
- $Init(S)$ spécifie les valeurs initiales des variables flexibles du système S.
- $Events(S)$ spécifie l'ensemble des événements de S ; cela signifie que nous listons les événements possibles définis dans la figure 1. Un événement e est défini comme suit :
$$e \triangleq grd(e) \text{ then } GS(e)$$
- $grd(e)$ est pour chaque événement, un prédicat qui est vrai dans un état s_i ssi il est possible de faire un pas à partir de cet état.
- $GS(e)$ est la substitution généralisée de l'événement e correspondant à la modification des variables d'état et le passage du système de l'état s_i vers l'état s_j.
- $BA_e(x, x')$ est le prédicat avant-après de l'événement e (où x est la valeur avant de v et x' est la valeur après de v obtenu après l'exécution de e). C'est une formule de la logique du premier ordre construite à partir des constantes, des paramètres ainsi que des occurrences des variables primées et non primées.
- $\rightarrow$ est une relation sur States(S) simulant l'exécution du système S.
- $Next(S)$ est une formule sur les variables primées et non primées du système S correspondant à la relation sur $States(S)$, nommée $\rightarrow$. $Next(S)$ correspond à un des trois événements défini dans la figure 1.
$$Next(S) \triangleq P(x, x') \vee (G(x) \wedge P(x, x')) \vee (\exists t.(G(t, x) \wedge P(x, x', t)))$$
$(s_i, s_{i+1}) \models Next(S) \triangleq s_i \rightarrow s_{i+1}$. La validité de $Next(S)$ est définie sur les paires d'états et la propriété définie sur les paires d'états est nommée relation.
Les formules $BA_e(x, x')$ et *Next* expriment des relations entre les variables primées et non primées, ce qui induit une équivalence sémantique entre $BA_e(x, x')$ et $Next$. De ce fait BA_e est interprété par la formule $Next$ dans TLA$^+$.
- $Invariants(S)$ est un ensemble de propriétés sur States(S) invariant pour S. $\varphi \in Invariants(S)$, si
1) $Init(S) \Rightarrow \varphi$
2) $\forall\, s_0, s_i \in States(S)\ :\ (s_0, \xi \models Init(S) \wedge (s_0 \rightarrow^* s_i)) \Rightarrow s_i, \xi \models \varphi$
- $Traces(S)$ est l'ensemble des traces générées à partir de $Init(S)$ utilisant $\rightarrow$. Une trace est notée par $\sigma = (s_0, s_1, ..., s_i, ...)$ et satisfait les contraintes suivantes :
1) $s_0, \xi \models Init(S)$ (l'état initial s_0 satisfait la condition initiale),
2) $\forall i \in \mathbb{N} : (s_i \rightarrow s_{i+1}) \vee (s_i = s_{i+1})$, chaque deux états successifs (s_i, s_{i+1}) satisfait le prédicat avant-après $BA_e(x, x')$ pour des événements e et des variables x, ou bien que toutes les valeurs des variables restent inchangées (étapes de bégaiement)

Soit $\sigma \in Traces(S)$. Une propriété φ sur les séquences d'états du système S peut être une propriété d'état, une propriété relationnelle ou une propriété temporelle. La sémantique sur les traces unifie la sémantique sur les états et paires d'états comme suit :

1) une propriété d'état φ est une propriété de trace comme suit : $\sigma, \xi \models \varphi$ si $s_0, \xi \models \varphi$.

2) une propriété relationnelle φ est aussi une propriété de trace en étendant la sémantique sur les paires d'états vers une sémantique sur les traces comme suit : $\sigma, \xi \models \varphi$ si $(s_0, s_1), \xi \models \varphi$.

Les propriétés temporelles contiennent les prédicats d'états, les propriétés relationnelles et la combinaison temporelle de ces propriétés. Notre extension est la même qu'en TLA$^+$ et un système S est spécifié par l'expression temporelle suivante :

Specification(S) $\triangleq$ $\wedge\, Init(S)$ définit les conditions initiales,
$\wedge \Box[Next(S)]_{<unprimed\_var(S)>}$ définit la construction des traces
$\wedge WF_{unprimed\_var(S)}(S)$ définit la contrainte d'équité faible
$\wedge SF_{unprimed\_var(S)}(S)$ définit la contrainte d'équité forte

