

Propriétés dynamiques avec hypothèses d'équité en B événementiel

Héctor Ruíz Barradas ^{***} — Didier Bert ^{**}

* *Universidad Autónoma Metropolitana Azcapotzalco*

** *Laboratoire LSR-IMAG, Grenoble*

{Hector.Ruiz,Didier.Bert}@imag.fr

RÉSUMÉ. Nous présentons une approche à la spécification, preuve et raffinement de propriétés de vivacité sous hypothèse d'équité faible dans le B événementiel. Les propriétés de vivacité sont divisées en deux classes : propriétés de base et propriétés générales. Nous donnons des obligations de preuve fondées sur le calcul des plus faibles préconditions pour prouver les propriétés de base. Les propriétés de vivacité générales sont prouvées par les définitions et théorèmes de la logique UNITY. Nous donnons les obligations de preuve de la préservation des propriétés de vivacité dans le raffinement. Notre démarche est illustrée par un exemple d'un système de gestion de processus

ABSTRACT. We present an approach to specification, proof and refinement of liveness properties under weak fairness assumptions in B event systems. Liveness properties are divided in two classes: basic properties and general properties. We give proof obligations by calculus of weakest preconditions in order to prove basic properties. General liveness properties are proved by definitions and theorems of the logic programming of UNITY. We give proof obligations of the preservation of liveness properties in the refinement. Our approach is showed by an example of a system modeling management of processes.

MOTS-CLÉS : B événementiel, UNITY, propriétés de vivacité, équité faible.

KEYWORDS: Event B, UNITY, liveness properties, weak fairness

1. Introduction

Dans le domaine de la spécification des systèmes, les propriétés de *sûreté*, ou propriétés invariantes, décrivent les conditions en dehors desquelles le système ne peut pas fonctionner normalement. Les propriétés de *vivacité* ou dynamiques, décrivent ce qui doit se passer de bien lorsque le système fonctionne. Elles prennent souvent la forme de conditions qui indiquent que si le système se trouve dans un certain état, alors, inévitablement, il atteindra un autre état attendu. Il existe beaucoup de formalismes pour exprimer les propriétés dynamiques, en particulier les logiques temporelles, linéaires ou arborescentes. La preuve de ces propriétés sur des systèmes finis est réalisée à l'aide de la technique de *model-checking*, c'est-à-dire en examinant tous les états du système. Des langages ou formalismes de spécification tels que TLA [LAM 94] ou UNITY [CHA 88] permettent également d'exprimer des propriétés de sûreté et des propriétés dynamiques. Dans ces cas-là, des systèmes logiques ont été définis dans le but de prouver les propriétés par déduction. De manière similaire, en [ABR 98] Abrial et Mussat introduisent des propriétés dynamiques dans les systèmes d'événements en **B** ainsi que les obligations de preuve associées, basées sur celles de la terminaison d'une boucle.

Lorsque les systèmes sont décrits par des événements ou des actions de manière non déterministes, les comportements possibles peuvent être très divers. Pour assurer les propriétés de vivacité, il faut alors, le plus souvent, soit ajouter des restrictions effectives dans les comportements du système, ce qui peut être difficile ou même impossible dans le cas où l'on modélise les comportements de l'environnement, soit imposer des *hypothèses d'équité*.

Le problème de l'équité dans les systèmes a été abordé depuis longtemps. L'équité peut s'exprimer sous différentes formes (faible, forte, etc.). Pour une analyse de l'équité, on peut se reporter à [FRA 86]. Dans les systèmes de transition avec équité, la vérification des propriétés de vivacité se fait par *model-checking*, souvent adapté pour prendre en compte les hypothèses d'équité par l'élimination des séquences d'exécution qui ne respectent pas ces hypothèses. Les difficultés principales de ces approches sont soit la complexité de l'algorithme, soit l'explosion du nombre d'états de l'automate. Pour avoir un état de l'art de ces techniques, on peut se reporter à [CHO 03].

Les formalismes TLA, UNITY et celui des systèmes d'actions [BAC 98] donnent la possibilité de tenir compte des hypothèses d'équité et définissent les obligations de preuve de propriétés de vivacité en présence de ces hypothèses. Dans le **B** événementiel [ABR 98], il n'y a pas de possibilité d'imposer des hypothèses d'équité. En revanche, les exemples montrent souvent la nécessité d'introduire un ordonnanceur abstrait dans les événements. Les conditions de raffinement des systèmes événementiels **B** assurent que les propriétés de vivacité du niveau supérieur sont préservées.

Dans [RUI 02], nous avons présenté une façon d'exprimer des propriétés de vivacité en **B**, en nous basant sur la logique de UNITY. Les propriétés sont divisées en propriétés "atomiques" (appelées aussi propriétés de base) et propriétés "générales". Les propriétés générales se démontrent dans le système logique de UNITY. Nous avons

défini les obligations de preuves pour démontrer les propriétés atomiques sans hypothèse d'équité (condition de *progrès minimal*) et avec une hypothèse d'*équité faible* (c'est-à-dire, si un événement est continûment activable, alors il est infiniment souvent activé).

Cet article est une version étendue de [RUI 04]. Nous rappelons d'abord les caractéristiques du B événementiel, ainsi que les principaux aspects de la logique UNITY et les considérations à prendre en compte pour son utilisation dans le B événementiel (paragraphe 2). Nous reformulons et généralisons les obligations de preuve pour les propriétés de vivacité atomiques avec hypothèse d'équité faible (paragraphe 3). Nous donnons une manière de prendre en compte les événements à non déterminisme non borné (substitution ANY) qui n'était pas traitée en [RUI 02] (paragraphe 3.1.2). Dans le paragraphe 4, nous donnons les obligations de preuve de préservation des propriétés de vivacité de base dans le raffinement. Là encore, nous examinons et traitons le cas où les événements ANY ont deux comportements distincts par rapport à la propriété à démontrer. Un aperçu de la vérification des obligations de preuve proposées est donné dans le paragraphe 4.4. Dans le paragraphe 5, nous développons un exemple de spécification d'un gestionnaire de processus qui doit donner l'accès à une ressource critique, avec l'hypothèse d'équité faible. La propriété de vivacité est que les processus qui ont demandé la ressource critique finissent par l'obtenir. Cet exemple est raffiné deux fois pour définir la politique d'accès à la ressource. Nous montrons quelles sont les obligations de preuve qui assurent la préservation de la propriété de vivacité du niveau abstrait. Nous comparons ensuite notre approche avec les obligations de preuve de Abrial-Mussat [ABR 98] et celles de Back-Xu [BAC 98] au paragraphe 6. Nous donnons enfin dans la conclusion des perspectives pour une poursuite de ce travail.

2. B et la logique UNITY

Dans ce paragraphe rappelons d'abord les caractéristiques du B événementiel. Nous présentons ensuite les aspects principaux de la logique UNITY nécessaires pour nos recherches. Particulièrement nous montrons les spécificités de la logique UNITY et les considérations à prendre en compte pour son utilisation dans la spécification et la preuve des propriétés de vivacité dans le cadre du B événementiel.

2.1. B événementiel

Le B événementiel est une extension du B classique qui permet de décrire des systèmes d'événements. Un programme en B événementiel comprend des déclarations statiques (ensembles, constantes), un état spécifié par des variables, une initialisation de l'état et un ensemble fini d'événements. Les événements sont constitués d'une *garde*, qui est un prédicat, et d'une *action*. Dans un état donné, un événement est dit *habilité* si et seulement si sa garde est vraie. Il peut alors être déclenché, ce qui signifie que la partie action de l'événement est exécutée, produisant ainsi un changement

atomique¹ de l'état. Parmi tous les événements habilités, le choix de l'événement qui se déclenche est *non déterministe*.

Les événements sont décrits à l'aide du langage des *substitutions généralisées* [ABR 96]. Dans cet article, nous utilisons deux formes d'événements :

SELECT P THEN S END	événement avec garde déterministe
ANY z WHERE P THEN S END	événement avec garde non déterministe

où P est un prédicat, S une substitution généralisée d'affectation et z une liste de variables différentes des variables de l'état avec la restriction que $\{z \mid P\}$ est un ensemble fini. Nous utilisons aussi les raccourcis : BEGIN S END qui est équivalent à SELECT *true* THEN S END et $x := \alpha$, qui signifie que x prend une valeur quelconque de l'ensemble fini α . Elle est équivalente à la substitution généralisée ANY z WHERE $z \in \alpha$ THEN $x := z$ END. Les affectations sont de la forme $x := \varphi$, avec x une variable et φ une expression, ou bien $x, y, \dots := \varphi, \psi, \dots$ qui signifie l'affectation simultanée de φ à x , ψ à y , etc.

Pour chaque sorte d'événement, on peut calculer la garde (notion de faisabilité dans le B-Book) :

$grad(\text{SELECT } P \text{ THEN } S \text{ END})$	\Leftrightarrow	P
$grad(\text{ANY } z \text{ WHERE } P \text{ THEN } S \text{ END})$	\Leftrightarrow	$\exists z \cdot P$

Pour illustrer ces notions, nous donnons un exemple de petit système :

```

SYSTEM Petit_Systeme
VARIABLES  $x$ 
INVARIANT  $x \in \mathbb{Z}$ 
INITIALISATION
   $x := -1$ 
EVENTS
   $e_1 \hat{=} \text{SELECT } x < 0 \text{ THEN } x := x - 1 \text{ END}$ 
   $e_2 \hat{=} \text{BEGIN } x := 0 \text{ END}$ 
END

```

On remarque que l'événement e_2 est toujours habilité. Les comportements possibles du *Petit_Systeme* sont les suivants :

- 1) l'événement e_1 est exécuté indéfiniment.
- 2) l'événement e_1 est exécuté zéro ou plusieurs fois ; puis e_2 s'exécute indéfiniment.

Ces comportements correspondent au fait que, pour des systèmes d'événements B tels qu'ils sont définis dans [ABR 98], la seule condition pour le déclenchement d'un

1. c'est-à-dire non interruptible.

événement est que sa garde soit vraie. Cette condition est appelée hypothèse de *progress minimal*. Un autre type d'hypothèse de fonctionnement est celle de l'*équité faible*. Elle consiste à dire que si un événement est continûment habilité, alors il est infiniment souvent activé. Le comportement du *Petit_Systeme* avec hypothèse d'équité faible est 2. La séquence 1 ne respecte pas cette hypothèse.

