Compositionnalités en B classique et étendu

tour d'horizon

Samuel Colin[1]

[1] Samuel.Colin@univ-valenciennes.fr
LAMIH/ROI, UMR CNRS 8530
Université de Valenciennes

CNAM, 2006-07-12

Plan

Sémantique (?) de la modularité
Exemple

Tiré de [Abrial, 1996] :

- Invariant : $b\_inf \leq b\_sup$
-

Increment_inf	Decrement_sup
PRE b_inf <b_sup **THEN** b_inf := b_inf + 1 **END**	**PRE** b_inf <b_sup **THEN** b_sup := b_sup - 1 **END**

- Chacune des opérations vérifie l'invariant
  $b\_inf \leq b\_sup \wedge b\_inf < b\_sup \Rightarrow b\_inf + 1 \leq b\_sup$
- Problème : $Increment\_inf \,||\, Decrement\_sup$

Le problème tel qu'il m'intéresse

red_light_door	close_gate
BEGIN start_red_light ; await gate =closed ; stop_red_light ; **POST** red_light =off **REQUIRES** $[\![red\_light = on]\!]$ **END**	**BEGIN** await red_light =on ; start_closing ; continue_closing ; end_closing ; **POST** gate =closed **REQUIRES** $[\![gate = closing]\!]$ **END**

Plan

Modularité inchangée

- [Bodeveix et al., 1999] redéfinit la sémantique du $\|$ (composants B usuels inchangés)
- [Treharne et al., 2003] fait appel à CSP, le composant de base est la machine B

Sémantique de la modularité

- [Bert et al., 1996] redéfinit la sémantique du $\|$ et fonde la modularité sur une algèbre de composants
- [Petit, 2003] propose une approche “langage” en instanciant un système de modules pour B à partir du langage de base
  $\Rightarrow$ Un plus grand nombre de langages cibles (ayant des clauses de modularité spécifiques)

Manipulation de la modularité

- [Büchi and Back, 1999] propose la composition à variables partagées.
  - La notion de composant est globalement inchangée
  - L'interface des machines est augmentée via des *contrats*
- [Lano et al., 1996] propose de voir les machines B comme des entités autonomes interrogeables (plutôt qu'appelables)
  - Les clauses de modularité ne changent pas
  - C'est la manière dont est vue une machine B qui change (composant qui "écoute" des requêtes)

Plan

Sémantique de la composition
Assemblage de composants

- [Bert et al., 1996] :

$$\begin{aligned} S \otimes T \quad \equiv \quad & trm(S) \wedge trm(T) \mid \\ & @x', y', z' . (prd_{z,x}(S) \wedge prd_{z,y}(T) \\ & \Longrightarrow \{z, x, y\} := \{z, x, y\}') \end{aligned}$$

$$(v := 1) \otimes (v := 2) \equiv (v :\in \emptyset)$$

- INCLUDES et USES définis en termes de primitives (algébriques) de composition : promotion, *hiding*, instanciation de paramètres, renommage, composition (cf au-dessus), enrichissement

Sémantique de la composition
(suite...)

- [Bodeveix et al., 1999] :

$$
\begin{aligned}
S \parallel T \quad \equiv \quad & trm(S) \wedge trm(T) \mid \\
& @\{x,y\}'.(prd_{x\setminus(x\cap y)}(S) \wedge prd_{y\setminus(x\cap y)}(T) \wedge \\
& (prd_{x\cap y}(S) \vee prd_{x\cap y}(T)) \\
& \Longrightarrow \{x,y\} := \{x,y\}')
\end{aligned}
$$

Composition laissée à un autre formalisme

[Treharne et al., 2003] combine CSP et B :

- Un composant est constitué d'une machine B et d'un contrôleur CSP :
  $RCTrl = inc \rightarrow inc\_or\_dec \rightarrow reset \rightarrow RCTrl$
- Seuls les contrôleurs communiquent entre eux
- Les machines B n'interagissent jamais directement
- La composition correspond à la composition parallèle de CSP
- Les propriétés de sûreté sont inférées via la sémantique de la combinaison d'un composant B et d'un contrôleur CSP

Composition en *assumption/commitment*

[Büchi and Back, 1999] : soit une machine qui peut être partagée par plusieurs autres machines, la méthode se base sur la vérification d'absence d'interférence :

- Une machine partagée M_s définit des rôles d'utilisation $R_1, R_2, \ldots$ (accès en lecture, en écriture)
- Un rôle $\equiv$ une manière d'appeler les opérations de la machine partagée (séquence, choix borné, etc)
- D'autres machines M incluent cette machine selon les rôles définis
- Il faut vérifier que les rôles *non définis* par la machine courante n'interfèrent pas avec l'invariant de la machine courante
- Par exemple, si M utilise le rôle R_1, il faut vérifier que $R_2, \ldots$ n'entre pas en conflit avec l'invariant de M

Composition par interface

[Lano et al., 1996] :

- Chaque machine est un composant interrogeable, et définit une clause `THREAD` précisant le mode d'interrogation et de réponse des opérations
- La composition (concurrente) est obtenue en définissant une machine asynchrone qui «lance» l'appel d'opération et attend la complétion de celle-ci
- Il s'agit d'une concurrence sans variable partagée (modularité «B classique»)

B événementiel

[RODIN, 2005] prend un point de vue global, donc il n'y a pas de développement «séparé». Cependant :