Où $WF_{unprimed\_var(S)}(S)$ définit la condition d'équité faible sur le système S et $SF_{unprimed\_var(S)}(S)$ définit la condition d'équité forte sur le système S.
$WF_{unprimed\_var(S)}(S)$ et $SF_{unprimed\_var(S)}(S)$ sont définis comme suit :

$WF_{unprimed\_var(S)}(S) \triangleq \bigwedge_{E \in WF\_Events(S)} WF_{unprimed\_var(S)}(E)$ et
$SF_{unprimed\_var(S)}(S) \triangleq \bigwedge_{E \in SF\_Events(S)} SF_{unprimed\_var(S)}(E)$

Où $WF\_Events(S)$ est l'ensemble des événements sous hypothèses d'équité faible et $SF\_Events(S)$ est l'ensemble des événements sous hypothèses d'équité forte. $WF_{unprimed\_var(S)}(E)$ dénote que l'événement E est exécuté sous hypothèses d'équité faible et $SF_{unprimed\_var(S)}(E)$ dénote que l'événement E est exécuté sous hypothèses d'équité forte. En effet, à chaque événement est associé une condition d'équité qui peut être faible, forte ou indéfinie.

Dans TLA$^+$, pour une action e la condition de permission $Enabled(e)$ est définie par la quantification existentielle sur les occurrences primées des variables d'état ; ainsi, le prédicat d'état $Enabled(e)$ est vrai pour les états qui ont un état successeur relié par une occurrence de l'événement e ($Enabled(e) \triangleq \exists x' : BA_e(x, x')$). La garde $grd(e)$ dans B est interprétée par la condition $Enabled(e)$ en TLA$^+$ parce qu'elles expriment les conditions de déclenchement des événements. Nous pouvons résumer la sémantique de l'extension de la notation B sur les traces par les équivalences suivantes en TLA$^+$:

$$grd(e) \quad \triangleq \quad Enabled(e).$$
$$BA_e \quad \triangleq \quad Next.$$

Interprétation de formules

Soit $\sigma = s_0 s_1 \ldots$ une séquence d'états et ξ la valuation des variables rigides de S. Soit e un événement où e = grd(e) then GS(e).

Formules de transition

$(s, s'), \xi \models GS(e)$ ssi $s \xrightarrow{e} s'$

$s, \xi \models [e]sp'$ ssi pour toute exécution de l'événement e, si $s \xrightarrow{e} s'$ alors l'état $s', \xi \models sp'$

$[e]sp$ est la notation ensembliste de la plus faible précondition $wp(e, sp)$. Cette plus faible condition assure que le prédicat d'état sp est vrai après exécution de e.

$s_i, \xi \models \langle e\rangle sp'$ ssi il existe une exécution de l'événement e, tel que si $s \xrightarrow{e} s'$ alors l'état $s', \xi \models sp'$

$\langle e\rangle sp$ est la plus faible précondition conjuguée $\neg wp(e, \neg sp)$ qui donne la possibilité d'atteindre le prédicat d'état sp après exécution de e.

Formules temporelles

Nous interprétons une formule temporelle comme une assertion sur les comportements. Dans les définitions ci-dessous, $\sigma|_i, \xi \models F$ signifie que la formule F est satisfaite sur le suffixe de σ à partir de i.

$\sigma, \xi \models \Box F$ ssi $\sigma|_i, \xi \models F$ pour tout $i \in \mathbb{N}$

La formule $\Box F$ (always F) asserte que F est vraie à tout instant durant le comportement σ.

$\sigma, \xi \models \Diamond F$ ssi $\sigma|_i, \xi \models F$ pour quelques $i \in \mathbb{N}$ $(\Diamond F \equiv \neg\Box\neg F)$

La formule $\Diamond F$(eventually F) asserte que F est vraie pour quelques suffixes du comportement σ.

Leads-to property. Cette formule asserte que chaque suffixe satisfaisant la propriété temporelle F est suivi par des suffixes satisfaisant la propriété temporelle G. ($F \rightsquigarrow G \equiv \Box(F \Rightarrow \Diamond G)$).

$\sigma, \xi \models F \rightsquigarrow G$ ssi pour tout $i \in \mathbb{N}$, si $\sigma|_i, \xi \models F$ alors $\sigma|_j, \xi \models G$ pour quelques $j \geq i$.