La sémantique des substitutions généralisées est donnée par le calcul de la *plus faible précondition*. On note $[S]R$ le plus faible prédicat sur l'état qui doit être vrai avant l'exécution de S pour que S termine dans un état qui satisfait le prédicat R . Pour les substitutions généralisées introduites ici, on a [ABR 96] :

Expression de wp	Résultat
$[x := \varphi] R$	remplacement de x par φ dans R
$[x, y, \dots := \varphi, \psi, \dots] R$	remplacement simultané de x par φ , y par ψ, \dots dans R
$[\text{SELECT } P \text{ THEN } S \text{ END}] R$	$P \Rightarrow [S] R$
$[\text{ANY } z \text{ WHERE } P \text{ THEN } S \text{ END}] R$	$\forall z \cdot (P \Rightarrow [S] R)$

Dans la dernière ligne ci-dessus, les variables de la liste z ne doivent pas être *libres*² dans R . En **B**, un système abstrait \mathcal{S} avec déclarations D , invariant I , initialisation U et événements $e_i \hat{=} E_i$ pour un ensemble fini d'indices $i \in L$, est *bien formé* si les conditions suivantes sont satisfaites :

- 1) l'initialisation établit l'invariant : $D \Rightarrow [U] I$
- 2) chaque événement e_i termine et préserve l'invariant : $D \wedge I \Rightarrow [E_i] I$
- 3) le système d'événements ne se bloque pas³ : $D \wedge I \Rightarrow \bigvee_{i \in L} \text{grad}(E_i)$

Nous utiliserons la notion de *choix borné* de substitutions, noté $T \parallel U$. La sémantique du choix est donnée par : $[T \parallel U]R \Leftrightarrow [T]R \wedge [U]R$. Nous noterons par S le choix borné d'événements $e_i \hat{=} E_i$ du système \mathcal{S} : $S = (\parallel_{i \in L} e_i)$ ou $S = (\parallel_{i \in L} E_i)$ avec l'extension de calcul : $[S]R \Leftrightarrow [\parallel_{i \in L} e_i]R \Leftrightarrow [\parallel_{i \in L} E_i]R$. D'autre part, dans la suite de cet article, nous dénotons les substitutions généralisées par A, C, F, G et H , les prédicats par P, Q et R , et les invariants par I et J .

2.2. La logique UNITY

UNITY a été proposé dans [CHA 88] comme une théorie pour le développement des programmes parallèles. Cette théorie comporte trois parties : une logique pour la spécification et preuve des propriétés des programmes, une notation de programmation abstraite pour décrire le comportement dynamique des programmes et une série

2. Une variable v est libre dans un prédicat P si elle apparaît dans P et qu'elle n'est pas sous la portée d'un quantificateur.

3. Cette condition est parfois donnée comme facultative [ABR 98].

de règles permettant de traduire cette notation en une notation concrète, proche de l'architecture matérielle où les programmes seront mis en œuvre. Dans les paragraphes suivants nous donnons un aperçu des aspects principaux de cette théorie concernant notre travail.

En UNITY, le développement d'un programme commence par la spécification abstraite des propriétés de vivacité et de sûreté. La spécification ne tient pas compte des contraintes de mise en œuvre ; ces contraintes sont introduites au fur et à mesure des raffinements de la spécification. Il doit être démontré que les propriétés initiales sont préservées tout au long des raffinements. La logique UNITY dispose d'un système de preuve permettant de démontrer la préservation des propriétés. Lorsque la spécification comporte des détails suffisants de mise en œuvre, un programme UNITY peut être dérivé. La notation de programmation UNITY est fondée sur l'exécution non déterministe et équitable des instructions d'affectation. De ce fait, un programme UNITY ne comporte pas de flot de contrôle et ne peut pas être mis en œuvre directement sur des machines conventionnelles. Pour cette raison, UNITY propose une série de règles informelles permettant de grouper ces instructions en fonction de l'architecture de la machine cible, à mémoire partagée ou à transfert de messages, pour pouvoir les exécuter.

La logique UNITY est une classe de logique temporelle linéaire conçue pour spécifier et prouver des propriétés de vivacité et de sûreté des programmes UNITY. Dans la proposition originale, le langage de spécification est structuré autour des formules temporelles construites à partir de trois relations fondamentales entre prédicats sur l'état du programme : *unless*, *ensures* et *leads-to*. Une propriété $P \text{ unless } Q$ décrit une propriété de sûreté ; cette propriété est vraie dans un programme si le prédicat P est vrai dans un état du calcul et continue à être vrai jusqu'à ce que Q devienne vrai. Les relations *ensures* et *leads-to* sont utilisées pour spécifier des propriétés de vivacité. Une propriété $P \text{ ensures } Q$ est vraie dans un programme si lorsque le calcul est dans un état où P est vrai, alors il arrive fatalement dans un autre état où Q devient vrai et P continue à être vrai jusqu'à ce que Q devienne vrai. La sémantique d'une propriété $P \text{ leads-to } Q$ est similaire à celle de $P \text{ ensures } Q$, cependant en $P \text{ leads-to } Q$ nous ne pouvons pas assurer que P reste vrai tandis que Q ne l'est pas. Une série de théorèmes a été développée pour ces relations fondamentales, ce qui permet d'avoir un système de preuve dans cette logique.

Un cas spécial de propriété de sûreté permet de spécifier des propriétés invariantes. Ces propriétés sont exploitées par l'axiome de substitution de la logique UNITY. Cet axiome indique qu'un prédicat invariant peut être remplacé par le prédicat *true* et vice-versa, dans n'importe quelle propriété. L'utilisation de l'axiome de substitution et les définitions originales des relations fondamentales dans [CHA 88] rendent incohérente (*unsound*) la logique UNITY, comme il a été rapporté dans [SAN 91a]. Dans cette même référence, et afin de résoudre les problèmes de la logique originale, une redéfinition des relations fondamentales est proposée fondée sur l'idée du « plus fort invariant » (*strongest invariant*). Par ailleurs, dans [SAN 91a], est aussi introduit le concept de propriétés sous-indicées, qui ont pour but de généraliser les relations fon-

damentales *unless* et *ensures*, ce qui permet d'éliminer l'axiome de substitution de la logique. Cependant dans [PRA 94], il est rapporté une erreur dans ces définitions et des nouvelles définitions des propriétés sous-indicées sont données qui résolvent les problèmes rencontrés. Comme nous n'utilisons pas l'axiome de substitution, ni les propriétés sous-indicées dans notre travail, nous ne détaillons pas ces résultats.

Dans [MIS 95b] et [MIS 95a] la logique UNITY a été mise à jour. Les relations fondamentales *unless* et *ensures* ont été redéfinies par de nouvelles relations. Cependant les changements n'affectent pas la spécification et la preuve des propriétés de vivacité. Tous les développements que nous présentons sont donc cohérents avec cette nouvelle version.

2.3. Extensions de la logique UNITY en B événementiel

Les définitions des relations *unless* et *ensures* dans [CHA 88] ou dans [SAN 91a], sont faites par des triplets de Hoare, ou de manière équivalente par des plus faibles préconditions, quantifiées universellement ou existentiellement sur l'ensemble d'instructions d'un programme UNITY. Un programme UNITY est formé d'un nombre fini d'instructions d'affectation (multiple) pouvant être conditionnées par des expressions booléennes. Les démonstrations en [CHA 88] des théorèmes des relations *unless* et *ensures* exploitent les propriétés des plus faibles préconditions de ce type d'instructions, particulièrement la distributivité par rapport à la conjonction. Cependant, le théorème d'impossibilité (*impossibility theorem*) de la relation *ensures* exige que toutes les instructions respectent la loi du miracle exclu⁴.

La plus faible précondition des événements dans le B événementiel sont distributives par rapport à la conjonction. Pour cette raison, des définitions des relations *unless* et *ensures*, similaires à celles dans [SAN 91a], où les plus faibles préconditions sont quantifiées sur l'ensemble d'événements d'un système B événementiel, satisfont plusieurs des théorèmes de la logique UNITY. Cependant, les événements modélisés par des instructions SELECT ou ANY, qui introduisent des commandes gardées, ne respectent pas la loi du miracle exclu. Pour cette raison, nous devons garantir que la définition de la relation *ensures* dans le B événementiel satisfait le théorème d'impossibilité ; cela est argumenté dans le paragraphe suivant, après la définition de la relation *ensures* ; ce résultat a déjà été évoqué dans [SAN 91b].

Un programme UNITY qui satisfait une propriété $P \text{ ensures } Q$ garanti, par définition de la relation *ensures*, l'existence d'une instruction utile capable d'établir la postcondition Q lorsqu'elle commence son exécution dans un état où le prédicat P est vrai. L'exécution d'une telle instruction est assurée par une hypothèse d'équité : n'importe quelle instruction dans un programme UNITY est infiniment souvent exécutée. Dans l'extension en B, comme nous le montrons dans le paragraphe suivant, la défi-

4. Une instruction A respecte la loi du miracle exclu si $[A] \text{ false} \Leftrightarrow \text{false}$.

inition de la relation *ensures* affirme l'existence d'un événement utile. Cependant, du fait que l'exécution d'un événement est conditionnée par l'état de sa garde, nous adoptons une hypothèse d'équité faible : tout événement dont la garde est continuellement habilitée est exécuté infiniment souvent.

Du point de vue sémantique nous considérons que n'importe quel événement e de la forme ANY x WHERE P_x THEN A_x END, décrit une famille d'événements e_x , $e_x = \text{SELECT } P_x \text{ THEN } A_x \text{ END}$, où x est quantifié sur l'ensemble des valeurs qui satisfont les contraintes P_x . De cette manière, l'exécution non déterministe de e est considérée comme le choix non déterministe de l'exécution des événements déterministes e_x . Cependant, afin de garantir que n'importe quel événement e_x sera finalement exécuté lorsque sa garde reste continuellement habilitée, nous avons restreint à un ensemble fini l'ensemble des valeurs introduites par des constructions de type ANY.

Dans notre travail, nous n'utilisons pas la relation *unless* pour spécifier des propriétés de sûreté, cependant une définition identique à celle de UNITY, comme il est défini dans [SAN 91a], est aussi appropriée dans notre contexte. Les considérations que nous venons de faire sur les événements de type ANY et la définition de notre relation *ensures* présentée dans le paragraphe 3.1.1 nous permettent de conclure la validité de tous les théorèmes et l'application de la logique UNITY dans le cadre du B événementiel.

3. Propriétés de vivacité en B événementiel

Nous présentons dans cette partie notre approche de la spécification et de la preuve de propriétés de vivacité sous l'hypothèse d'équité faible.

Les propriétés de vivacité sont divisées en deux classes : propriétés de vivacité de base et propriétés de vivacité générales. Les propriétés de vivacité de base permettent de spécifier des transitions particulières effectuées par l'activation d'un événement, tandis que les propriétés de vivacité générales spécifient des transitions réalisées par l'exécution de plusieurs événements.

3.1. Propriétés de vivacité de base

Les propriétés de vivacité de base sont spécifiées par des formules de la forme $G \cdot P \gg_w Q$ (prononcer : « par l'événement G , P assure Q »), où P et Q sont des prédicats sur l'état du système et G un événement ou un choix d'événements. Un système vérifie une telle propriété si G est habilité (c'est-à-dire, sa garde est vraie) dans un état où le prédicat $P \wedge \neg Q$ est vrai et s'il est capable d'établir le prédicat Q lors de son activation et, de plus, tous les autres événements du système préservent le prédicat P ou établissent Q . Le rôle de notre supposition d'équité faible est primordial dans ce type de propriétés. En effet, si le système dispose d'un événement habilité dans un état où P est vrai qui est capable d'établir Q , et en plus l'activation de tous

les autres événements du système préservent P , par notre supposition d'équité, cet événement sera finalement activé en établissant Q .

