Explicitation du contrôle de développements B *événementiel*

Nicolas Stouls (Nicolas.Stouls@imag.fr)*
Marie-Laure Potet (Marie-Laure.Potet@imag.fr)
LSR-IMAG - 681, rue de la Passerelle, BP72, 38402 St Martin d'Hères Cedex

Le 27 juillet 2006

Résumé

Initialement la méthode B a été conçue et utilisée comme une méthode de développement de logiciels dans lesquels on s'intéresse aux données et aux traitements. L'introduction des événements, et la notion associée de raffinement, permet la prise en compte des aspects comportementaux des systèmes. Le raffinement devient alors un outil très puissant qui permet de décrire les systèmes suivant différents degrés de granularité. Les modèles sont alors plus complets et les processus de spécification et de raffinement deviennent, quant à eux, plus complexes. Il devient nécessaire d'offrir des outils de plus haut niveau permettant d'assister à la fois la spécification et la preuve. Nous présentons l'outil GénéSyst qui permet de visualiser le comportement de modèles B sous forme de systèmes de transitions symboliques. Cette approche, proposée initialement dans [?], est étendue ici pour prendre en compte le raffinement. Les systèmes de transitions construits sont alors hiérarchisés.

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

1 Introduction

L'utilisation de méthodes formelles telles que la méthode B permet d'obtenir un très bon niveau de confiance dans le développement, mais ne donne aucune garantie de correction de la spécification par rapport au problème posé. C'est pourquoi il est souvent nécessaire de proposer des visualisations graphiques de la spécification. Le but du travail présenté ici est de décrire une méthode d'extraction du contrôle d'un système B événementiel qui permette de produire un système de transitions étiquetées associé à la spécification donnée. Cette méthode est étendue pour prendre en compte le raffinement.

Dans la section 2 nous présentons rapidement l'approche B événementiel, en explicitant une sémantique de traces possible. Dans la section 3 nous décrivons le principe de construction d'un système de transitions symbolique en prouvant la préservation des aspects observationnels. Cette construction est étendue à la démarche de construction par

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

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

raffinement dans la section 4. Nous proposons une première approche qui permet de préserver la structure du système de transitions en le projetant par raffinement. Nous étudions alors la possibilité de hiérarchiser le système de transitions obtenu, afin de permettre la visualisation à différents niveaux d'abstraction.

2 Notions de B événementiel

Le B événementiel, introduit par J.-R. Abrial [?, ?], est une méthode de développement formel ainsi qu'un langage de spécification. Un développement B événementiel commence par la rédaction d'une spécification, dont on prouve la cohérence. On construit ensuite des raffinements, pour lesquels on établit la cohérence interne et la correction vis-à-vis de la spécification.

2.1 Composants B

```
MACHINE        parking
CONSTANTS      NbMax
PROPERTIES     NbMax > 0
VARIABLES      NbVoit
INVARIANT      NbVoit ∈ 0..NbMax
INITIALISATION NbVoit := 0
OPERATIONS
 entrer ≙ SELECT NbVoit < NbMax
          THEN   NbVoit := NbVoit + 1 END ;
 sortir ≙ SELECT NbVoit > 0
          THEN   NbVoit := NbVoit − 1 END
END
```

FIG. 1 – Exemple d'un système B événementiel.

Un composant B (de machine ou de raffinement) contient des clauses qui permettent de décrire les données (CONSTANTS, VARIABLES ...), leurs propriétés (PROPERTIES, INVARIANT ...) et la dynamique du système (INITIALISATION, OPERATIONS ...).

Dans la méthode B, les propriétés sont décrites en logique du premier ordre. Les événements et l'initialisation sont, quant à eux, exprimés sous forme de substitutions généralisées. Les événements sont définis, dans la clause OPERATIONS, sous la forme : $NomEvenement \mathrel{\hat{=}} CorpsEvenement$.

La figure 1 décrit un exemple de spécification B événementiel modélisant un parking. La clause MACHINE désigne un composant de spécification. Dans la suite, nous parlerons de système. Si S est un composant, nous désignons par $Interface(S)$ l'ensemble de ses événements.

2.2 Les substitutions généralisées

Les substitutions généralisées primitives considérées ici sont l'assignation (x:=E), la garde (SELECT P THEN T END), le choix borné (CHOICE T_1 OR T_2 END), le choix non

borné (VAR z IN T END) et la séquence $(T_1 \ ; \ T_2)$[1]. Mathématiquement, nous notons la garde : $P \Longrightarrow T$; le choix borné : $T_1 \ [\!] \ T_2$; et le choix non borné : $@z \cdot T$.

Ces substitutions nous permette d'en construire d'autres telles que la conditionnelle : IF P THEN T_1 ELSE T_2 END. Celle-ci se définit par $P \Longrightarrow T_1 \ [\!] \ \neg P \Longrightarrow T_2$.

2.3 Calcul sur les substitutions

En B événementiel, le calcul de la plus faible pré-condition est noté $[T]R$, où T est une substitution et R une post-condition. On dit que la plus faible pré-condition $P = [T]R$ est telle que si P est vérifiée, alors R est vérifiée après exécution de T.