Propriété de vivacité sous hypothèse d'équité faible. Un comportement est d'équité faible pour quelques événements e ssi e apparaît infiniment souvent quand il est autorisé indéfiniment $(WF(e) = \Diamond\Box\, grd(e) \Rightarrow \Box\Diamond GS(e))$.

$\sigma, \xi \models WF(e)$ ssi il existe $j \in \mathbb{N}$ tel que pour tout $i \geq j$, $\sigma|_i, \xi \models grd(e)$ alors pour tout $n \in \mathbb{N}$, il existe $m \in \mathbb{N}$ tel que pour tout $k \geq n + m$, $(s_i, s_k), \xi \models GS(e)$.

Propriété de vivacité sous hypothèse d'équité forte. Un comportement est d'équité forte pour un événement e ssi e apparaît infiniment souvent quand il est autorisé infiniment souvent $(SF(e) = \Box\Diamond\, grd(e) \Rightarrow \Box\Diamond GS(e))$.

$\sigma, \xi \models SF(e)$ ssi pour tout $i \in \mathbb{N}$, il existe $j \in \mathbb{N}$ tel que pour tout $l \geq i + j$, $\sigma|_l, \xi \models grd(e)$ alors pour tout $n \in \mathbb{N}$, il existe $m \in \mathbb{N}$ tel que pour tout $k \geq n + m$, $(s_i, s_k), \xi \models GS(e)$.

4.4. *Règles de preuves de propriétés de vivacité*

Dans le formalisme de B, nous proposons les règles WF, SF et WFO pour la vérification de propriétés de vivacité.

Sous hypothèse d'équité faible
Soit S un système d'événements B et $WF(S)$ est l'ensemble des événements exécutés sous hypothèse d'équité faible du système S. La règle suivante est utilisée pour prouver une propriété de *"Leadsto"* sous hypothèse d'équité faible.

$$WF.\ \frac{\begin{array}{l} I \wedge P \wedge \neg Q \Rightarrow [e](P \vee Q) \text{ pour tout événement } e \text{ de } S \\ \text{il existe un événement } e \in S \text{ où :} \\ \quad \text{I} \wedge P \wedge \neg Q \Rightarrow \langle e \rangle\ true \wedge [e]Q \\ \quad e \in WF(S) \end{array}}{\text{S} \models P \rightsquigarrow Q}$$

Dans cette règle, P et Q sont des prédicats d'état, I est l'invariant du système S. Par la première prémisse, n'importe quel successeur d'un état satisfaisant P doit satisfaire P ou Q, ainsi P est vrai tant que Q ne l'est pas. Par la deuxième prémisse, il existe un successeur d'un état satisfaisant P doit satisfaire Q et s'assure qu'en tous ces états, l'événement e est permis ($\langle e \rangle\ true$ signifie la condition de faisabilité de l'événement e), et la condition d'équité faible assure que fatalement l'événement e se produira, à moins que Q ne soit devenu vraie avant. Enfin, la troisième prémisse s'assure que e est un événement pour lequel l'équité faible est assurée.

Preuve de propriétés de vivacité sous hypothèse d'équité faible
Supposons que $\sigma = s_0, s_1, ...s_i, ...$ est une trace satisfaisant $\Box I \wedge WF(e)$, et que P est satisfait à l'état s_i. Nous devons prouver que Q est satisfait dans quelques états s_j avec $j \geq i$. Soit s_0 l'état initial et s_i satisfait P. Supposons qu'aucun état successeur ne satisfait Q, ainsi tous les états successeurs doivent satisfaire P. Par la seconde prémisse, il existe un successeur d'un état satisfaisant P dans lequel un événement e est permis et son exécution mène toujours dans un état satisfaisant Q (contradiction). Ainsi, à partir d'un état s_i satisfaisant P, nous pouvons atteindre un état $s_j (j \geq i)$ satisfaisant Q par l'exécution d'un événement e sous hypothèse d'équité faible.