- La notion de décomposition/recomposition est harmonieuse au niveau des événements (disjonction des gardes)
- La décomposition est en fait similaire au raffinement
- Les contraintes de décomposition sont similaires à des contraintes d'assumption/commitment
- Peu d'exemples illustrant ces propriétés, cependant

Composition en sémantique temporelle

red_light_door	close_gate
BEGIN start_red_light ; await gate =closed ; stop_red_light ; **POST** red_light =off **REQUIRES** $[\![red\_light = on]\!]$ **END**	**BEGIN** await red_light =on ; start_closing ; continue_closing ; end_closing ; **POST** gate =closed **REQUIRES** $[\![gate = closing]\!]$ **END**

Composition en sémantique temporelle
(suite)

[Colin et al., 2004] :

- Seules les contraintes temporelles des opérations incluses sont utilisées
  $\Rightarrow$ compositionnel, temporellement parlant
- Il faut cependant toujours connaître les corps des opérations incluses, en parcourant transitivement les chaînes d'inclusion :
  - Peut être amoindri par le mécanisme d'abstraction de B
  - L'entrelacement a malheureusement besoin de spécifications précises (d'un point de vue «sémantique opérationnelle»)

Ce dont je n'ai pas parlé

- Utilisation de B avec UML
- D'autres travaux de la communauté qui me sont inconnus

Plan

La modularité de B
Problèmes

- Peu claire à l'origine
- Contraintes de composition de machines peu naturelles
- Sémantique de la composition embryonnaire

La modularité de B
Les solutions proposées

- Méthodes utilisées :
  - Des changements/extensions de la sémantique
  - Utilisation avec un formalisme plus adapté
- Difficultés rencontrées :
  - Partage de variables
  - Rester compatible avec le raffinement
- Résultats obtenus :
  - Se rapprocher d'une sémantique opérationnelle rend le raffinement plus ardu
  - Les méthodes de type assumption/commitment sont globalement plus adaptables, en fonction de la sémantique utilisée

Autres travaux

- (Courte) exploration de la temporalité dans UML
- Principal développeur de BRILLANT ([BRILLANT,],[Colin et al., 2005])
- Exploration de la preuve de projets B (avec PhoX)

Compétences

- Logiques, logiques temporelles
- Assistants de preuve
- Sûreté de systèmes concurrents et/ou à contraintes temporelles
- TEX/LATEX
- OCaml
- XML, XSL
- PHP, XHTML, CSS
- Blender, Gimp, Inkscape
- Libriste en général, Debianiste en particulier

Références I


Abrial, J.-R. (1996).
The B Book - Assigning Programs to Meanings.
Cambridge University Press.


Bert, D., Bowen, J. P., King, S., and Waldén, M., editors (2003).
ZB'2003 – Formal Specification and Development in Z and B,International Conference of B and Z Users, Turku, Finland, June 4-6, 2003, Proceedings, volume 2651 of *Lecture Notes in Computer Science (Springer-Verlag)*, Turku, Finland. Springer.


Bert, D., Potet, M.-L., and Rouzaud, Y. (1996).
A study on components and assembly primitives in B.
In [Habrias, 1996], pages 47–62.


Bodeveix, J.-P., Filali, M., and Munoz, C. (1999).
A formalization of the B method in Coq and PVS.
In [BUGM99, 1999], pages 32–48.


BRILLANT.
BRILLANT.
http://gna.org/projects/brillant.

Références II


Büchi, M. and Back, R. (1999).
Compositional symmetric sharing in B.
In [Wing et al., 1999], pages 431–451.


BUGM99 (1999).
FM'99 – B Users Group Meeting – Applying B in an industrial context : Tools, Lessons and Techniques. Springer-Verlag.


Colin, S., Mariano, G., and Poirriez, V. (2004).
Duration calculus : A real-time semantic for B.
In *First International Colloquium on Theoretical Aspects of Computing*, Guiyang, China. UNU-IIST.


Colin, S., Petit, D., Rocheteau, J., Marcano, R., Mariano, G., and Poirriez, V. (2005).
BRILLANT : An open source and XML-based platform for rigourous software development.
In *SEFM (Software Engineering and Formal Methods)*, Koblenz, Germany. AGKI (Artificial Intelligence Research Koblenz), IEEE Computer Society Press.


Habrias, H., editor (1996).
Proceedings of the 1st Conference on the B method, Putting into Practice methods and tools for information system design, 3 rue du Maréchal Joffre, BP 34103, 44041 Nantes Cedex 1. IRIN Institut de recherche en informatique de Nantes.

Références III


Lano, K., Fiadeiro, J., and Dick, J. (1996).
Extending B AMN with concurrency.
Technical report, Dept. of Computing, Imperial College.


Petit, D. (2003).
Génération automatique de composants logiciels sûrs à partir de spécifications formelles B.
Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis.


RODIN (2005).
`http://rodin.cs.ncl.ac.uk/`.


Treharne, H., Schneider, S., and Bramble, M. (2003).
Composing specifications using communication.
In [Bert et al., 2003], pages 58 – 78.


Wing, J. M., Woodcock, J., and Davies, J., editors (1999).
Proceedings of FM'99 : World Congress on Formal Methods, number 1709 in Lecture Notes in Computer Science (Springer-Verlag). Springer Verlag.

Questions ?