Définition 1 Calcul du $\mathcal{WP}$ sur les substitutions primitives

Substitution	*Réduction*	*Condition*
$[x, y := E, F]R$	$[z := F][x := E][y := z]R$	$z \backslash E, F, R$
$[T_1 \ [\!] \ T_2]R$	$[T_1]R \ \wedge \ [T_2]R$	
$[P \Longrightarrow T]R$	$P \Rightarrow [T]R$	
$[@ \ z \cdot T]R$	$\forall z \cdot [T]R$	$z \backslash R$
$[T_1 \ ; \ T_2]R$	$[T_1][T_2]R$	

La notation $z \backslash R$ signifie que la variable z n'est pas libre dans R.

Notons $\langle T \rangle R$ le conjugué $\neg[T]\neg R$ de $[T]R$. Intuitivement, si la formule $\langle T \rangle R$ est vérifiée, alors il existe une exécution de T qui vérifie R.

Définition 2 Cohérence d'une machine.
Soit S une machine et I son invariant. S est dite cohérente si l'initialisation $Init$ et chacun des événements Ev vérifient :

$$[Init]I \qquad et \qquad I \Rightarrow [Ev]I$$

Voici deux prédicats définis sur les substitutions généralisées : le prédicat avant-après, noté $\mathsf{prd}_x(T)$, et la faisabilité, notée $\mathsf{fis}(T)$. Si x désigne les variables du système, alors $\mathsf{prd}_x(T)$ caractérise l'ensemble des couples (x, x') tels que x' désigne les valeurs de l'état après exécution de S et x désigne celles avant exécution de T. Le prédicat $\mathsf{fis}(T)$ décrit le domaine de $\mathsf{prd}_x(T)$.

Définition 3 Définition formelle de $\mathsf{prd}_x(T)$ et de $\mathsf{fis}(T)$

Symbole	*Définition*	*Définition mathématique*
$\mathsf{prd}_x(T)$	*Le prédicat avant-après de T pour x*	$\langle T \rangle (x' = x)$
$\mathsf{fis}(T)$	*La faisabilité de la substitution T*	$\exists x' \cdot \mathsf{prd}_x(T)$

[1] Nous ne considérons pas ici la substitution pré-conditionnée, car elle n'est pas utilisée dans le cadre du B événementiel. En effet, en B événementiel, un événement se déclenche spontanément lorsque sa garde est vérifiée. Il s'ensuit que toute substitution termine. Nous ne parlerons donc pas de terminaison.

2.4 Evénements et traces

Dans la suite nous considérons que les événements sont sous la forme suivante :

Définition 4 Forme d'un événement

$$Ev \mathrel{\hat{=}} G \Longrightarrow T \qquad avec \qquad G \Rightarrow \mathsf{fis}(T)$$

Le prédicat G est appelé la *garde* de Ev et T son *action*. On les note respectivement $Garde(Ev)$ et $Action(Ev)$. Cette forme symbolique peut être obtenue en montrant que les parties gauches et droites des égalités suivantes ont des prd équivalents. En particulier, nous avons $\mathsf{prd}_x(Ev) \Leftrightarrow Garde(Ev) \ \wedge \ \mathsf{prd}_x(Action(Ev))$.

$x := E$	=	$\mathsf{true} \Longrightarrow x := E$
$G_1 \Longrightarrow (G_2 \Longrightarrow T)$	=	$G_1 \ \wedge \ G_2 \Longrightarrow \ T$
$G_1 \Longrightarrow T_1 \parallel G_2 \Longrightarrow T_2$	=	$(G_1 \ \vee \ G_2) \Longrightarrow (G_1 \Longrightarrow T_1 \parallel G_2 \Longrightarrow T_2)$
$@ \ z \cdot G \Longrightarrow T$	=	$\exists z \cdot G \Longrightarrow @ \ z \cdot (G \Longrightarrow T)$
$G_1 \Longrightarrow T_1 \ ; \ G_2 \Longrightarrow T_2$	=	$(G_1 \ \wedge \ \langle T_1 \rangle G_2) \Longrightarrow (T_1 \ ; \ G_2 \Longrightarrow T_2)$

Enfin, pour pouvoir représenter et raffiner des systèmes B événementiel, la sémantique de ces systèmes doit être précisée. En accord avec la notion de raffinement (section 2.5), nous choisissons ici une vue *event-based* qui permet d'observer un système sous la forme des événements déclenchables.

Définition 5 Trace d'un système B événementiel
Une suite finie d'occurences d'événements $Init; oc_1; oc_2; \ldots; oc_n$ *est une trace d'un système* S *si et seulement si* $Init$ *est l'initialisation de* S, $\{oc_1, \ldots, oc_n\} \subseteq Interface(S)$ *et* $\mathsf{fis}(Init; oc_1; oc_2; \ldots; oc_n) \Leftrightarrow \mathsf{true}$.

On désigne par $Traces(S)$ l'ensemble des traces du système S.

Lemme 1 Caractérisation d'une trace

$$oc_0; \ldots; oc_n \in Traces(S) \Leftrightarrow$$
$$\exists x_0, \ldots, x_{n+1} \cdot (\bigwedge_{i=0}^{n} ([x := x_i] Garde(oc_i) \wedge [x, x' := x_i, x_{i+1}] \mathsf{prd}_x(Action(oc_i))))$$

avec x *l'espace de variables du système* A.

Ce lemme sera utile pour la démonstration du théorème section 3.3. Enfin, rappelons que $\mathsf{prd}_x(Init)$ ne dépend pas de x et que $Garde(Init) \Leftrightarrow \mathsf{true}$.