Sous hypothèse d'équité forte
Soit S un système d'événements B et $SF(S)$ l'ensemble des événements d'équité forte. Similaire à la règle précédente, la règle suivante est utilisée pour prouver une formule de *"Leadsto"* sous hypothèse d'équité forte.

$$SF.\ \frac{\begin{array}{l} I \wedge P \wedge \neg Q \Rightarrow [e](P \vee Q) \text{ pour tout événement } e \text{ de } S \\ \text{il existe un événement } e \in S \text{ où :} \\ \quad \text{I} \wedge P \wedge \neg Q \Rightarrow [e]Q \\ \quad S \models \Box(I \wedge P \wedge \neg Q) \Rightarrow \Diamond grd\ (e) \\ \quad e \in SF(S) \end{array}}{\text{S} \models P \rightsquigarrow Q}$$

Dans cette règle, P et Q sont des prédicats d'état, I est l'invariant du système S, e est un événement d'équité forte. Nous supposons que σ est un comportement satisfaisant $\Box I \wedge SF(e)$ et P est satisfait à l'état s_i. Nous devons prouver que Q est satisfait par quelques états s_j avec $j \geq i$. Par la première prémisse, n'importe quel successeur d'un état satisfaisant P doit satisfaire P ou Q. Par la deuxième prémisse, il existe un événement $e \in S$ où son exécution à partir d'un état satisfaisant p évolue vers un état satisfaisant Q. La troisième prémisse s'assure qu'en tous ces états, l'événement e est permis, et ainsi la condition d'équité forte assure que par la suite que e se produira, à moins que Q ne soit devenu vrai avant. Enfin, la dernière prémisse assure que e est un événement avec une équité forte.

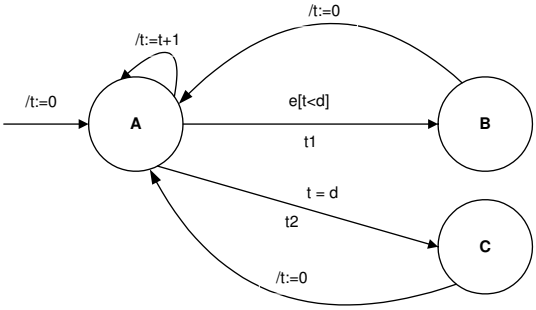

Figure 3. *Statechart représentant le time-out*

Utilisation de la règle WFO
La règle WFO est utilisée pour vérifier les propriétés de vivacité en utilisant les relations bien formées. $(S, \prec)$ est une relation binaire telle qu'il ne va pas exister une chaîne descendante infinie $x_1 \prec x_2, ...$ d'éléments $x_i \in S$. F et G sont des formules temporelles.

$$\text{WFO.}\frac{\begin{array}{c}(S,\prec)\ \ est\ une\ relation\ bien\ \ formée\ \ sur\ un\ ensemble\ S \\ \forall x \in S : F(x) \rightsquigarrow G \vee (\exists y \in S : (y \prec x) \wedge F(y))\end{array}}{(\exists x \in S : F(x)) \rightsquigarrow G \quad (x\ n'est\ pas\ libre\ dans\ G)}$$

Dans cette règle, x et y sont des variables rigides tels que x n'apparaît pas dans G et y n'apparaît pas dans F. La deuxième hypothèse de la règle est elle-même une formule temporelle qui exige que chaque occurrence de F pour n'importe quelle valeur $x \in S$ est suivie soit d'une occurrence de G, ou par une formule F pour une valeur de y plus petite. Puisque la première hypothèse s'assure qu'il ne peut pas y avoir une chaîne descendante infinie des valeurs dans S, fatalement G doit être vrai.

Autres règles de vérification

$$\frac{P \rightsquigarrow Q \quad Q \rightsquigarrow R}{P \rightsquigarrow R}\ (trans) \qquad \frac{P \rightsquigarrow Q \quad R \rightsquigarrow Q}{P \vee R \rightsquigarrow Q}\ (disj)$$

$$\frac{P \Rightarrow Q}{P \rightsquigarrow Q}\ (dedu) \qquad \frac{P \rightsquigarrow Q}{(\exists x : P(x)) \rightsquigarrow (\exists x : Q(x))}\ (exists)$$

Ces règles sont utilisées pour combiner des formules élémentaires de $Leadsto$.

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 que le système est à l'état *A*, si un signal de l'environnement apparaît dans un intervalle de *d* unités de temps alors le système passe à l'état *B*. Si rien ne s'est passé à l'issue de la période *d*, le système passe à l'état *C*. Pour avoir un comportement infini, les états B et C rebouclent sur l'état A avec une remise à 0 du temps. Le système est représenté par un statechart à la figure 3.

