Contribution à l'intégration de temporalité au formalisme B

Utilisation du calcul des durées en tant que sémantique temporelle pour B

Samuel Colin[1]

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

INRETS/ESTAS
Villeneuve d'Ascq

UVHC, Mardi 3 octobre 2006

Problématiques de sûreté
Exemples

- Le train doit s'arrêter à un signal de feu rouge (sûreté)
- Un train en panne respecte cette spécification $\Rightarrow$ tout train qui part d'une gare arrive à la suivante, en moins de n heures (vivacité)
- le contrôleur de vitesse doit coopérer avec le système de traitement des signaux (équité)

Problématiques de sûreté
Informatisation

- Utilisation de systèmes automatisés puis informatisés
- Généralisation aux systèmes critiques (transport, économie)
- Comment garantir que le système conçu répond à ces exigences de criticité ?

$\Rightarrow$ Méthodes de conception adaptées

- Les programmes doivent interagir avec l'environnement dans lequel il se trouvent
- Des phénomènes physiques doivent être pris en compte

$\Rightarrow$ Ces contraintes physiques font intervenir le temps

Méthode B
Présentation

- Parmi les exemples de succès industriels, [Behm et al., 1999]
- Basée sur la théorie des ensembles et la logique classique
- L'évolution *dynamique* des composants B est modélisée par les substitutions. Leur sémantique est basée sur les transformateurs de prédicat
- Raffinement : un composant peut être remplacé par un autre, pourvu qu'il se comporte de la même manière

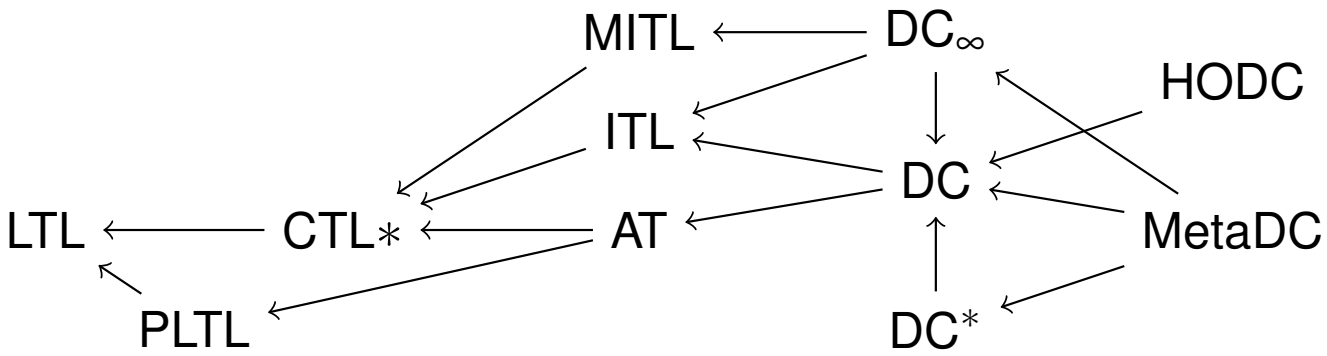
$\Rightarrow$ Qu'en est-il de son utilisation dans le cadre de contraintes temporelles ?

Problématique
Questions posées

- Aider les industriels à conserver leur expertise : B est-il suffisant pour exprimer toute contrainte temporelle ?
- S'aider de formalismes tiers conçus dans ce contexte : quels liens peuvent exister entre B et ces formalismes ?
- Valider les systèmes proposés : Quelles approches de validation pour ces formalismes ?

Plan

Logiques temporelles
Classification



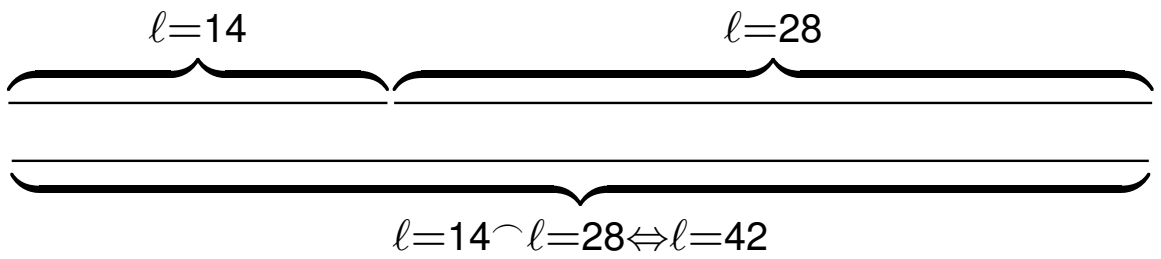
	LTL_{77}	CTL^*_{93}	$PLTL_{99}$	$MITL_{91}$	AT_{90}	ITL_{85}	DC_{91}
Alternatives		√		√	√	√	√
Temps quantifié			√	√	√	√	√
Temps continu				√	√	√	√
Temps dense						√	√
Exemple	×	×	×	×	×	?	√

[Bouajjani et al., 1995] :

$$\mathit{Exemple} \equiv \Box\left(\exists x.\left(\begin{array}{l}(([\![\neg S]\!] \wedge \ell = x)^\frown [\![S]\!]) \wedge \ell > x+1) \\ \Rightarrow \ell = x+1^\frown [\![\neg S]\!]^\frown \mathbf{true}\end{array}\right)\right)$$

Dans tout sous-intervalle, si S survient, S durera moins d'une seconde.