2.5 Raffinement en B événementiel

Dans la méthode B, un raffinement se présente sous la forme d'un composant appelé *refinement*. Les variables peuvent être raffinées et l'invariant de liaison décrit la relation entre les variables de l'abstraction et celles du raffinement. Les événements du raffinement R doivent contenir au moins tous ceux de l'abstraction S (i.e. $Interface(S) \subseteq Interface(R)$).

La figure 2 décrit un raffinement du parking qui introduit un contrôleur. La variable *cc*

permet d'ordonnancer les événements de l'environnement (*entrer* / *sortir*) et du contrôleur (*controler_entree*/*controler_sortie*). Ce raffinement sera étendu en figure 8, section 4.2.2.

REFINEMENT $parking\_r1$
REFINES $parking$
VARIABLES $NbVoit, cc$
INVARIANT $(cc \in -1..1) \ \wedge \ (cc = -1 \ \Rightarrow \ NbVoit < NbMax) \ \wedge \ (cc = 1 \ \Rightarrow \ NbVoit > 0)$
INITIALISATION $NbVoit := 0 \ || \ cc := 0$
OPERATIONS

entrer	$\widehat{=}$ SELECT $NbVoit < NbMax \wedge (cc = 0)$	THEN	$NbVoit := NbVoit + 1 \ \| \ cc := 1$ END ;
controler_entree	$\widehat{=}$ SELECT $cc = 1$	THEN	$cc := 0$ END ;
sortir	$\widehat{=}$ SELECT $NbVoit > 0 \wedge (cc = 0)$	THEN	$NbVoit := NbVoit - 1 \ \| \ cc := -1$ END ;
controler_sortie	$\widehat{=}$ SELECT $cc = -1$	THEN	$cc := 0$ END

END

FIG. 2 – Exemple d'un parking avec contrôleur (inactif)

Nous donnons ci-après les obligations de preuves liées au raffinement. Si I est l'invariant d'une spécification S et J celui de son raffinement R, alors on appelle *invariant de liaison* la conjonction de I et J.

Définition 6 Raffinement de substitutions
Le raffinement d'une substitution T_S par une substitution T_R, pour l'invariant de liaison L, est noté $T_S \sqsubseteq_L T_R$ et est défini par :

$$L \Rightarrow [T_R]\langle T_S \rangle L$$

Dans la méthode B, le raffinement s'effectue par partie, c'est-à-dire que l'initialisation abstraite se raffine par l'initialisation concrète et que chaque événement est raffiné par sa définition concrète. Les obligations de preuve permettant d'établir le raffinement sont :

Pour l'initialisation :	$[Init_R]\langle Init_S \rangle J$
Pour les événements Ev_R de $Interface(S)$:	$I \ \wedge \ J \Rightarrow [Ev_R]\langle Ev_S \rangle J$
Pour les nouveaux événements NEv_R :	$I \ \wedge \ J \Rightarrow [NEv_R]\langle skip \rangle J$

Un nouvel événement ne doit pas prendre la main indéfiniment afin de garantir que le système raffiné ne diverge pas plus souvent que son abstraction. Pour cela, on utilise une expression V, appelée variant, définie sur $\mathbb{N}$, qui décroit strictement à chaque exécution d'un nouvel événement NEv_R ($I \ \wedge \ J \Rightarrow [v := V][NEv_R](V < v)$). De plus, on doit prouver $I \ \wedge \ J \Rightarrow V \geq 0$.

Les traces associées à un raffinement sont définies comme pour les spécifications.

Propriété 1
Soit R un raffinement et S son système abstrait associé, alors le lien entre les traces de S et de R peut être caractérisé de la manière suivante :

$$\forall t \cdot (t \in Traces(R) \Rightarrow \exists t' \cdot (t' \in Traces(S) \ \wedge \ abs(t) = t'))$$

Où $abs(t)$ est la trace t modulo les nouveaux événements, définie par :

$$abs(\epsilon) = \epsilon$$
$$abs(oc; t) = \textbf{si } oc \in Interface(S) \textbf{ alors } oc; abs(t) \textbf{ sinon } abs(t)$$

3 Construction d'un système de transitions à partir d'un système B événementiel

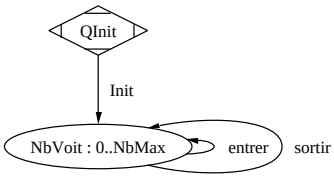


FIG. 3 – **STE** symbolique associé au parking

Classiquement un système met en jeu deux aspects orthogonaux : les données et le contrôle. La séparation entre ces deux aspects est principalement conceptuelle. La représentation par des données donne un cadre homogène pour raisonner, alors que la mise en évidence du contrôle peut permettre de mieux comprendre le fonctionnement d'un système. L'approche B événementiel est principalement basée sur les données. Ceci oblige, comme dans l'exemple du raffinement du parking, à introduire une variable d'ordonnancement. Par contre, la spécification est écrite sous une forme homogène puisqu'il n'y a pas besoin de différencier les invariants en fonction des états. L'objectif ici est donc, partant d'un système B événementiel, de proposer des visualisations de son comportement par des systèmes de transitions.

Les domaines de valeurs manipulés par les systèmes B sont potentiellement infinis. Nous choisissons donc des systèmes de transitions symboliques qui permettent de représenter un ensemble potentiellement infini de valeurs par un seul état. La figure 3 donne l'exemple d'un système de transitions symbolique représentant les comportements du parking.