Etape 1 : Modèle B du système. Nous appelons *Etats* l'ensemble des trois états du système A, B, C. d est une constante qui désigne la durée maximale à passer dans l'état A. Les variables $etat\_sys, t, e$ désignent respectivement l'état du système, le compte du temps et l'occurrence d'un événement e. L'événement *"AtoA"* exprime les conditions sous lesquelles le système reste à l'état *A* (pas de réception de l'événement *e* et la valeur de *t* n'a pas atteint *d* unités de temps). L'événement *"AtoB"* exprime la condition de transit de l'état *A* vers *B* sous l'occurrence de l'événement *e*. L'événement *"AtoC"* exprime la condition de transit de l'état *A* vers *C*. L'événement *"Change"* permet de changer d'une manière aléatoire l'occurrence de l'événement *e*. L'événement *"BtoA"* reboucle l'état B sur l'état A et l'événement *"CtoA"* reboucle l'état C vers l'état A.

MODEL $time\_out$
SETS
Etats = {*A,B, C*} ;
CONSANTS
d
PROPERTIES
d= 10
VARIABLES
etat_sys, t, e
INVARIANT
etat_sys ∈ *Etats* ∧ *t* ∈ NATURAL ∧ *e* ∈ BOOL ∧ $t \leq d$
INITIALISATION
etat_sys :=*A* || *t* :=*0* || *e* :=*FALSE*
EVENTS
AtoA ≙ **SELECT** *etat_sys*=*A* ∧ *e*= FLASE ∧ t<d
THEN *etat_sys* :=*A* || *t* := *t+1* **END** ;
AtoB ≙ **SELECT** *etat_sys*=*A* ∧ *e*= TRUE ∧ t<d
THEN *etat_sys* :=*B* **END** ;
AtoC ≙ **SELECT** *etat_sys*=*A* ∧ *t*<*d* **THEN** *etat_sys* :=*C* **END** ;
BtoA ≙ **SELECT** *etat_sys*=*B* **THEN** *etat_sys* :=*A* || *t* :=*0* **END** ;
CtoA ≙ **SELECT** *etat_sys*=*C* **THEN** *etat_sys* :=*A* || *t* :=*0* **END** ;
Change ≙ **BEGIN** *e* : : BOOL **END**
END

Etape 2 : Preuves. Le modèle a été prouvé par le prouveur automatique de l'AtelierB et les invariants sont préservés par les événements.

Etape 3 : Modèle B temporel. 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 exécutés sous hypothèse d'équité faible.

2) Une clause EVENTUALITY qui décrit les propriétés de fatalité exprimant que le système atteindra fatalement l'état B ou C.

MODEL $time\_out$
SETS *Etats* = {*A,B, C*} ;
CONSANTS *d*
PROPERTIES *d*= 10
VARIABLES *etat_sys, t, e*
INVARIANT
etat_sys $\in$ *Etats* $\wedge$ *t*$\in$ NATURAL $\wedge$ *e*$\in$ BOOL$\wedge$ $t \leq d$
INITIALISATION
etat_sys :=*A* || *t* :=*0* || *e* :=*FALSE*
EVENTS
AtoA $\triangleq$ **SELECT** *etat_sys*=*A* $\wedge$ *e*= FLASE $\wedge$ t<d **THEN** *etat_sys* :=*A* || *t* := *t+1* **END** ;
AtoB $\triangleq$ **SELECT** *etat_sys*=*A* $\wedge$ *e*= TRUE $\wedge$ t<d **THEN** *etat_sys* :=*B* **END** ;
AtoC $\triangleq$ **SELECT** *etat_sys*=*A* $\wedge$ *t*<*d* **THEN** *etat_sys* :=*C* **END** ;
Change $\triangleq$ **BEGIN** *e* : : BOOL **END** ;
BtoA $\triangleq$ **SELECT** *etat_sys*=*B* **THEN** *etat_sys* :=*A* || *t* :=*0* **END** ;
CtoA $\triangleq$ **SELECT** *etat_sys*=*C* **THEN** *etat_sys* :=*A* || *t* :=*0* **END** ;
FAIRNESS WF(AtoB) $\wedge$ WF(AtoC)
EVENTUALITY *etat_sys = A* $\leadsto$ *etat_sys = B* $\vee$ *etat_sys = C*
END

Etape 4 : Traduction du modèle B temporel en un module TLA$^+$. Pour pouvoir vérifier les propriétés d'équité et de fatalité, nous transformons le modèle B temporel en un module TLA$^+$. Ce module décrit les variables, les prédicats et les actions de la spécification. La transformation d'un modèle B temporel en un module TLA$^+$ est réalisée par le prototype B2TLA$^+$ que nous avons développé.