Comme illustration, nous pouvons vérifier que, avec l'hypothèse d'équité faible, la propriété $e_2 \cdot true \gg_w x = 0$ est valide dans le *Petit_Systeme*.

3.1.1. Les obligations de preuve

Les obligations de preuve pour une propriété $G \cdot P \gg_w Q$ se placent dans le cas d'un système \mathcal{S} avec invariant I composé d'événements F_i , où i appartient à un certain ensemble fini d'indices $L : S = \prod_{i \in L} F_i$. La syntaxe et la sémantique des événements F_i est celle décrite dans le paragraphe 2.1. Soit K un sous-ensemble non vide de L et G l'événement : $G = \prod_{i \in K} F_i$. Les obligations de preuve dans un système \mathcal{S} pour une propriété de vivacité de base avec hypothèse d'équité faible sont⁵ :

	ANTECEDENT	CONSEQUENT
WF0	$I \wedge P \wedge \neg Q \Rightarrow [S] (P \vee Q)$	$G \cdot P \gg_w Q$
WF1	$I \wedge P \wedge \neg Q \Rightarrow \text{grd}(G) \wedge [G] Q$	

L'obligation de preuve WF0 sert à garantir que tous les événements du système \mathcal{S} préservent P ou établissent Q , tandis que WF1 permet de reconnaître l'événement habilité qui établit la postcondition Q . Cet événement est qualifié d'"utile" (*helpful*) dans la littérature.

Nous devons noter que les obligations de preuve WF0 and WF1 ne donnent pas une définition de la relation \gg_w . Cette relation correspond à la relation *ensures* de la logique UNITY. Nous définissons la relation \gg_w d'une manière similaire à celle présentée en [SAN 91a] ; la définition est fondée sur l'idée du plus fort invariant (*strongest invariant*) SI caractérisant les états atteignables d'un système :

$$G \cdot P \gg_w Q \Leftrightarrow (SI \wedge P \wedge \neg Q \Rightarrow [S] (P \vee Q)) \wedge (SI \wedge P \wedge \neg Q \Rightarrow \text{grd}(G) \wedge [G] Q)$$

Du fait que $SI \Rightarrow I$, les antécédents WF0 and WF1 garantissent cette définition.

Comme nous l'avons anticipé dans le paragraphe 2.3, la définition de notre relation de vivacité de base satisfait tous les théorèmes de la relation *ensures* dans [CHA 88], particulièrement le théorème d'impossibilité qui, dans notre notation, est donné par

$$\frac{G \cdot P \gg_w false}{\neg P}$$

En effet, par la substitution de P et *false* dans la définition de la relation \gg_w , pour n'importe quel événement G , nous déduisons $SI \Rightarrow \neg P$. Du fait que SI est un in-

5. Dans ce tableau, l'antécédent de la règle est formé par la conjonction des implications WF0 et WF1.

riant et en utilisant la règle de substitution définie dans [PRA 94], nous concluons SI , et par modus ponens $\neg P$.

3.1.2. Séparation d'événements non déterministes

Dans le paragraphe précédent, le choix de l'événement G est fait sur la base "syntaxique" des événements déclarés dans le système \mathcal{S} . Lorsque le système comporte des événements non déterministes de la forme : $e \hat{=} \text{ANY } x \text{ WHERE } P_x \text{ THEN } A_x \text{ END}$ où P_x est un prédicat dans l'état du système où la variable x est libre et A_x est une expression dans le langage des substitutions généralisées qui contient x libre, il est possible que cet événement ait des comportements qui préservent P sans forcément établir Q et d'autres comportements qui établissent Q . Dans ce cas, il est nécessaire de trouver un renforcement de la garde R_x qui assure que l'événement e établit Q , le complémentaire devant préserver P . L'événement e est alors décomposé en deux événements du point de vue du comportement : $e = e_1 \parallel e_2$ où e_1 et e_2 sont les événements suivants :

$$e_1 \hat{=} \text{ANY } x \text{ WHERE } P_x \wedge R_x \text{ THEN } A_x \text{ END} \quad [1]$$

$$e_2 \hat{=} \text{ANY } x \text{ WHERE } P_x \wedge \neg R_x \text{ THEN } A_x \text{ END} \quad [2]$$

Afin de référencer le découpage d'événements e_1 et e_2 dans les obligations de preuve WF0 et WF1, nous introduisons les notations suivantes :

$$e_1 = e \text{ if } R_x \quad [3]$$

$$e_2 = e \text{ if } \neg R_x \quad [4]$$

Ainsi e_1 devient l'événement utile G dans l'obligation de preuve WF1. L'événement e_2 est considéré par l'obligation de preuve WF0 et correspond aux événements du système qui préservent le prédicat $P \wedge \neg Q$. Nous remarquons que, du fait que les plus faibles préconditions des événements e et $(e_1 \parallel e_2)$ sont les mêmes, un système \mathcal{S} contenant un ensemble d'événements \mathcal{E} , où $e \in \mathcal{E}$, est équivalent à un système \mathcal{S}' contenant l'ensemble d'événements $(\mathcal{E} - \{e\}) \cup \{e_1, e_2\}$. Finalement nous notons que l'ensemble d'événements d'un système reste toujours fini, car nous limitons à un ensemble fini les valeurs des variables introduites par les événements de type ANY.

3.2. Propriétés de vivacité générales

Les propriétés de vivacité générales sont spécifiées par des formules de la forme $P \rightsquigarrow Q$ (prononcer : « P conduit à Q »). Cette formule spécifie qu'un système atteint fatalement un état Q lorsque son exécution arrive dans un état où P est vrai. Il y a trois différences fondamentales entre ce type de propriété et une propriété de vivacité de base :

1) La transition d'états de P à Q peut se faire par l'activation d'un ou plusieurs événements atomiques.

2) Une propriété générale ne garantit pas que le prédicat P est préservé tant que le système n'a pas atteint l'état où Q est vrai.

3) La définition des propriétés générales ne dépend pas directement des hypothèses d'équité comme les propriétés de base.

Une propriété $P \rightsquigarrow Q$ est vraie dans un système si elle peut être déduite par un nombre fini d'application des règles suivantes :

	ANTECEDENT	CONSEQUENT
BRL	$G \cdot P \gg_w Q$	$P \rightsquigarrow Q$
TRA	$P \rightsquigarrow R, R \rightsquigarrow Q$	$P \rightsquigarrow Q$
DSJ	$\forall m \cdot (m \in M \Rightarrow P_m \rightsquigarrow Q)$	$(\bigvee_{m \in M} P_m) \rightsquigarrow Q$

Dans la règle DSJ, M dénote n'importe quel ensemble dont ses éléments servent à indexer une famille de prédicats P_m . En utilisant les mêmes définitions des propriétés de vivacité de la théorie UNITY, nous pouvons tirer profit des théorèmes de cette théorie, définis dans [CHA 88], afin de raisonner sur les propriétés de vivacité dans le cadre du B événementiel. Comme un exemple de ces théorèmes nous présentons les théorèmes d'induction et d'élimination (*cancellation*) :

	ANTECEDENT	CONSEQUENT
IND	$\forall v \cdot ((P \wedge V = v) \rightsquigarrow (P \wedge V < v \vee Q))$	$P \rightsquigarrow Q$
CAN	$P \rightsquigarrow Q \vee R, R \rightsquigarrow R'$	$P \rightsquigarrow Q \vee R'$

Le théorème d'induction IND mérite quelques commentaires. Il utilise un variant V défini sur un ensemble bien fondé. Par cette règle, afin de prouver la propriété $P \rightsquigarrow Q$ nous devons montrer que le système, lorsqu'il arrive dans un état où le prédicat P est vrai, et le variant a une valeur v , atteint fatalement un état où le prédicat P continue à être vrai mais le variant décroît, ou le prédicat Q devient vrai.

Les spécifications des propriétés de vivacité dans un système B événementiel seront données par des relations \rightsquigarrow . Lorsque nous voulons prouver que des telles propriétés sont vraies dans un système donné, nous montrons qu'il existe des propriétés de base qui sont vraies dans le système et nous appliquons les règles nécessaires pour montrer la propriété souhaitée. A leur tour, les propriétés de base doivent être prouvées en utilisant les obligations de preuve WF0 et WF1 si l'on fait des hypothèses d'équité faible.

4. Raffinement des propriétés de vivacité

Dans ce paragraphe nous étudions le raffinement des propriétés de vivacité. Nous considérons un système S et une propriété de vivacité générale \mathcal{P} . Nous nous inté-

ressons à la préservation de la propriété \mathcal{P} lorsque le système \mathcal{S} est raffiné dans un système \mathcal{T} .

Comme nous l'avons indiqué dans le paragraphe 3.2, la preuve de \mathcal{P} dans \mathcal{S} dépend d'un certain nombre de propriétés de base. Afin de simplifier, nous supposons que \mathcal{P} dépend uniquement d'une propriété de base \mathcal{Q} . Ainsi, lorsque le système \mathcal{S} est raffiné en le système \mathcal{T} , nous devons montrer la préservation de la propriété \mathcal{P} . Or, la propriété \mathcal{P} ne dépendant que de la propriété de base \mathcal{Q} , nous devons garantir que cette dernière propriété est préservée dans le raffinement \mathcal{T} . A cause des nouveaux événements dans le raffinement, une propriété atomique abstraite \mathcal{Q} devient une propriété générale concrète. Cependant, il n'est pas nécessaire de faire la preuve complète de cette propriété concrète, car la propriété abstraite a été démontrée. Il suffit de donner les conditions de préservation de la propriété \mathcal{Q} dans le système raffiné.

Dans la suite, nous présentons d'abord rapidement le raffinement de systèmes en \mathbf{B} , ensuite les obligations de preuve de préservation d'une propriété de base avec hypothèse d'équité faible et enfin le cas de raffinement avec événements non déterministes disjoints.

4.1. Raffinement de systèmes \mathbf{B}

Le raffinement en \mathbf{B} est le moyen de détailler un modèle en s'assurant que les propriétés du niveau abstrait sont préservées dans le niveau concret. Le raffinement étant transitif, un utilisateur peut développer son modèle par une chaîne de raffinements. Nous donnons maintenant les conditions d'un raffinement en \mathbf{B} -événementiel, telles qu'elles sont présentées en [ABR 98].