3.1 Système de transitions symbolique

Définition 7 Système de transitions étiquetées
Un système de transitions étiquetées peut être vu comme un quadruplet (N, Q_{Init}, U, W) *tel que :*
- N *est un ensemble d'états,*
- Q_{Init} *est l'état initial* ($Q_{Init} \in N$),
- U *est un ensemble d'étiquettes,*
- W *est une relation de transition* ($W \subseteq N \times U \times N$).

Dans un système de transitions étiquetées symbolique, les états sont caractérisés par des prédicats et les transitions sont conditionnées. Nous avons choisi d'expliciter au mieux les conditions sur les transitions en utilisant la forme générale suivante pour les étiquettes :

$$(D, A, Ev)$$

Soit $(E, (D, A, Ev), F)$ une transition, alors le prédicat D exprime la condition de déclenchabilité de l'événement Ev depuis l'état E et le prédicat A exprime l'atteignabilité de F par Ev depuis E.

Définition 8 Franchissement d'une transition
On note :

$$(E_1, x_1) \rightsquigarrow^{(D,A,Ev)} (E_2, x_2)$$

le franchissement de la transition $(E_1, (D, A, Ev), E_2)$*, où* $x_1 \in E_1$ *et* $x_2 \in E_2$ *sont les valeurs de* x *avant et après le franchissement. Celui-ci est correct si et seulement si :*

1. $[x := x_1](E_1 \ \wedge \ D \ \wedge \ A)$
2. $[x, x' := x_1, x_2]\mathsf{prd}_x(Action(Ev))$
3. $[x := x_2]E_2$

Définition 9 Chemin d'un système de transitions étiquetées
Toute suite d'occurrences d'événement $oc_0; \ldots; oc_n$ *d'un système de transitions étiquetées* T *est un chemin si et seulement si on peut trouver une séquence* $E_0; \ldots; E_{n+1}$ *d'états de* N *et une séquence de transitions telles que :*

$$\exists x_0, \ldots, x_{n+1} \cdot (E_0 = Q_{Init} \wedge \bigwedge_{i=0}^{n}((E_i, x_i) \rightsquigarrow^{(D_i,A_i,Action(oc_i))} (E_{i+1}, x_{i+1})))$$

On note $Chemins(T)$ *l'ensemble des chemins de* T*.*

3.2 Construction des états

Nous allons décrire maintenant le principe de construction d'un système de transitions étiquetées à partir d'un système B événementiel. Nous avons choisi de laisser l'utilisateur fournir les états. Ceux-ci doivent garantir que l'on représente exactement les valeurs vérifiant l'invariant.

Condition 1 Complétude des états.
Soit $\{E_1, \ldots, E_n\}$ *un ensemble de prédicats caractérisant des états. Cet ensemble est dit complet vis-à-vis d'un invariant* I *si et seulement si :*

$$I \Leftrightarrow \bigvee_{i=1}^{n} E_i$$

Il suffit que les états fournis couvrent l'invariant. On construit pour la suite l'ensemble des états $N = \bigcup_{i=1}^{n}\{E_i \ \wedge \ I\} \cup \{\mathsf{true}\}$, avec $Q_{Init} = \mathsf{true}$.

3.3 Construction de la relation de transition

Tout d'abord, définissons la construction d'une transition :

Définition 10 Construction d'une transition
Soit E *et* F *deux états et* Ev *une substitution. Alors la transition* $(E, (D, A, Ev), F)$ *est valide si et seulement si les conditions* D *et* A *vérifient :*

a) $E \Rightarrow (D \Leftrightarrow Garde(Ev))$
b) $E \ \wedge \ Garde(Ev) \Rightarrow (A \Leftrightarrow \langle Action(Ev)\rangle F)$
c) $D \not\Leftrightarrow \mathsf{false} \ \wedge \ A \not\Leftrightarrow \mathsf{false}$

Remarque : Par application de la définition du $\mathcal{WP}$ conjugué, la condition *b)* est équivalente à :

$$E \ \wedge \ Garde(Ev) \Rightarrow (A \Leftrightarrow \exists x' \cdot (\mathsf{prd}_x(Action(Ev)) \ \wedge \ [x := x']F))$$

L'algorithme de calcul de la relation de transition W, est appliqué depuis Q_{Init} et inductivement sur les états atteignables :

- Calcul d'une transition valide, s'il en existe, pour chaque (E, Ev, F) avec : $E = Q_{Init}$, $Ev = Init$ et $F \in N - \{Q_{Init}\}$.
- Calcul d'une transition valide, s'il en existe, pour chaque (E, Ev, F) avec : $E \in N - \{Q_{Init}\}$, $Ev \in Interface(S)$ et $F \in N - \{Q_{Init}\}$.

La difficulté consiste à déterminer les conditions de déclenchabilité et d'atteignabilité. Nous nous intéressons ici aux trois cas possibles suivants : true, false et autre. Ceci permet de ramener la recherche des conditions à la preuve des formules de la figure 4.