MODULE $time\_out$

CONSTANTS *d, A, B, C,*
VARIABLES *etat_sys , t, e*
ASSUME *d* $\in$ *NATURAL*
Etat_sys = {A, B, C}
TypeInvariant = $\wedge$ *t* $\in$ *NATURAL* $\wedge$ *etat_sys* $\in$ *Etat_sys* $\wedge$ $t \leq d$
$\wedge$ *e* $\in${*TRUE, FALSE*}
Init = $\wedge$ *etat_sys = A* $\wedge$ *t = 0* $\wedge$ *e = FALSE*
AtoA = $\wedge$ *etat_sys = A* $\wedge$ *t < d* $\wedge$ *e = FALSE* $\wedge$ *t' = t + 1*
$\wedge$ $UNCHANGED_{<etat\_sys,e>}$
AtoB = $\wedge$ *etat_sys = A* $\wedge$ *t < d* $\wedge$ *e = TRUE* $\wedge$ *etat_sys' = B*
$\wedge$ $UNCHANGED_{<t,e>}$
AtoC = $\wedge$ *etat_sys = A* $\wedge$ *t = d* $\wedge$ *etat_sys' = C*
$\wedge$ $UNCHANGED_{<t,e>}$
BtoA = $\wedge$ *etat_sys = B* $\wedge$ *etat_sys' = A* $\wedge$ *t' = 0*
$\wedge$ $UNCHANGED_{<e>}$
CtoA = $\wedge$ *etat_sys = C* $\wedge$ *etat_sys' = A* $\wedge$ *t' = 0*
$\wedge$ $UNCHANGED_{<e>}$
change = $\wedge$ *e' $\in$ {TRUE, FALSE}*
$\wedge$ $UNCHANGED_{<etat\_sys,t>}$

Next = AtoA $\vee$ **AtoB** $\vee$ **AtoC** $\vee$ **change** $\vee$ **BtoA** $\vee$ **CtoA**
trvars = <etat_sys, t, e>
FAIRNESS = $WF_{trvars}(AtoB) \wedge WF_{trvars}(AtoC)$
Spec = Init $\wedge\Box[Next]_{trvars}\wedge$ **FAIRNESS**
Liveness = $\wedge$ *etat_sys=A* $\leadsto$ *etat_sys = B* $\vee$ *etat_sys = C*
THEOREM Spec $\Rightarrow \Box$ **TypeInvariant**
THEOREM Spec $\Rightarrow$ **Liveness**

Etape 5 : Vérification des propriétés de vivacité 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 module TLA$^+$ vers un modèle compréhensible par Isabelle utilisant le codage de TLA$^+$ en Isabelle. Le modèle ci-dessous est un module TLA$^+$ en Isabelle. Les propriétés d'équité et de fatalité ont été prouvées par le prouveur de théorèmes Isabelle.

time_out.thy

Timeout = TLA+
datatype *etat_sys = A || B || C*
constdefs *d : : nat "d==10"*
constdefs *etat_sys : : Etats stfun*
t : : nat stfun e : : stpred
rules *base "basevars (etat_sys, t, e)"*
constdefs *InitTimeout : : stpred*
"InitTimeout == PRED etat_sys = #A & t=#0 & e = #False"
AtoA : : action
"AtoA == ACT $ etat_sys = #A & $ t < #d & $ e = #False & etat_sys $= #A & t $ = $ t + # 1 & unchanged e "
AtoB : : action
"AtoB == ACT $ etat_sys = #A & $ t < #d & $ e = #TRUE & etat_sys $= #B & unchanged (t, e) "
AtoC : : action
"AtoC == ACT $ etat_sys = #A & $ t = #d & etat_sys $= # C & unchanged (t, e) "
BtoA : : action
"BtoA == ACT $ etat_sys = #B & etat_sys $= # A & t $ = $ # 1 & unchanged (e) "
CtoA : : action
"CtoA == ACT $ etat_sys = #C & etat_sys $= # A & t $ = $ # 1 & unchanged (e) "
change : : action
"change == ACT e $: : {# True, # FALSE} & unchanged (t, etat_sys) "
Next : : action
"Next == ACT $ (AtoA | AtoB | AtoC | change | BtoA | CtoA)"
Timeout : : temporal
"Timeout == TEMP (Init InitTimeout
$\& \Box[Next]_{(-etat\_sys,t,e)}$
$\& \Box WF(AtoB)_{-(etat\_sys,t,e)}$
$\& \Box WF(AtoC)_{-(etat\_sys,t,e)}$*)"*
Live : : temporal
"Live == TEMP (etat_sys = #A $\leadsto$ *etat_sys = #B* $\vee$ *etat_sys = #C)"*