Un système \mathcal{S} avec déclarations D , variables x spécifiées par un invariant I , initialisation U et ensemble d'événements $(e_i \hat{=} E_i)_{i \in L}$ est raffiné par un « raffinement de système » \mathcal{T} avec déclarations D' , variables y spécifiées par J , initialisation U' et événements $(e_i \hat{=} E'_i)_{i \in L} \cup (h_k \hat{=} H_j)_{k \in K}$ si les conditions suivantes sont remplies :

1) Le prédicat J spécifie les variables y et définit la relation entre les variables y et x . C'est une relation de *collage* qui s'exprime formellement par $v = \{y, x \mid I \wedge J\}$.

2) L'initialisation et les événements e_i du raffinement raffinent l'initialisation et les événements de même nom e_i du système abstrait, respectivement. La relation de raffinement entre substitutions généralisées A et C par un invariant de collage J est notée $A \sqsubseteq_{I \wedge J} C$.

3) Il y a éventuellement de nouveaux événements h_k dans le système raffiné. Ces événements raffinent la substitution sans effet « skip ».

4) Les nouveaux événements (h_k) du système raffiné ne doivent pas pouvoir prendre le contrôle indéfiniment (pas de *livelocks*).

5) La vivacité du système abstrait doit être préservée dans le système raffiné (pas d'introduction de blocage).

Nous ne détaillons pas les conditions 4 et 5. Le lecteur intéressé peut se reporter au document de référence. L'obligation de preuve qui assure que la relation de raffinement est valide entre deux substitutions généralisées dans le contexte des systèmes \mathcal{S} et \mathcal{T} , sous les hypothèses des déclarations statiques D et D' , est définie par :

$$A \sqsubseteq_{I \wedge J} C \Leftrightarrow (I \wedge J \Rightarrow [A] \neg [C] \neg J)$$

Pour les nouveaux événements h_k , la condition de raffinement de skip se ramène à :

$$\text{skip} \sqsubseteq_{I \wedge J} C \Leftrightarrow (I \wedge J \Rightarrow [C] J)$$

ce qui est la préservation de l'invariant J .

Dans notre approche, il n'est pas nécessaire d'imposer la condition 4. En effet, même si les nouveaux événements prennent le contrôle, l'événement utile des propriétés de base sera fatalement exécuté, en imposant que la garde concrète de l'événement utile devienne vraie et grâce à notre hypothèse d'équité faible. Cela est suffisant pour préserver les propriétés de base du modèle abstrait. De plus, la préservation de ces propriétés dans le raffinement garantit la préservation de la vivacité du système (condition 5). Au paragraphe 6, nous comparerons nos conditions de préservation avec celles de Abrial-Mussat [ABR 98].

4.2. Les obligations de preuve de préservation des propriétés de base

Soit Q une propriété de vivacité de base, de la forme $G \cdot P \gg_w Q$. Nous supposons que la preuve de Q dans le système \mathcal{S} avec variable d'état x et invariant I , considère S comme étant le choix entre les événements F et G : $S = F \parallel G$, où G est l'événement utile de la propriété Q et F le choix des événements de S différents de G . Lorsque le système \mathcal{S} est raffiné dans un système \mathcal{T} avec variable d'état y et invariant de collage J , les événements F et G sont raffinés dans \mathcal{T} en événements concrets F' et G' . En plus, de nouveaux événements H sont introduits dans \mathcal{T} . Selon le paragraphe précédent, les événements dans \mathcal{S} et \mathcal{T} sont mis en relation par la relation de raffinement : $F \sqsubseteq_{I \wedge J} F'$, $G \sqsubseteq_{I \wedge J} G'$ et $\text{skip} \sqsubseteq_{I \wedge J} H$. Par cette relation, n'importe quelle postcondition R , établie par les événements F et G dans \mathcal{S} , est aussi établie par les événements concrets F' et G' de \mathcal{T} sous la supposition de l'invariant de collage. De plus, les nouveaux événements H , ne modifiant pas les données abstraites, préservent aussi le prédicat R . De cette manière, si nous avons prouvé dans \mathcal{S} , par WF0 que $P \wedge \neg Q \Rightarrow [F](P \vee Q)$ est vrai, sous la supposition de l'invariant I , le choix d'événements $F' \parallel H$ dans le raffinement \mathcal{T} préserve aussi $P \vee Q$ dans l'espace d'états raffiné, sous la supposition de l'invariant de \mathcal{S} et l'invariant de collage J c'est à dire : $I \wedge J \wedge \neg Q \wedge P \Rightarrow [F' \parallel H] \exists x \cdot ((P \vee Q) \wedge I \wedge J)$. D'autre part, par WF1 nous avons prouvé $P \wedge \neg Q \Rightarrow [G] Q$ dans \mathcal{S} , et par la relation de raffinement, $I \wedge J \wedge \neg Q \wedge P \Rightarrow [G'] \exists x \cdot (Q \wedge I \wedge J)$ est aussi vrai dans \mathcal{T} . Cependant nous ne pouvons pas garantir $I \wedge J \wedge \neg Q \wedge P \Rightarrow \text{grd}(G')$ car nous avons $\text{grd}(G') \Rightarrow \exists x \cdot (I \wedge J \wedge \text{grd}(G))$ par la relation de raffinement (en effet, les gardes sont renforcées dans le raffinement). Pour cette raison, nous proposons deux nouvelles

obligations de preuve, suffisantes pour garantir que la propriété Q est préservée dans le raffinement \mathcal{T} .

La première obligation de preuve, nommée LIP (*Liveness Preservation*) indique que nous devons prouver que le raffinement \mathcal{T} doit faire fatalement une transition d'un état où $I \wedge J \wedge \neg Q \wedge P \wedge \neg \text{grad}(G')$ est vrai, dans un autre état où $\text{grad}(G')$ est vrai. De cette manière, l'événement concret G' sera habilité pour établir la postcondition Q dans l'espace d'états raffiné. Mais du fait que nous ne pouvons pas déterminer l'instant où G' sera habilité, nous devons aussi garantir que $\exists x \cdot (I \wedge J \wedge \neg Q \wedge P \wedge \text{grad}(G'))$ n'est pas falsifié par les événements F' et H . Cette garantie est donnée par la deuxième obligation de preuve nommée SAP (*Safety Preservation*). Le tableau suivant nous montre les obligations de preuve qui doivent être satisfaites par le raffinement \mathcal{T} :

LIP	$I \wedge J \wedge P \wedge \neg Q \wedge \neg \text{grad}(G') \rightsquigarrow \text{grad}(G')$
SAP	$I \wedge J \wedge P \wedge \neg Q \wedge \text{grad}(G') \Rightarrow [F' \parallel H] \text{grad}(G')$

Remarque :

Lorsque nous devons prouver dans un système raffiné \mathcal{T} la préservation d'une propriété de base $G \cdot P \gg_w Q$, vérifiée par les obligations de preuve WF0 et WF1 dans un système \mathcal{S} , et que l'événement utile G ne change pas dans \mathcal{T} ($G = G'$), les obligations de preuve LIP et SAP deviennent trivialement vraies. En effet, puisque $G = G'$, et selon WF1 $P \wedge \neg Q \Rightarrow \text{grad}(G)$, la partie gauche de la propriété LIP devient *false*. Alors, la propriété *false* $\rightsquigarrow Q$ est impliquée par $G \cdot \text{false} \gg_w Q$, qui est trivialement vraie. D'autre part, du fait que $P \wedge \neg Q$ est préservé par F dans \mathcal{S} , et que par WF1 $P \wedge \neg Q \Rightarrow \text{grad}(G)$, l'obligation de preuve SAP est vraie aussi trivialement.

4.3. Raffinement d'événements non déterministes disjoints

Dans le paragraphe 3.1.2 nous avons introduit une notation pour identifier dans un événement non déterministe e des comportements $e \text{ if } \neg R$ qui préservent un certain prédicat $P \wedge \neg Q$ et d'autres $e \text{ if } R$ qui établissent un autre prédicat Q , c'est à dire $e = (e \text{ if } R) \parallel (e \text{ if } \neg R)$. De cette manière $e \text{ if } \neg R$ est vu comme un événement dans le choix d'événements F de l'obligation de preuve WF0 tandis que $e \text{ if } R$ est un événement du choix G de l'obligation de preuve WF1, pour prouver une certaine propriété $G \cdot P \gg_w Q$. Dans ce paragraphe nous étudions les conditions suffisantes que doit avoir le raffinement e' de e afin de garantir que les comportements des événements $e \text{ if } \neg R$ et $e \text{ if } R$ sont préservés dans e' .

Comme nous voulons préserver les comportements de $e \text{ if } R$ et $e \text{ if } \neg R$ dans e' , nous devons trouver un renforcement R' de la garde concrète de e' et des événements $e' \text{ if } R'$ et $e' \text{ if } \neg R'$, selon les définitions [3] et [4], tels que $e' = (e' \text{ if } R') \parallel (e' \text{ if } \neg R')$

et en plus que $e \text{ if } R \sqsubseteq_{I \wedge J} e' \text{ if } R'$ et $e \text{ if } \neg R \sqsubseteq_{I \wedge J} e' \text{ if } \neg R'$. Ces conditions de raffinements sont garanties par la validité des implications suivantes :

$$I \wedge J \Rightarrow [e' \text{ if } R'] \neg[e \text{ if } R] \neg J \quad [5]$$

$$I \wedge J \Rightarrow [e' \text{ if } \neg R'] \neg[e \text{ if } \neg R] \neg J \quad [6]$$

Cependant, la condition de raffinement dans le B événementiel assure que si e' est un raffinement de e l'implication suivante est vraie : $I \wedge J \Rightarrow [e'] \neg[e] \neg J$. En remplaçant e par ses événements disjoints [1] et [2], l'implication est équivalente à : $I \wedge J \Rightarrow [e'] (\exists x \cdot (P \wedge R \wedge \neg[A] \neg J) \vee \exists x \cdot (P \wedge \neg R \wedge \neg[A] \neg J))$. A partir de cette implication, pour satisfaire les conditions de raffinement [5] et [6], nous devons garantir que $e' \text{ if } R'$ établit la postcondition $\neg \exists x \cdot (P \wedge \neg R \wedge \neg[A] \neg J)$, et que $e' \text{ if } \neg R'$ établit la postcondition $\neg \exists x \cdot (P \wedge R \wedge \neg[A] \neg J)$. Ces deux obligations de preuve supplémentaires sont exprimées par les implications suivantes :

$$I \wedge J \Rightarrow [e' \text{ if } R'] \forall x \cdot (P \wedge \neg R \Rightarrow [A] \neg J) \quad [7]$$

$$I \wedge J \Rightarrow [e' \text{ if } \neg R'] \forall x \cdot (P \wedge R \Rightarrow [A] \neg J) \quad [8]$$

Ainsi, la condition de raffinement et les OP [7] et [8] nous permettent de conclure le raffinement des événements disjoints [5] et [6].