	Obligations de preuve	Valeur des conditions
(1)	$\forall x \cdot (E \Rightarrow Garde(Ev))$	$D_{(E,Ev)} \Leftrightarrow \mathsf{true}$
(2)	$\forall x \cdot (E \Rightarrow \neg Garde(Ev))$	$D_{(E,Ev)} \Leftrightarrow \mathsf{false}$
(3)	$\neg((1) \ \vee \ (2))$	$D_{(E,Ev)} \Leftrightarrow Garde(Ev)$
(4)	$E \ \wedge \ Garde(Ev) \Rightarrow \langle Action(Ev)\rangle F$	$A_{(E,Ev,F)} \Leftrightarrow \mathsf{true}$
(5)	$E \ \wedge \ Garde(Ev) \Rightarrow [Action(Ev)]\neg F$	$A_{(E,Ev,F)} \Leftrightarrow \mathsf{false}$
(6)	$\neg((4) \ \vee \ (5))$	$A_{(E,Ev,F)} \Leftrightarrow \langle Action(Ev)\rangle F$

FIG. 4 – Valeur des conditions en fonction des obligations de preuve vérifiées

Théorème Egalité des traces
Si S est un système B *événementiel pour lequel l'invariant I a été établi et T un système de transitions étiquetées généré à partir de S, alors :*

$$Traces(S) = Chemins(T)$$

Démonstration Montrons que $t \in Chemins(T) \Leftrightarrow t \in Traces(S)$
avec $t \hat{=} oc_0; \ldots; oc_n$ un chemin de T si et seulement si (Def. 9) :

$$\exists x_0, \ldots, x_{n+1} \cdot (E_0 = Q_{Init} \ \wedge \ \textstyle\bigwedge_{i=0}^{n}(E_i, x_i) \rightsquigarrow^{(D_i, A_i, Action(oc_i))} (E_{i+1}, x_{i+1}))$$

En utilisant la définition 8, on se ramène à :

$$\exists x_0, \ldots, x_{n+1} \cdot (E_0 = Q_{Init} \ \wedge \ \textstyle\bigwedge_{i=0}^{n}([x := x_i](E_i \ \wedge \ D_i \ \wedge \ A_i) \\ \wedge \ [x, x' := x_i, x_{i+1}]\mathsf{prd}_x(Action(oc_i)) \ \wedge \ [x := x_{i+1}]E_{i+1}))$$

Par application de la définition 10, on peut remplacer D_i par $Garde(oc_i)$ et A_i par $\exists x' \cdot (\mathsf{prd}_x(Action(oc_i)) \ \wedge \ [x := x']E_{i+1})$. La formule ci-dessus se simplifie donc en :

$$(1) \quad \exists x_0, \ldots, x_{n+1} \cdot (E_0 = Q_{Init} \ \wedge \ \textstyle\bigwedge_{i=0}^{n}([x := x_i](E_i \ \wedge \ Garde(oc_i)) \\ \wedge \ [x, x' := x_i, x_{i+1}]\mathsf{prd}_x(Action(oc_i)) \ \wedge \ [x := x_{i+1}]E_{i+1}))$$

Nous allons montrer que cette formule est équivalente au lemme 1 (section 2.4) définissant les traces d'un système B *événementiel, pour lequel l'invariant I a été établi. C'est-à-dire :*

$$(2) \qquad \exists x_0, \ldots, x_{n+1} \cdot (\bigwedge_{i=0}^{n}([x := x_i]Garde(oc_i) \wedge [x, x' := x_i, x_{i+1}]\mathsf{prd}_x(Action(oc_i)) \ \wedge \ [x := x_{i+1}]I))$$

L'implication (1) $\Rightarrow$ (2) *est vérifiée car les états E_i sont tels que $E_i \Rightarrow I$ (condition 1 section 3.2).*

Pour montrer l'implication (2) $\Rightarrow$ (1)*, nous devons exhiber une liste d'états (Q_{Init}, E_1, ..., E_{n+1}) telle que ces états vérifient* (1)*. Ceci découle du fait que $Q_{Init} =$* true *et de la condition 1 qui garantit $I \Rightarrow \bigvee_{j=1}^{m} E_j$. Donc, si I est vrai alors un des états E_j est nécessairement vérifié.*

□

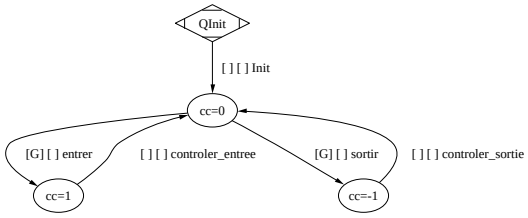


FIG. 5 – Exemple de système de transitions associé au raffinement figure 2.

En figure 5, nous donnons un exemple de système de transitions généré à partir du raffinement Parking_r1 (figure 2). Une condition y est notée [ ] si elle est réductible à true. Sinon, elle y est notée $[G]$. Les transitions *controler_entree* et *controler_sortie* ne sont pas conditionnées car *cc* est instanciée. En revanche, *entrer* et *sortir* dépendent de *NbVoit* et sont donc conditionnées.

4 Génération de système de transitions étiquetées pour les raffinements

Nous nous intéressons maintenant à une construction similaire pour les raffinements. Il est possible de considérer un raffinement comme une spécification à part entière [**?**, **?**] et donc d'appliquer la méthode précédente. Néanmoins, notre objectif est de mettre en évidence le lien introduit par le raffinement, en préservant la structure du système de transitions associé à la spécification. Dans le cas du B événementiel, le raffinement prend en compte deux aspects :

1. Le changement de représentation des données. Les événements doivent alors être redéfinis pour prendre en compte les nouvelles représentations des données et aussi pour préciser les calculs. En particulier le non-déterminisme peut être levé.
2. L'affinage de la dynamique du système. Le raffinement autorise le renforcement des gardes, ce qui peut restreindre la déclenchabilité de certains événements. De plus,

l'introduction de nouveaux événements permet d'affiner la granularité d'observation du système, en ajoutant dans transitions non observables au niveau abstrait, comme le bégaiement dans TLA ([**?**]). Cet aspect du raffinement est souvent codé en introduisant explicitement des variables de contrôle, comme la variable *cc* de l'exemple, ou par des propriétés sur des variables du système.