Le modèle utilise les notions de function d'état (*stfun*), prédicat d'état (*stpred*), d'actions (*action*) et de formules temporelles (*temporal*). En effet, les variables non booléennes sont considérées comme des fonctions d'état (*stfun*), les variables booléennes et l'initialisation (*Init*) sont des prédicats d'état (*stpred*). Pour les actions, la valeur de la variable avant l'exécution de la substitution est dénotée par $ *variable* et la valeur de la variable après l'exécution de la substitution par *variable* $.

6. Conclusion

Dans ce papier, nous nous sommes intéressés à la spécification et la vérification de systèmes réactifs. Nous avons proposé une méthode de vérification de propriétés de vivacité sur de tels systèmes à l'aide du B événementiel. En considérant la limitation du B aux propriétés d'invariance, nous proposons d'appliquer le langage TLA$^+$ pour la vérification de telles propriétés. Nous étendons en particulier l'expressivité et la sémantique d'un modèle B pour obtenir un modèle B temporel. En transformant le modèle B étendu en un module TLA$^+$, nous pouvons vérifier ces propriétés grâce au prouveur de théorèmes Isabelle. L'apport de la combinaison de ces deux méthodes s'explique par le traitement des propriétés d'équité et de fatalité tout en profitant du cadre de développement en B, et de la puissance de spécification et de vérification avec TLA$^+$. Dans nos futurs travaux, nous planifions dans un premier temps au niveau du raffinement, la vérification de la préservation des propriétés exprimées dans les nouvelles clauses FAIRNESS et EVENTUALITY. De plus, nous envisageons à tester cette approche sur un système industriel de contrôle commande complexe. Enfin, nous planifions à vérifier des propriétés temporelles sur de tels systèmes en prenant en compte l'approche proposée dans ce papier.

Remerciements Nous remercions Stephan Merz pour ses commentaires et son aide.

7. Bibliographie

[ABR 96] ABRIAL J.-R., *The B book - Assigning Programs to Meanings*, Cambrige University Press, 1996.

[ABR 98a] ABRIAL J.-R., MUSSAT L., « Introducing Dynamic Constraints in B. », BERT D., Ed., *B'98 : The 2nd International B Conference*, vol. 1393 de *Lecture Notes in Computer Science (Springer-Verlag)*, Montpellier, avril 1998, Springer Verlag, p. 83–128.

[ABR 98b] ABRIAL J.-R., MUSSAT L., « Introducing Dynamic Constraints in B. », *B*, 1998, p. 83-128.

[BAR 02] BARRADAS H.-R., BERT D., « Specification and Proof of Liveness Properties under Fairness Assumptions in B Event Systems », *IFM*, 2002, p. 360–379.

[BAR 05] BARRADAS H.-R., BERT D., « Proof obligations for specification and refinement of liveness properties under weak fairness », février 09 2005.

[BEL 00] BELLEGARDE F., DARLOT C., JULLIAND J., KOUCHNARENKO O., « Reformulate Dynamic Properties during B Refinement and Forget Variants and Loop Invariants », *Lecture Notes in Computer Science*, vol. 1878, 2000, p. 230–245.

[BER 01] BERT D., « Preuve de propriétés d'équité en B : Preuve de l'algorithme d'arbitrage du bus SCSI-3 », *AFADL'2001*, ADER/LORIA, juin 2001, p. 221–241.

[LAM 91] LAMPORT L., « The Temporal Logic of Actions », rapport n^o 79, décembre 1991, Digital Equipment Corporation, Systems Research Centre.

[LAM 93] LAMPORT L., « Hybrid Systems in TLA$^+$ », GROSSMAN R. L., NERODE A., RAVN A. P., RISCHEL H., Eds., *Hybrid Systems*, vol. 736 de *Lecture Notes in Computer Science*, Springer-Verlag, 1993, p. 77–102.

[LAM 02] LAMPORT L., *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers*, Addison-Wesley, 2002.