Finalement, si dans une abstraction nous avons prouvé $e \text{ if } R \cdot P \gg_w Q$, en accord avec les obligations de preuve LIP et SAP, et suivant les preuves de [7] et [8], nous devons prouver $I \wedge J \wedge \neg Q \wedge P \wedge \neg(\text{grd}(e' \text{ if } R')) \rightsquigarrow \text{grd}(e' \text{ if } R')$ et $I \wedge J \wedge \neg Q \wedge P \wedge \text{grd}(e' \text{ if } R') \Rightarrow [e' \text{ if } \neg R' \parallel F' \parallel H] \text{grd}(e' \text{ if } R')$ pour garantir la préservation de la propriété de base dans le raffinement.

4.4. Argumentation sur la validité des obligations de preuve

Les obligations de preuve concernant la preuve des propriétés de base (WF0, WF1), ainsi que la preuve de préservation des propriétés de base dans le raffinement (SAP, LIP) ont été formellement vérifiées dans [RUI 05]. Dans ce rapport nous avons suivi une approche complètement ensembliste inspirée de [ABR 98], où l'atteignabilité d'un prédicat est démontrée par la terminaison d'une itération d'événements. Pour traiter l'équité, l'itération a été modélisée à l'aide d'un opérateur de choix équitable nommé *dovetail*, proposé dans [BRO 94]. Dans les paragraphes suivants nous donnons un résumé informel des résultats sur la validation des obligations de preuve. Cette explication est donnée en termes de prédicats.

Nous considérons un système \mathcal{S} avec choix borné d'événements S et un événement utile G . A l'aide de l'opérateur *dovetail* (∇) une boucle équitable est définie de la manière suivante :

$$Y(Q)(G) = \neg Q \Longrightarrow ((S ; Y(Q)(G)) \nabla G)$$

La garde de $Y(Q)(G)$ assure que l'itération avance lorsque le système se trouve dans un état où le prédicat Q est faux. L'opérateur *dovetail* garantit que la branche de l'itération $S ; Y(Q)(G)$ n'est pas infiniment exécutée et que G est fatalement exécuté si sa garde est toujours vraie.

Pour établir une propriété de base $G \cdot P \gg_w Q$, WF1 garantit que le système est dans un état où la garde de G est vrai si le système reste dans n'importe quel état où $P \wedge \neg Q$ sont vrais et en plus, que l'exécution de G dans ces états établit Q . D'autre part, par WF0, le système est garanti de rester dans un état où P est vrai lorsque Q ne l'est pas. De cette manière, l'exécution de S dans la branche de la boucle $S ; Y(Q)(G)$, préserve P et la garde de G , et, à chaque itération, l'exécution de G est faisable. Ces conditions sont suffisantes pour que l'opérateur *dovetail* assure l'exécution de G et ainsi, l'établissement de Q . Il a été calculé dans [RUI 05] que WF0 et WF1 sont des conditions suffisantes pour montrer que $P \wedge \neg Q$ impliquent la plus faible précondition de $Y(Q)(G)$ pour établir Q .

Le système T raffiné de S , conduit à un nouvel espace d'états où les états concrets sont liés aux états abstraits par un invariant de collage. Les prédicats P et Q de l'abstraction deviennent des prédicats P' et Q' respectivement dans l'espace du système raffiné. Ces prédicats sont vrais dans les états concrets qui sont liés par l'invariant de collage, aux états abstraits où les prédicats abstraits sont vrais.

Les événements du système raffiné T sont composés des événements qui raffinent les événements de S et des nouveaux événements qui raffinent *skip*. Le choix borné d'événements dans T qui raffinent les événements abstraits est dénoté S' et le choix borné des nouveaux événements est dénoté par H . De cette manière, la boucle raffinée devient

$$Y'(Q')(G') = \neg Q' \implies (((S' \parallel H) ; Y'(Q')(G')) \nabla G')$$

où G' correspond au raffinement de l'événement utile G .

Les conditions de raffinement garantissent que les implications suivantes sont vraies dans T :

$$\begin{aligned} P' \wedge \neg Q' &\Rightarrow [S' \parallel H] (P' \vee Q') \\ P' \wedge \neg Q' &\Rightarrow [G'] Q' \end{aligned}$$

Comme nous pouvons constater, WF0 est complètement préservé par raffinement, tandis que WF1 est partiellement préservé. En effet, le raffinement conduit au renforcement des gardes des événements raffinés, et $P' \wedge \neg Q'$ n'impliquent pas nécessairement la garde de G' . Pour cette raison WF0 et WF1 deviennent des conditions suffisantes pour garantir une correction *partielle* de l'établissement de Q' par $Y'(Q')(G')$.

Pour remédier au problème du renforcement des gardes dans le raffinement, nous avons montré dans [RUI 05] que la préservation totale et partielle de WF0 et WF1 respectivement, ainsi que l'obligation de preuve SAP sont des conditions suffisantes pour prouver la propriété de base suivante :

$$G' \cdot P' \wedge \text{grd}(G') \gg_w Q'$$

Cette propriété indique que le système raffiné va fatalement établir le prédicat Q' par l'exécution de G' lorsque T arrive dans un état où $P' \wedge \text{grd}(G')$ est vrai.

Finalement, cette dernière propriété et l'obligation de preuve LIP sont suffisants pour montrer que la propriété

$$P' \rightsquigarrow Q'$$

est vraie dans T , ce qui nous donne la preuve de préservation d'une propriété de base dans le système raffiné.

5. Un exemple de spécification et raffinement

Dans cette section, nous développons en B événementiel l'exemple d'un système de gestion de processus qui accèdent à une section critique. Nous commençons par spécifier d'une manière abstraite les transitions des processus sans tenir compte des politiques d'accès à la section critique. De plus, par une propriété de vivacité, nous spécifions que les processus ne doivent pas attendre indéfiniment l'entrée à la section critique. Nous montrons comment prouver cette propriété dans la spécification. Ensuite nous procédons à deux raffinements du système pour définir les conditions d'accès à la section critique. Evidemment, nous devons garantir la préservation de la propriété de vivacité spécifiée dans le premier niveau. Dans le premier raffinement nous introduisons un nouvel événement qui modélise un serveur de jetons. De cette manière un processus ne peut entrer en section critique que s'il dispose d'un jeton. Les obligations de preuve pour la préservation de la propriété de vivacité demandent de montrer que n'importe quel processus qui attend un jeton va l'avoir. Dans le dernier raffinement un autre nouvel événement est introduit qui modélise un système de communication. Cette abstraction du système de communication va permettre au serveur de jetons de recevoir les requêtes de jetons demandés par les processus. Afin de montrer la préservation de notre propriété de vivacité, nous devons prouver que toutes les requêtes envoyées par les processus arrivent au serveur de jetons.

5.1. Spécification abstraite du système

Nous considérons un système B événementiel Sys qui gère des processus dont les identificateurs appartiennent à un ensemble PR . Le système a une variable st pour enregistrer l'état des processus. Un processus peut être dans un état *inactif* (ID), en *attente* d'entrer dans la section critique (WT), ou *actif* dans la section critique (AC). Ainsi la variable st est du type $PR \rightarrow ST$ où $ST = \{ID, WT, AC\}$ est l'ensemble des états des processus. A ce niveau de spécification, le nombre maximum de processus dans la section critique n'est pas fixé. Dans le premier raffinement, nous imposerons qu'il n'y ait pas plus d'un seul processus dans la section critique. Les déclarations de l'état du système s'expriment en B par le texte de la figure 1.

Le système Sys contient trois événements : rqt , acq et rel . L'événement rqt est activé lorsqu'un processus dans l'état inactif demande l'accès à la section critique. Après

```

SYSTEM Sys
SETS
  PR;
  ST = {ID, WT, AC}
VARIABLES
  st
INVARIANT
  st ∈ PR → ST
INITIALISATION
  st := PR × {ID}

```

Figure 1. Etat du système abstrait de gestion de processus

l'activation de cet événement, le processus concerné change à l'état d'attente. Lorsque le processus entre dans la section critique, l'événement *acq* est activé et l'état du processus devient actif. Finalement quand un processus quitte la section critique, l'événement *rel* s'active et l'état du processus devient inactif. En considérant les ensembles des processus inactif *Idl*, en attente *Wtg* et actifs *Act* définis par $Idl = st^{-1}[\{ID\}]$, $Wtg = st^{-1}[\{WT\}]$ et $Act = st^{-1}[\{AC\}]$, les événements du système *Sys* sont décrits à la figure 2.

```

EVENTS
  rqd ≙ ANY p WHERE p ∈ PR ∧ p ∈ Idl THEN st(p) := WT END ;
  acq ≙ ANY p WHERE p ∈ PR ∧ p ∈ Wtg THEN st(p) := AC END ;
  rel ≙ ANY p WHERE p ∈ PR ∧ p ∈ Act THEN st(p) := ID END
END

```

Figure 2. Événements du système abstrait de gestion de processus

Nous voulons spécifier que tous les processus qui demandent l'accès à la section critique vont l'avoir finalement. Formellement cette propriété est énoncée par :

$$q \in Wtg \rightsquigarrow q \in Act \quad [9]$$

où *q* est quantifié universellement dans *PR*. Nous remarquons que *Sys* peut avoir d'autres propriétés de vivacité, cependant notre démarche considère les propriétés énoncées explicitement comme étant d'intérêt et garantit leur préservation au cours des raffinements.

Afin de démontrer que la propriété [9] est respectée dans le système *Sys* nous pouvons prouver qu'il possède la propriété de base suivante :

$$G \cdot q \in Wtg \gg_w q \in Act \quad [10]$$

pour un certain événement G de Sys . Par l'application de la règle de base BRL pour la relation \rightsquigarrow , nous concluons alors [9] à partir de [10]. Il reste maintenant à déterminer l'événement G : la preuve de la propriété [10] doit être faite par l'application des règles WF0 et WF1 définies dans le paragraphe 3.1. WF0 demande de prouver que tous les événements de Sys préservent $q \in Wtg$ ou établissent $q \in Act$. En effet, comme les événements rqt et rel n'enlèvent pas d'éléments de l'ensemble Wtg , le prédicat $q \in Wtg$ est préservé par ces événements. L'événement acq a des comportements qui préservent $q \in Wtg$ et un comportement qui établit $q \in Act$. Donc, sous l'hypothèse de l'invariant, la preuve WF0 est déchargée :

$$q \in Wtg \wedge q \notin Act \Rightarrow [rqt \parallel acq \parallel rel] (q \in Wtg \vee q \in Act)$$

WF1 requiert de trouver un événement habilité de Sys qui soit capable d'établir la postcondition $q \in Act$. En utilisant la notation introduite dans le paragraphe 3.1.2, nous renforçons la garde de acq par le prédicat $p = q$, et nous avons ainsi l'événement $acq \text{ if } p = q$ qui établit $q \in Act$ et qui est habilité dans l'état $q \in Wtg$. De cette manière WF1 est prouvée :

$$q \in Wtg \wedge q \notin Act \Rightarrow ([acq \text{ if } p = q] q \in Act) \wedge grd(acq \text{ if } p = q)$$

Dans le paragraphe 5.2, nous raffinons le système Sys pour préciser une politique particulière d'entrée à la section critique. Nous montrons comment ce raffinement préserve la propriété de vivacité (9).