Ces aspects du raffinement ont différents impacts sur la visualisation. Le premier point ne modifie pas intrinsèquement la dynamique du système, et donc sa visualisation. Néanmoins, la réduction du non-déterminisme ou le renforcement des gardes peut simplifier la représentation. Nous pouvons donc construire de manière systématique un système de transitions associé à un raffinement, en préservant la structure du système de transitions associé à la spécification abstraite (voir section 4.1). Ceci correspond à la notion de *least refinement* introduite par J. Von Wright [**?**]. L'aspect du raffinement qui a le plus d'influence sur la dynamique du système est l'introduction de nouveaux événements. Elle permet de préciser le comportement pour une granularité plus fine d'observation. Le fait d'être dans un état au niveau abstrait peut être représenté par de la dynamique au niveau raffiné. Pour décrire cela nous utilisons la notion d'état hiérarchique, comme on peut la trouver dans les statecharts [**?**]. Dans la section 4.2 nous décrivons cette approche en proposant différentes constructions possibles pour les transitions, suivant le degré de précision désiré.

Dans les sections qui suivent, S désignera une spécification, R son raffinement par l'invariant de liaison L et T un système de transitions. De plus, E_S et F_S désigneront deux états quelconques de S.

4.1 Projection selon L

Il est possible de dériver un système de transitions raffiné T_R à partir du système T_S associé à la spécification abstraite, et de la relation de raffinement L. T_R est appelé le projeté de T_S par L. Afin d'introduire la définition de la projection d'un système, définissons la projection d'un état. On suppose ici que l'ensemble x_S des variables de S est disjoint de l'ensemble x_R des variables de R. Dans le cas où certaines variables sont conservées lors du raffinement, il est toujours possible de les renommer.

Définition 11 Projection d'un état
Soit E_S un état de S. La projection $Proj_L(E_S)$ de E_S selon L est définie de la manière suivante :

$$Proj_L(E_S) \hat{=} \exists x_S \cdot (L \ \wedge \ E_S)$$

Où x_S désigne les variables de S.

Définition 12 Algorithme de projection d'un système de transitions
La projection $Proj_L(T_S)$ *de* $T_S = (N_S, Q_{Init_S}, U_S, W_S)$ *par* L *est définie par :*

1. *Les états de la projection sont les projections des états de* N_S
$$Q_{Init_R} = \mathsf{true} \text{ et } N_R = \{Proj_L(q) \mid q \in N_S - \{Q_{Init_S}\}\} \cup \{Q_{Init_R}\}$$
2. *On applique l'algorithme de construction d'un système de transitions étiquetées suivant, initialement décrit en section 3.3 :*
   - *Calcul d'une transition valide, s'il en existe, pour chaque* (E, Ev, F) *avec :* $E = Q_{Init_R}$, $Ev = Init_R$ *et* $F \in N_R - \{Q_{Init_R}\}$.
   - *Calcul d'une transition valide, s'il en existe, pour chaque* (E, Ev, F) *avec :* $E \in N_R - \{Q_{Init_R}\}$, $Ev \in Interface(R)$ *et* $F \in N_R - \{Q_{Init_R}\}$.

Par construction, $Proj_L(T_S)$ est un système de transitions associé à R, le raffinement de S par L. On a donc $Chemins(Proj_L(T_S)) = Traces(R)$. Cette égalité peut être prouvée selon le même schéma que la démonstration de la section 3.3.

Propriété 2
Si $(Proj_L(E_S), (D', A', Ev_R), Proj_L(F_S))$ *est une projection de la transition* (E_S, (D, A, Ev_S), F_S), *alors on a :*
$$E \ \wedge \ L \ \wedge \ D' \Rightarrow D$$

Cette propriété explicite le fait que toute transition déclenchable dans R doit nécessairement l'être aussi dans S, si le raffinement a été prouvé. Elle peut notamment permettre d'alléger la définition 12. En effet, si $Ev \in Interface(S)$, alors on peut affiner les contraintes sur E lors du calcul d'une transition. Ainsi, pour toute transition (E_S, Ev, H) de l'abstraction on calcule les transitions (E, Ev, F) avec : $E = Proj_L(E_S)$ et $F \in N_R - \{Q_{Init_R}\}$. Aucune autre transition ne pourra être étiquetée par Ev.

4.2 Représentations hiérarchiques

4.2.1 Raffinement des états

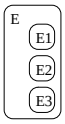


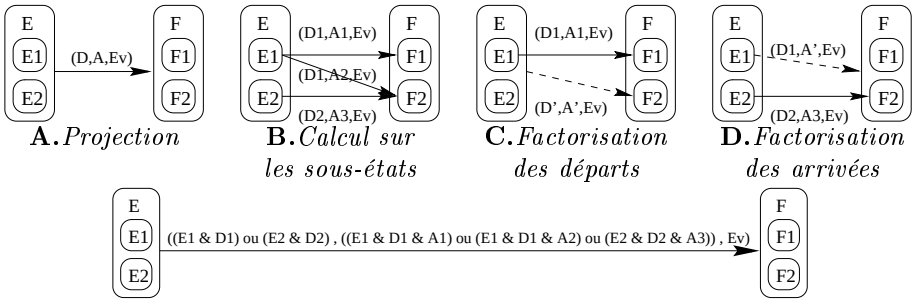
FIG. 6 – État hiérarchisé.