5.2. Introduction d'un ordonnanceur abstrait

Le raffinement $Sys1$ du système Sys introduit un nouvel événement désigné par srv . Ce nouvel événement modélise un serveur qui donne le droit d'accès à la section critique aux processus demandant l'entrée. Le droit d'accès est représenté par un jeton ; le nombre de jetons est limité à un seul, de cette manière il y a un processus au maximum dans la section critique dans un instant donné. Le serveur dispose d'une queue où sont stockées les demandes d'accès. Lorsque l'événement srv est activé, le jeton est alloué au premier processus dans la queue. L'événement raffiné acq peut être activé lorsque le jeton est accordé à un processus en attente. Après l'activation de acq , la demande d'entrée du processus qui entre dans la section critique est enlevée de la queue du serveur. Lorsque le processus quitte la section critique l'événement rel est activé et le jeton est libéré.

L'allocation du jeton à un processus est enregistrée dans une variable tk , de type $PR \rightarrow BOOL$. Dans l'état initial $tk(r) = false$ pour tout r dans PR . Lorsque le jeton est alloué à un processus r , $tk(r)$ devient $true$. La queue du serveur est modélisée par une séquence injective sq de processus ($sq \in \text{iseq}(PR)$). Cette séquence contient tous les processus en état d'attente ($Wtg = \text{ran}(sq)$). Nous dénotons par Tk le sous-ensemble de processus qui ont le jeton alloué ($Tk = tk^{-1}[\{true\}]$). La cardinalité de Tk est inférieure ou égale à un, et cela est un invariant de $Sys1$. Le processus en état d'attente qui a le jeton est le premier processus dans la séquence ($Wtg \cap$

$Tk \subseteq \{\text{first}(sq)\}$). Le processus en état actif est le seul à avoir le jeton ($Act \subseteq Tk$). Finalement aucun processus en état inactif n'a le jeton ($Tk \subseteq PR - Idl$). Les événements de *Sys1* sont donnés à la figure 3.

```

EVENTS
  rqt ≐ ANY p' WHERE p' ∈ PR ∧ p' ∈ Idl
      THEN
        st(p'), sq := WT, sq ← p'
      END ;
  acq ≐ ANY p' WHERE p' ∈ PR ∧ p' ∈ Wtg ∧ p' ∈ Tk
      THEN
        st(p'), sq := AC, tail(sq)
      END ;
  rel ≐ ANY p' WHERE p' ∈ PR ∧ p' ∈ Act
      THEN
        st(p'), tk(p') := ID, false
      END ;
  srv ≐ SELECT sq ≠ [] ∧ Tk = ∅
      THEN
        tk(first(sq)) := true
      END
END

```

Figure 3. Événements du premier raffinement

La preuve de la préservation de [9] a besoin d'identifier dans *Sys1* le raffinement des événements acq if $p = q$ et acq if $p \neq q$. Or, ces événements sont raffinés par les événements concrets acq if $p' = q$ et acq if $p' \neq q$ respectivement. La preuve de ce raffinement est dans les obligations de preuve [7] et [8], que nous ne développons pas ici. Notons que, à partir de maintenant, les obligations de preuve de l'exemple (OP) sont plus fortes que ce qui est établi au paragraphe 4.2. La partie gauche des OP que nous donnons ici sont impliquées par la partie gauche des OP théoriques. La transitivité de \sim et \Rightarrow garantit donc le résultat. Nous faisons ce renforcement, qui ne nuit pas à la preuve, pour faciliter la lecture des formules. Moyennant cette remarque, les obligations LIP et SAP de préservation de [9] sont :

$$q \in Wtg \wedge \neg \text{grd}(acq \text{ if } p' = q) \sim \text{grd}(acq \text{ if } p' = q)$$

$$q \in Wtg \wedge \text{grd}(acq \text{ if } p' = q) \Rightarrow [rqt \parallel rel \parallel acq \text{ if } p' \neq q \parallel srv] \text{grd}(acq \text{ if } p' = q)$$

Le calcul de $\text{grd}(acq \text{ if } p' = q)$ nous donne $q \in Wtg \wedge q \in Tk$. Ainsi, les obligations de preuve deviennent :

$$q \in Wtg \wedge q \notin Tk \sim q \in Wtg \wedge q \in Tk \quad [11]$$

$$q \in Wtg \wedge q \in Tk \Rightarrow [rqt \parallel rel \parallel srv \parallel acq \text{ if } p' \neq q] q \in Wtg \wedge q \in Tk \quad [12]$$

Nous nous concentrons sur la preuve de la propriété de vivacité [11] car la preuve de la propriété de sûreté [12] est facilement déchargée par le calcul de prédicats et le calcul des substitutions. Cette propriété [11] dit qu'un processus qui est en attente et qui n'a pas le jeton finit par l'obtenir, lui permettant ainsi de rentrer dans la section critique, puisque les autres événements ne peuvent pas le lui enlever (propriété [12]). Comme le fait d'obtenir le jeton, pour un processus en attente, n'est pas immédiat dans le système, nous pouvons chercher à démontrer [11] en passant par l'application de la règle d'induction IND avec la propriété suivante :

$$q \in Wtg \wedge q \notin Tk \wedge V(q) = n \rightsquigarrow \\ (q \in Wtg \wedge q \notin Tk \wedge V(q) < n) \vee (q \in Wtg \wedge q \in Tk) \quad [13]$$

où $V(q)$ dénote un variant. La définition du variant peut avoir un effet important sur la complexité de la preuve. Ainsi par exemple, une solution est de définir $V(q)$ par un ordre lexicographique :

$$V(q) = [sq^{-1}(q) + \text{card}(sq[1..sq^{-1}(q)] - Tk), \text{card}(Act)]$$

où le premier composant est l'addition de la position du processus q dans la séquence et du nombre de processus qui attendent le jeton dans l'intervalle $1..sq^{-1}(q)$ et le deuxième composant est le nombre de processus dans la section critique. En utilisant ce variant, par l'application de WF0 et WF1, nous pouvons prouver :

$$(acq \parallel rel \parallel srv) \cdot q \in Wtg \wedge q \notin Tk \wedge V(q) = n \gg_w \\ (q \in Wtg \wedge q \notin Tk \wedge V(q) < n) \vee (q \in Wtg \wedge q \in Tk)$$

Ensuite, par l'application de la règle IND nous déduisons [13]. Cette preuve que nous venons d'esquisser est, bien sûr, faisable. Cependant, lorsque nous raffinerons *Sys1*, la préservation de (13) nécessite de traiter le choix d'événements $acq \parallel rel \parallel srv$ comme un seul événement dans les obligations de preuve LIP et SAP. Cette contrainte implique des formules plus longues et peut-être difficiles à prouver. De plus, la préservation de la propriété reste dépendante de trois événements. Or, le raffinement d'un système se fait en général par le raffinement effectif d'un nombre limité d'événements, les autres n'étant pas changés. Pour cette raison, d'un point de vue méthodologique, nous préférons décomposer la preuve de [13] en plusieurs étapes et de faire en sorte que les propriétés de vivacité de base ne dépendent que d'un seul événement. De plus, cela illustre comment la logique UNITY permet de raisonner sur les propriétés de vivacité.

Le variant simple que nous utilisons est la position du processus q dans la séquence : $V(q) = sq^{-1}(q)$. Nous posons les prédicats avant (P) et après (Q) de la règle d'induction. Le prédicat P indique que q est en attente, qu'il n'a pas le jeton et qu'il est dans la queue du serveur à la position n ($n > 0$) :

$$P \equiv q \in Wtg \wedge q \notin Tk \wedge sq^{-1}(q) = n$$

Le prédicat Q indique que q a avancé dans la queue ou qu'il a le jeton.

$$Q \equiv (q \in Wtg \wedge q \notin Tk \wedge sq^{-1}(q) < n) \vee (q \in Wtg \wedge q \in Tk)$$

Chaque événement acq , rel et srv a un effet atomique qui apporte un élément de la preuve de la propriété [13], sous l'hypothèse P . Ainsi, s'il n'y a pas de processus dans la section critique et qu'un processus (qui n'est pas q) a le jeton, alors l'événement acq permet de faire avancer le processus dans la file d'attente, ce qui établit Q ; c'est la propriété [14] associée à acq . Si un processus est dans la section critique, alors l'événement rel permet de l'enlever, ce qui contribue ainsi à rendre la garde de srv vraie; c'est la propriété [15]. Enfin, si aucun processus n'a le jeton, alors par srv , un des processus l'obtient, qui peut être le processus q , et sinon, cela ensuite permet de faire avancer les processus dans la file; c'est la propriété [16]. Formellement, ces propriétés de base s'écrivent :

$$acq \cdot P \wedge Act = \emptyset \wedge Tk \neq \emptyset \gg_w Q \quad [14]$$

$$rel \cdot P \wedge Act \neq \emptyset \gg_w P \wedge Act = \emptyset \quad [15]$$

$$srv \cdot P \wedge Tk = \emptyset \gg_w (P \wedge Act = \emptyset \wedge Tk \neq \emptyset) \vee (q \in Wtg \wedge q \in Tk) \quad [16]$$

Il reste maintenant à démontrer la propriété [13] à l'aide des règles de la logique UNITY. Alors que l'établissement des propriétés de base peut s'appuyer sur une compréhension opérationnelle du fonctionnement du système, la preuve UNITY s'obtient par une combinaison déductive des règles. Ainsi la validation du comportement est réalisée par un raisonnement déductif, qui est bien dans l'esprit de la méthode B. La preuve détaillée de [13] est :

1. $P \wedge Act = \emptyset \wedge Tk \neq \emptyset \rightsquigarrow Q$; BRL sur [14].
2. $P \wedge Act \neq \emptyset \rightsquigarrow P \wedge Act = \emptyset$; BRL sur [15].
3. $P \wedge Tk = \emptyset \rightsquigarrow (P \wedge Act = \emptyset \wedge Tk \neq \emptyset) \vee (q \in Wtg \wedge q \in Tk)$; BRL sur [16].
4. $P \wedge Tk = \emptyset \rightsquigarrow Q$; CAN sur 3, 1.
5. $Tk = \emptyset \equiv Act = \emptyset \wedge Tk = \emptyset$; de l'invariant.
6. $P \wedge Act = \emptyset \wedge Tk = \emptyset \rightsquigarrow Q$; 5 et 4.
7. $P \wedge Act = \emptyset \rightsquigarrow Q$; DSJ 6 et 1.
8. $P \wedge Act \neq \emptyset \rightsquigarrow Q$; TRA 2 et 7.
9. $P \rightsquigarrow Q$; DSJ 7 et 8.

Les propriétés de base [14], [15] et [16] sont prouvées par l'application des obligations de preuve WF0 et WF1. Nous prouvons ainsi que la propriété abstraite [10] est préservée dans $Sys1$. La preuve de préservation dépend de trois nouvelles propriétés de base qui doivent elles aussi être préservées dans le raffinement de $Sys1$.

5.3. Introduction d'un canal de communication

Dans le système $Sys1$ un nouvel événement est apparu modélisant un serveur. Nous voulons mettre en œuvre ce serveur dans un environnement distribué. Pour cela dans $Sys2$, le raffinement de $Sys1$, nous introduisons un autre événement appelé com qui modélise un système de communication et un canal de communication, modélisé

par une séquence injective de processus appelée ch . Lorsqu'un processus demande l'accès à la section critique, il envoie son identificateur au serveur. Cet envoi est modélisé par l'insertion de l'identificateur du processus dans la séquence ch . Lorsque le canal de communication n'est pas vide, l'événement com peut être activé. L'activation de cet événement provoque le transfert du message du canal de communication ch à la file d'attente locale du serveur modélisée par la séquence injective de processus ls . Le raffinement du système ne change pas les autres événements. La queue abstraite sq dans $Sys1$ et son raffinement ls dans $Sys2$ sont mises en relation par l'invariant de collage : $sq = ls \hat{\ } ch$. Les événements de $Sys2$ sont décrits à la figure 4.

```

EVENTS
   $rqf \hat{=} ANY p \text{ WHERE } p \in PR \wedge p \in Idl$ 
    THEN
       $st(p), ch := WT, ch \leftarrow p$ 
    END ;
   $acq \hat{=} ANY p \text{ WHERE } p \in PR \wedge p \in Wtg \wedge p \in Tk$ 
    THEN
       $st(p), ls := AC, tail(ls)$ 
    END ;
   $rel \hat{=} ANY p \text{ WHERE } p \in PR \wedge p \in Act$ 
    THEN
       $st(p), tk(p) := ID, false$ 
    END ;
   $srv \hat{=} SELECT ls \neq [] \wedge Tk = \emptyset$ 
    THEN
       $tk(first(ls)) := true$ 
    END ;
   $com \hat{=} SELECT ch \neq []$ 
    THEN
       $ls, ch := ls \leftarrow first(ch), tail(ch)$ 
    END
END

```

Figure 4. Événements du second raffinement

Dans ce raffinement, l'objectif est de prouver la préservation des trois propriétés de base [14], [15] et [16]. Pour cela nous devons prouver les obligations de preuve LIP et SAP pour chaque propriété. Nous ne présentons que les obligations de preuve correspondant à la vivacité :

$$P \wedge Act = \emptyset \wedge Tk \neq \emptyset \wedge \neg grd(acq) \rightsquigarrow grd(acq) \quad [17]$$

$$P \wedge Act \neq \emptyset \wedge \neg grd(rel) \rightsquigarrow grd(rel) \quad [18]$$

$$P \wedge Tk = \emptyset \wedge \neg grd(srv) \rightsquigarrow grd(srv) \quad [19]$$

En accord avec la remarque du paragraphe 4.2, les propriétés de vivacité [17] et [18] sont trivialement vraies car les implications suivantes sont vraies : $P \wedge Act = \emptyset \wedge Tk \neq \emptyset \Rightarrow grd(acq)$ et $P \wedge Act \neq \emptyset \Rightarrow grd(rel)$.

La propriété de vivacité [19] est déduite par l'application de la règle BRL sur la propriété :

$$com \cdot P \wedge Tk = \emptyset \wedge ls = [] \gg_w Tk = \emptyset \wedge ls \neq [] \quad [20]$$

qui est prouvée par l'application des obligations de preuve WF0 et WF1.

La preuve des propriétés [17], [18] et [19] garantit la préservation de la propriété abstraite $q \in Wtg \rightsquigarrow q \in Act$ dans *Sys2*. Si nous continuons le raffinement de *Sys2* nous devons garantir la préservation des propriétés [14], [15] héritées de *Sys1* et de la propriété [20].

6. Comparaison avec d'autres travaux

Nous pouvons comparer nos résultats avec les travaux qui fournissent des obligations de preuves et démontrent les propriétés de vivacité de manière déductive, en présence ou non d'hypothèses d'équité.

Dans [RUI 02], en ce qui concerne la preuve avec hypothèse d'équité faible, nous avons donné les obligations de preuve de la propriété de base notée " P ENSURES Q " dans le cas d'événements déterministes, sans distinguer l'événement utile G qui établit le prédicat Q . La première condition WF0 sur le choix E des événements du système formulait les deux possibilités :

$$\text{WF0} \quad I \wedge P \wedge \neg Q \Rightarrow [E] (P \vee Q)$$

La deuxième condition indiquait qu'il existait au moins un événement dont le comportement établissait Q . C'est-à-dire :

$$\text{WF1} \quad I \wedge P \wedge \neg Q \Rightarrow \neg [E] \neg Q$$

La nouvelle obligation de preuve WF1 de la propriété " $G \cdot P \gg_w Q$ " donnée au paragraphe 3.1.1 est plus explicite et plus générale. Elle doit préciser l'événement utile G . De plus, elle est applicable à tout type d'événements (déterministes ou non déterministes). D'autre part, nous ne donnons pas les conditions de raffinement des propriétés de base, ni de partage d'événements ANY.

Dans [MAT 01], les modalités proposées en [ABR 98] sont reformulées. Il s'agit de propriétés dynamiques sans hypothèse d'équité. Il y a une modalité de futur immédiat, dont la forme syntaxique générale et l'obligation de preuve sont, avec $I(x)$ l'invariant sur x :

```

ANY  $t$  WHERE
   $C(t, x)$ 
THEN
   $E1$  OR ... OR  $En$            $I(x) \wedge C(t, x) \wedge x = x_0 \Rightarrow [Ei] Q(t, x_0, x)$ 
ESTABLISH
   $Q(t, x_0, x)$ 
END

```

Si l'on compare avec nos obligations de preuve de propriété atomique avec hypothèse d'équité faible, on a les différences suivantes : nos hypothèses contiennent en plus la condition que Q n'est pas vrai ; la première condition WFO demande que les événements qui n'établissent pas Q préservent P (hypothèse d'équité faible) ; il doit y avoir au moins un événement qui établit Q dont la garde est vraie (ce qui n'est pas requis dans la modalité ESTABLISH) et enfin les événements G établissent Q (cette dernière condition est semblable à celle de J.-R. Abrial, en identifiant $E1$ OR ... OR En à G).

Dans ce même rapport [MAT 01], il est défini une modalité pour le futur imprécis dont la forme syntaxique et les obligations de preuves sont les suivantes, avec $T(t)$ le prédicat de typage des variables t introduites, $P(t, x)$ le prédicat qui caractérise l'état de départ qui est maintenu vrai durant l'évolution du système en parcourant les événements $E1, \dots, En$, $Q(t, x)$ le prédicat qui est fatalement atteint et $V(t, x)$ un variant sur l'ensemble bien fondé des entiers naturels :

```

ANY  $t$  WHERE
   $T(t)$ 
THEN
   $E1$  OR ... OR  $En$           Obligations de preuves
                               avec  $K$  mis pour  $I(x) \wedge T(t) \wedge P(t, x) \wedge \neg Q(t, x)$  :
MAINTAIN                       $K \Rightarrow \text{grd}(E1) \vee \dots \vee \text{grd}(En)$ 
   $P(t, x)$                      $K \Rightarrow [Ei] (\neg Q(t, x) \Rightarrow P(t, x))$ 
UNTIL                           $K \Rightarrow V(t, x) \in \mathbb{N}$ 
   $Q(t, x)$ 
VARIANT                         $K \wedge V(t, x) = n \Rightarrow [Ei] (\neg Q(t, x) \Rightarrow V(t, x) < n)$ 
   $V(t, x)$ 
END

```

Les propriétés $P \rightsquigarrow Q$ que nous introduisons dans cet article sont, dans leur esprit, semblable à cette modalité UNTIL. Comme dans UNITY, nous ne proposons pas d'obligations de preuve puisque cette modalité peut être prouvée par différentes suites de déduction. Cependant, on peut choisir de faire la déduction suivante en passant par la règle IND qui utilise le variant V et la règle de base BRL. Nous appellerons cette déduction BIR (pour *basic induction rule*) :

$$\frac{\frac{G \cdot P \wedge V = n \gg_w (P \wedge V < n) \vee Q}{P \wedge V = n \rightsquigarrow (P \wedge V < n) \vee Q}}{P \rightsquigarrow Q}$$

Après simplification, les obligations de preuve de la première formule pour prouver $P \rightsquigarrow Q$, avec le variant V , le choix d'événements utiles G et le choix de tous les événements du système S , sont :

$$\text{WF0} : I \wedge P \wedge V = n \wedge \neg Q \Rightarrow [S]((P \wedge V \leq n) \vee Q)$$

$$\text{WF1} : I \wedge P \wedge V = n \wedge \neg Q \Rightarrow \text{grd}(G) \wedge [G]((P \wedge V < n) \vee Q)$$

Si l'on identifie $T(t) \wedge P(t, x)$ avec le prédicat P et les événements $E1 \text{ OR } \dots \text{ OR } En$ au choix d'événements G , WF1 est équivalente aux obligations de preuve de la modalité UNTIL. La condition sur V , expression sur un ensemble bien fondé, n'est pas explicite dans la logique de UNITY, mais elle est également présente dans les deux cas. La seule différence vient de WF0. Elle dit que le système préserve $P \wedge V = n$ ou établit $(P \wedge V < n) \vee Q$. WF0, WF1 et l'hypothèse d'équité faible montrent que G doit être activé infiniment souvent, ce qui implique que fatalement Q est atteint au bout d'un "temps" fini.

En ce qui concerne le raffinement, le rapport et l'article [ABR 98] donnent deux conditions qui assurent que les propriétés de vivacité sont bien préservées. Ces conditions sont les conditions 4 et 5 de la section 4.1 que nous reformulons ici :

4) les nouveaux événements ne peuvent pas prendre le contrôle indéfiniment (cela est prouvable à l'aide d'un variant associé au système raffiné),

5) dans une modalité UNTIL, le raffinement des événements $E1 \text{ OR } \dots \text{ OR } En$ (notés E), qui assurent la progression, doit préserver la vivacité. Cela signifie qu'il existe toujours une garde vraie dans les événements raffinés $E1' \text{ OR } \dots \text{ OR } En'$ (notés E') ou dans les nouveaux événements H , c'est-à-dire : $I \wedge J \wedge \text{grd}(E) \Rightarrow \text{grd}(E' \parallel H)$.