Classiquement, un raffinement donne une vision plus fine de la dynamique d'un système. On est donc souvent amené à introduire des variables de contrôle (cf. exemple fig. 2). Ainsi, nous aimerions définir des états relatifs à ces variables. Pour cela, nous associons un ensemble d'états $\{E_{R_1}, \ldots, E_{R_n}\}$ à chaque état $Proj_L(E_S)$ de R tel que :
$$\bigvee_{i=1}^{n} E_{R_i} \Leftrightarrow Proj_L(E_S)$$

En figure 6, nous donnons l'exemple d'un état E décomposé en E_1, E_2 et E_3. Nous parlons du *sur-état* E et des *sous-états* E_i.

4.2.2 Raffinement des transitions

Plusieurs méthodes permettent de représenter les transitions entre deux états hiérarchisés E et F. La première consiste à construire les transitions selon la définition 12 (ex : figure 7.A). La déclenchabilité et l'atteignabilité sont alors relatives aux états E et F, c'est-à-dire à la disjonction des états E_i et F_i. On peut aussi construire toutes les transitions du raffinement (ex : figure 7.B), en appliquant l'algorithme directement sur les sous-états. L'avantage est que la déclenchabilité et l'atteignabilité peuvent être rendues plus précises.



A. *Projection* **B.** *Calcul sur les sous-états* **C.** *Factorisation des départs* **D.** *Factorisation des arrivées*

E. *Factorisation avec conservation de la sémantique*

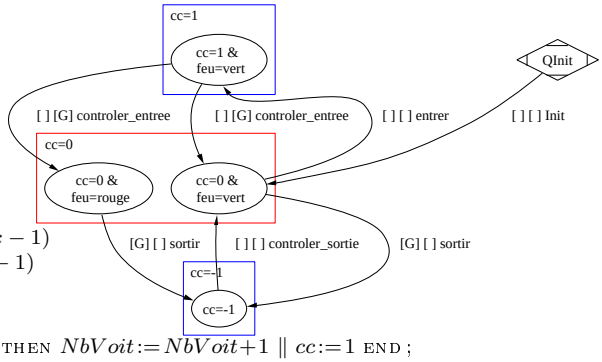
FIG. 7 – Représentations possibles des transitions entre deux sur-états.

Enfin, dans un but d'amélioration de la lisibilité, nous pouvons factoriser les transitions du raffinement. La première solution (flèches pointillées) consiste à factoriser les transitions relatives à un même événement Ev, et dont les conditions d'atteignabilité et de déclenchabilité sont équivalentes pour chaque sous-état (ex : figures 7.C, 7.D). Par exemple, la figure 7.C montre un cas de factorisation où l'on peut factoriser deux transitions atteignant F_2 (figure 7.B) pour n'en former plus qu'une, partant de E et non plus de chacun de ses sous-états. Cette factorisation est valide si $E_1 \Rightarrow (D' \Leftrightarrow D_1)$, $E_2 \Rightarrow (D' \Leftrightarrow D_2)$, $E_1 \wedge D_1 \Rightarrow (A' \Leftrightarrow A_2)$ et $E_2 \wedge D_2 \Rightarrow (A' \Leftrightarrow A_3)$. Nous considérons ici qu'un sur-état est une classe d'états dont la relation de congruence est l'existence d'une certaine transition. La sémantique d'une transition $(E, (D, A, Ev), F)$ factorisée selon cette première méthode est qu'il existe, pour chaque couple (E_i, F_j), une transition valide $(E_i, (D, A, Ev), F_j)$.

La seconde méthode de factorisation que nous allons aborder ici est une méthode systématique permettant de conserver la même sémantique pour les transitions factorisées que pour les transitions entre sous-états (flèches pleines). L'idée est simplement de préciser, dans les conditions de déclenchabilité et d'atteignabilité, les transitions autorisées. Ainsi, la sémantique d'une transition $(E, (D, A, Ev), F)$ est que pour toute valeur de E, si elle vérifie les conditions D et A, alors elle peut mener à F par Ev. Ainsi, les figures 7.B et 7.E sont équivalentes. Lors de l'utilisation de cette seconde méthode, les seule factorisations intéressantes en terme de lisibilité sont celles pour lesquelles les conditions de déclenchabilité et d'atteignaibilité se simplifient.

La figure 8, décrit un raffinement de Parking_r1 (fig. 2) et illustre le calcul des transitions sur les sous-états. Ce raffinement introduit la gestion par le contrôleur d'un feu d'entrée dans le parking.

REFINEMENT $parking\_r2$
REFINES $parking\_r1$
SETS $Couleur\_feu = \{vert, rouge\}$
VARIABLES feu
INVARIANT $feu \in Couleur\_feu$
$\wedge ((cc = 0 \wedge feu = vert) \Rightarrow NbVoit < NbMax)$
$\wedge ((cc = 0 \wedge feu = rouge) \Rightarrow NbVoit = NbMax)$
$\wedge (cc = 1 \Rightarrow feu = vert)$
$\wedge ((cc = -1 \wedge feu = rouge) \Rightarrow NbVoit = NbMax - 1)$
$\wedge ((cc = -1 \wedge feu = vert) \Rightarrow NbVoit < NbMax - 1)$
INITIALISATION $NbVoit := 0 \| feu := vert \| cc := 0$



EVENTS
entrer $\hat{=}$ SELECT $cc{=}0 \wedge feu = vert$ THEN $NbVoit{:=}NbVoit{+}1 \| cc{:=}1$ END ;
controler_entree $\hat{=}$ SELECT $cc{=}1$ THEN IF $NbVoit{=}NbMax$ THEN $feu{:=}rouge$ END $\|cc{:=}0$ END ;
sortir $\hat{=}$ SELECT $NbVoit{>}0 \wedge (cc{=}0)$ THEN $NbVoit{:=}NbVoit - 1 \| cc{:=}-1$ END ;
controler_sortie $\hat{=}$ SELECT $cc{=}-1$ THEN IF $feu{=}rouge$ THEN $feu{:=}vert$ END $\|cc{:=}0$ END
END

FIG. 8 – Exemple du parking raffiné et son système de transitions associé

Comparons les transitions des figures 2 et 8. Nous pouvons constater que l'expansion de l'événement *entrer* a permis de limiter son déclenchement à un seul des sous-états de $cc = 0$. De même, la transition *controler_sortie* n'atteint plus qu'un seul sous-état. Nous pouvons aussi constater que les transitions *sortir* et *controler_entree* sont indépendantes des particularités de chaque sous-état de $cc = 0$. Elles pourraient donc être factorisées entre les sur-états $cc = 0$ et $cc = 1$. Ici, les deux méthodes de factorisation sont applicables.

Par exemple, les transitions :

$$(cc = 0 \ \wedge \ feu = rouge, (NbVoit > 0, true, sortir), cc = -1)$$
$$\text{et } (cc = 0 \ \wedge \ feu = vert, (NbVoit > 0, true, sortir), cc = -1)$$

se factorisent en :

$$(cc = 0, (NbVoit > 0, true, sortir), cc = -1)$$

avec les deux méthodes de factorisation, alors que les transitions :

$$(cc = 1 \ \wedge \ feu = vert, (true, NbVoit = NbMax, controler\_entree), cc = 0 \ \wedge \ feu = rouge)$$
$$\text{et } (cc = 1 \ \wedge \ feu = vert, (true, NbVoit < NbMax, controler\_entree), cc = 0 \ \wedge \ feu = vert)$$

ne peuvent se factoriser qu'avec la seconde méthode (flèche pleine), ce qui donne alors :

$$(cc = 1 \ \wedge \ feu = vert, (true, true, controler\_entree), cc = 0)$$

5 Conclusion

Les travaux présentés ici sont inspirés de l'article [?] dans lequel les auteurs proposent une méthode pour construire des systèmes de transitions qui décrivent une abstraction d'un système B événementiel. Le système construit fait abstraction des gardes, les traces

du système B sont donc inclusent dans les chemins du système de transitions. L'abstraction obtenue permet ainsi de vérifier des propriétés de sûreté, par exemple par model checking. L'extension proposée ici a pour objectif de donner plus d'informations sur les transitions, en explicitant les conditions de franchissement. Les systèmes de transitions manipulés sont donc symboliques et permettent une représentation exacte des comportements du système B. Il est toujours possible de construire une abstraction en oubliant les conditions de franchissement. D'autre part l'approche proposée dans [?] a été étendue pour la prise en compte du processus de raffinement.

Un certain nombre de travaux se sont intéressés à la représentation dans des modèles B des aspects dynamiques d'un système décrit dans un formalisme proche des automates [?] (pour ne citer que l'un des derniers). Dans ce papier nous nous intéressons à l'approche inverse : extraire d'un système B événementiel une représentation sous forme d'automate. La problématique n'est pas la même puisque différentes formes de modélisation sont possibles : le résultat n'est pas un codage univoque du modèle B. Une approche similaire a déjà été envisagée pour TLA (la logique temporelle des actions de L. Lamport) d'abord dans [?], puis étendue successivement dans [?] et [?] pour prendre en compte certaines propriétés de vivacité ainsi que le raffinement. Comme dans [?] les diagrammes construits décrivent des abstractions du comportement du système. Ces travaux n'introduisent pas de méthode systématique permettant de calculer le système de transitions.

L'approche proposée dans ce papier est en partie implémentée dans l'outil GénéSyst. Partant d'une spécification B contenant les informations sur les états, GénéSyst produit des systèmes de transitions qui peuvent être actuellement visualisés à l'aide des outils GraphViz, de AT&T, et CADP [?]. Les obligations de preuve qui permettent d'établir les prédicats de déclenchabilité et d'atteignabilité sont actuellement obtenues en utilisant l'*AtelierB*. GénéSyst produit donc des machines dont les assertions contiennent les formules correspondant à la figure 4 et le prouveur de l'*AtelierB* permet de valider ou non ces obligations de preuve. L'outil ne supporte pas encore la factorisation des transitions, comme proposé dans la section 4.2. La suite du travail consiste d'une part à valider l'approche et l'outil sur des études de cas et d'autre part à développer l'approche proposée. En effet l'objectif n'est pas tant la visualisation de développement B par des systèmes de transition, que le développement conjoint de spécifications alliant des systèmes de transitions et des modélisations événementielles. En effet, dans la pratique, la construction d'un modèle B événementiel nécessite de raisonner globalement sur la dynamique, à la fois pour établir les gardes des événements et pour élaborer les invariants. Ce processus devient accru dans le cas du raffinement où les invariants sont de plus en plus complexes. L'objectif est donc d'exploiter une représentation explicite du comportement pour aider à la construction d'invariants, comme ceci peut être fait par exemple dans les systèmes de transition [?].