Dans notre étude, nous n'imposons pas la condition 4 sur les nouveaux événements, dans le cas de raffinement de systèmes abstraits. Nous ne nous intéressons pas à la préservation de n'importe quelle propriété de vivacité, mais simplement à des propriétés avec un événement utile G , raffinés par G' . La préservation impose que l'on atteigne fatalement la garde de G' , dans l'image de l'état abstrait $I \wedge P$, c'est-à-dire : $I \wedge J \wedge P \wedge \neg \text{grd}(G') \rightsquigarrow \text{grd}(G')$. D'autre part, il faut assurer que cette garde est préservée par les nouveaux événements et les autres événements raffinés qui pourraient se déclencher. On peut montrer que la première condition est impliquée par les conditions de [ABR 98]. Notre deuxième condition associée à l'hypothèse d'équité indique que l'événement G' est fatalement activé. Dans la théorie de Abrial-Mussat qui n'a pas d'hypothèses d'équité, cette activation est due aux conditions sur les nouveaux événements, qui ne modifient pas l'état abstrait et dont toute séquence d'exécution doit terminer sans introduire de blocages (condition 5) au cours de la progression vers le but de la propriété de vivacité.

Le cadre de Back et Xu [BAC 98] est celui des systèmes d'actions (événements), qui sont des systèmes de commandes gardées. La notion de *progression* vers un but est assurée par une famille de prédicats (le variant) qui traduisent la condition de décroissance d'une variable dans un ensemble bien fondé. Une action est *utile* dans un

certain état r si elle fait décroître le variant. Une action est *utile ou stable*⁶ dans r si elle fait décroître le variant ou maintient r et la valeur du variant. Un sous-ensemble d'actions $A_i \in WF$ est déclaré faiblement équitable dans des états r_i .

Le théorème de terminaison avec équité faible établit que, partant d'un état p , le système termine si les conditions suivantes sont satisfaites :

- 1) p implique qu'il existe une suite bien fondée de prédicats de décroissance,
- 2) dans r_i , l'action A_i est utile et habilitée,
- 3) dans ce même sous-état r_i , les actions de $A_{j, j \neq i}$ de l'ensemble WF sont utiles ou stables.
- 4) lorsque tous les r_i sont faux (c'est-à-dire, il n'y a pas de condition d'équité), tous les événements sont utiles,

Dans notre approche, tous les événements du système sont faiblement équitables dans l'état caractérisé par $P \wedge \neg Q$, donc le cas 4 n'a pas de correspondant.

Le théorème de correction totale établit les conditions pour atteindre q à partir d'un état p . En plus des conditions de terminaison, les auteurs ajoutent la condition explicite que chaque événement termine lorsque la garde est vraie, ce qui est aussi notre cas, et que lorsque le système se bloque (les gardes sont fausses), alors le prédicat q est vrai. On remarque que l'on a les mêmes conditions que celles obtenues pour BIR, en prenant pour r le prédicat $P \wedge \neg Q$, qui est celui dans lequel les conditions WF0 et WF1 s'appliquent.

Dans les conditions de raffinement d'un système avec hypothèse d'équité faible, [BAC 98] donne la condition de non-divergence sur les nouveaux événements (comme dans le raffinement de systèmes de Abrial-Mussat [ABR 98]). Il ajoute une condition de "préservation d'équité" qui dit que si une action A de l'ensemble WF est exécutée sous l'hypothèse d'équité, alors le système concret finit par exécuter le raffinement de A . Cette condition, bien qu'exprimée très différemment, correspond à nos conditions de préservation par raffinement des propriétés de vivacité atomiques.

7. Conclusion

Nous avons présenté une approche à la spécification, preuve et raffinement des propriétés d'équité sous hypothèses d'équité faible dans le B événementiel. L'approche est inspirée en partie du formalisme UNITY. Nous donnons des obligations de preuve fondées sur le calcul des plus faibles préconditions pour prouver les propriétés de vivacité de base, et nous proposons une technique pour appliquer ces obligations de preuve dans le cadre d'événements non déterministes (ANY), qui sont spécifiques du langage B. Les propriétés de vivacité générales sont prouvées par les définitions et théorèmes de la logique UNITY. Nous proposons d'accompagner le raffinement d'un système par la preuve de préservation des propriétés de vivacité et nous donnons les obligations

6. En anglais, *hepful or stay*.

de preuve nécessaires à cet effet. Notre démarche est illustrée par un exemple d'un système de gestion de processus que nous développons à travers deux raffinements.

Les conditions de raffinement génèrent des obligations de preuves liées aux propriétés de vivacité à préserver. En réalité, ce fait est dépendant de la méthode choisie, puisque ce sont précisément les propriétés de vivacité et leur préservation qui guident le raffinement. En [ABR 98], la condition d'existence d'un variant et de la décroissance de ce variant pour les nouveaux événements d'un système raffiné a été énoncée de façon indépendante des propriétés de vivacité à préserver. Cependant, l'autre condition de raffinement est la préservation du fait que le système abstrait ne se bloque pas pendant sa progression vers Q dans un P until Q sous les hypothèses $P \wedge \neg Q$ (paragraphe 6, condition 5), ce qui est clairement dépendant de la propriété à préserver. Enfin, nous relaxons la condition de décroissance du variant pour tous les nouveaux événements, qui est une technique quelquefois difficile à mettre en œuvre, et nous la remplaçons par un jeu de possibilités qui vont de la règle IND (introduction d'un variant) à l'utilisation des théorèmes appropriés de la logique UNITY.

Dans un rapport [RUI 05], nous donnons les justifications théoriques des obligations de preuve que nous avons présentées. Ces preuves sont développées dans un cadre ensembliste et utilisent un opérateur de choix équitable.

A partir du travail présenté dans cet article, une perspective consiste à implémenter le générateur d'obligations de preuve pour nos propriétés de vivacité de base. La complexité de ces obligations de preuve est grande et le support d'outils pour leur génération est indispensable. La décision d'introduire une référence à l'événement utile dans les propriétés de vivacité permet de guider facilement cet outil.

Nous expérimentons notre démarche sur des exemples plus importants. Il s'avère que le raffinement d'un système abstrait doit être guidé par le raffinement et la preuve des propriétés de vivacité en cours sur le système. C'est un aspect méthodologique qu'il faudrait approfondir. Dans la même ligne, nous étudions d'autres obligations de preuve permettant de continuer le raffinement d'un système *avec* hypothèses d'équité faible dans un système *sans* hypothèses d'équité (avec progrès minimal). Ce cas de raffinement est très intéressant lorsque le comportement d'un système est garanti par les événements spécifiés dans un certain niveau de raffinement sans qu'il y ait besoin d'introduire des hypothèses d'équité supplémentaires. Le raffinement du système introduit naturellement l'ordonnanceur concret des événements qu'il n'était pas nécessaire d'introduire à un niveau plus abstrait.

8. Bibliographie

- [ABR 96] ABRIAL J.-R., *The B Book - Assigning Programs to Meanings*, Cambridge University Press, August 1996.
- [ABR 98] ABRIAL J.-R., MUSSAT L., « Introducing Dynamic Constraints in B », BERT D., Ed., *B'98 : Recent Advances in the Development and Use of the B Method*, LNCS 1393, Springer-Verlag, 1998, p. 83–128.

- [BAC 98] BACK R. J. R., XU Q., « Refinement of fair action systems », *Acta Informatica*, vol. 35, 1998, p. 131–165.
- [BRO 94] BROY M., NELSON G., « Adding Fair Choice to Dijkstra’s Calculus », *ACM Transactions on Programming Languages and Systems*, vol. 16, n° 3, 1994, p. 924–938.
- [CHA 88] CHANDY K. M., MISRA J., *Parallel Program design : A Foundation*, Addison-Wesley, 1988.
- [CHO 03] CHOUALI S., « Contribution du raffinement à la vérification de systèmes sous hypothèses d’équité », Thèse de l’Université de Franche-Comté, Besançon, Décembre 2003.
- [FRA 86] FRANCEZ N., *Fairness*, Springer-Verlag, 1986.
- [LAM 94] LAMPORT L., « The Temporal Logic of Actions », *ACM Trans. Program. Lang. Syst.*, vol. 16, n° 3, 1994, p. 872–923.
- [MAT 01] MATISSE, « Event B Reference Manual », MATISSE : Methodologies and Technologies for Industrial Strength Systems Engineering, http://www.atelierb.societe.com/index_uk.html, 2001.
- [MIS 95a] MISRA J., « A logic for concurrent programming : Progress », *Journal of Computer and Software Engineering*, vol. 3, n° 2, 1995, p. 273–300.
- [MIS 95b] MISRA J., « A logic for concurrent programming : Safety », *Journal of Computer and Software Engineering*, vol. 3, n° 2, 1995, p. 239–272.
- [PRA 94] PRASETYA I. S. W. B., « Error in the UNITY Substitution Rule for Subscripted Operators », *Formal Aspects of Computing*, vol. 6, 1994, p. 466–470.
- [RUI 02] RUIZ-BARRADAS H., BERT D., « Specification and Proof of Liveness Properties under Fairness Assumptions in B Event Systems », *Integrated Formal Methods, IFM2002, LNCS 2335*, Springer-Verlag, May 2002, p. 360–379.
- [RUI 04] RUIZ-BARRADAS H., BERT D., « Propriétés dynamiques avec hypothèses d’équité en B événementiel », *Approches Formelles dans l’Assistance au Développement de Logiciels, AFADL’2004*, Besançon, France, June 2004, p. 299–313.
- [RUI 05] RUIZ-BARRADAS H., BERT D., « Proof Obligations for Specification and Refinement of Liveness Properties under Weak Fairness », rapport n° 1071-I LSR 20, 2005, LSR-IMAG, Grenoble.
- [SAN 91a] SANDERS B. A., « Eliminating the Substitution Axiom from UNITY Logic », *Acta Informatica*, vol. 3, 1991, p. 189–205.
- [SAN 91b] SANDERS B. A., « On the UNITY design decisions », *Research Directions in High-Level Parallel Programming Languages*, LNCS 574, Springer-Verlag, 1991, p. 50–63.

Article reçu le ...
Version révisée le 8 juillet 2005.
Rédacteur responsable : ...

***Héctor Ruíz Barradas** est enseignant-chercheur à l'Universidad Autónoma Metropolitana à México. Il s'intéresse au développement de systèmes à partir des méthodes formelles. Il est doctorant à l'Université Joseph Fourier de Grenoble sous la direction de Didier Bert. Il mène ses activités de recherche au laboratoire Logiciels, Systèmes, Réseaux à l'IMAG. Actuellement ses recherches portent sur la sémantique de systèmes d'événements avec équité.*

***Didier Bert** est chargé de recherche au CNRS. Membre du laboratoire Logiciels, Systèmes, Réseaux à l'IMAG, ses activités de recherche portent sur les méthodes de spécification algébrique et le développement rigoureux de logiciels avec la méthode B. Il anime des activités de la communauté B : groupe de travail au GDR ALP du CNRS, conférence B, site web. Il est également membre du groupe WG1.3 (Foundations of Software Specification) de l'IFIP.*