- Logique classique
- Découpage d'intervalle "chop" $\frown$
- Variable spéciale ℓ (longueur d'intervalle), définie sur $\mathbb{R}$
- Exemple :



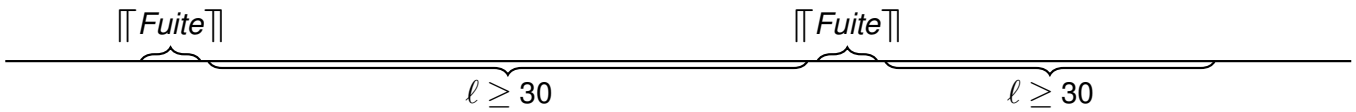
Logiques temporelles
Calcul des durées (DC)

- Logique d'intervalle + durée d'états $\int$
- États (événements) : $0, 1, \vee, \neg$
- Notations :

$[\![S]\!]$	$\equiv$	$\int S = \ell \wedge \ell > 0$	«S est tout le temps vrai»
$\Diamond P$	$\equiv$	$True^\frown P^\frown True$	«il existe un sous-intervalle vérifiant P»
$\Box P$	$\equiv$	$\neg \Diamond \neg P$	«tout sous-intervalle vérifie P»

- Exemple : soit $Fuite \equiv Gaz \wedge \neg Flamme$

$$\Box([\![Fuite]\!] \Rightarrow \ell < 1)$$

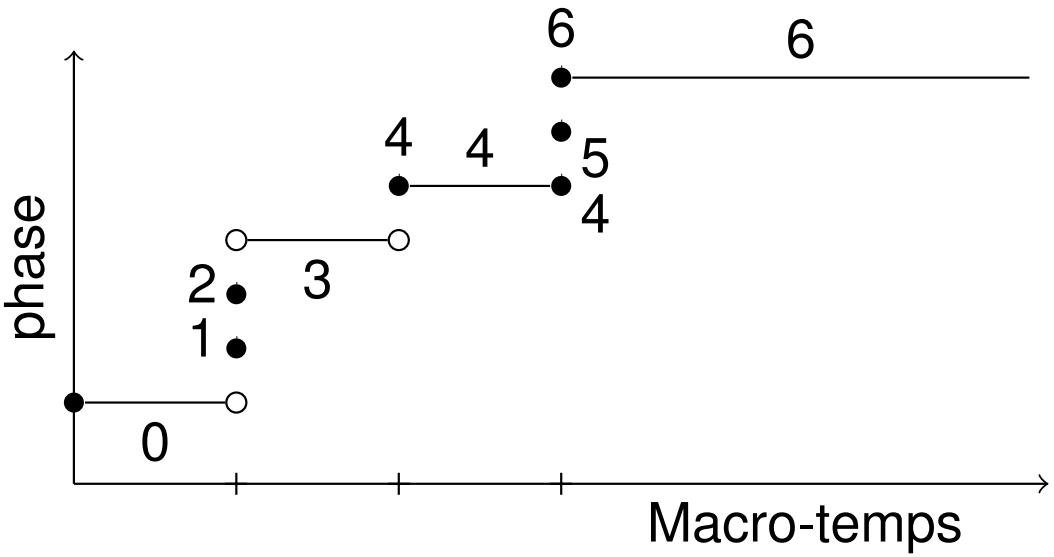


$$\Box([\![Fuite]\!]^\frown[\![\neg\ Fuite]\!]^\frown[\![Fuite]\!] \Rightarrow \ell \geq 30)$$

$$\Rightarrow \ell > 60 \Rightarrow 20 \int Fuite < \ell$$

- Opérateur de *répétition* : découpage arbitraire d'un intervalle en sous-intervalle
- Exemples :
  - $\overbrace{(\ell = 1)^* \wedge \ell = 8}^{\underbrace{1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1}}$
  - $\overbrace{[\![S]\!]^* \wedge \ell = 8}^{\underbrace{[\![S]\!] \qquad [\![S]\!] \qquad [\![S]\!] \qquad [\![S]\!]}}$

- DC au temps faiblement monotone, avec itération
- DC* + *micro-temps* (phase, progression d'un état)
- Variable spéciale η, valeur entière



Logiques temporelles
Ce qu'elles apportent

- Modélisation fine du comportement dynamique
- Peuvent être exprimés : concurrence, non-terminaison, ordonnancement, intervalles infinis, ...
- *Qui peut le plus peut le moins* :
  - Modélisations continue et discrète du temps
  - D'autres formalismes en forment des sous-classes

Plan

Méthode B
expression du temps/de la concurrence

- Méthode B pure :
  - Plusieurs variables représentant des horloges $\Rightarrow$ Preuve plus difficile
  - Traduction d'un fragment de TLTL en B [Bodeveix et al., 2004] $\Rightarrow$ limité à ce fragment de TLTL, les conjonctions multiplient quadratiquement la taille des machines résultantes
- B associé à CSP [Schneider and Treharne, 2002] :
  - Les composants B sont des briques de base
  - L'architecture/l'interaction des composants est obtenue avec CSP

$\Rightarrow$ la temporalité se réduit à des variables d'horloge

Méthode B
expression du temps/de la concurrence (suite)

- B étendu :
  - Expression de la concurrence en B [Lano and Dick, 1996] :
    - Les machines sont des composants autonomes interrogeables
    - Des clauses supplémentaires expriment des protocoles d'utilisation

    $\Rightarrow$ limité à une logique temporelle discrète
  - Automates des régions exprimés sous forme de modèles eventB [Hammad et al., 2003] : $\Rightarrow$ expressivité afférente aux automates des régions

$\Rightarrow$ Au vu des propriétés de DC énoncées plus haut, il est intéressant d'étudier un «B+DC»

- Sémantique des substitutions avec WDC* (*opérationnelle* et non plus dénotationnelle)
- Conservation des propriétés des substitutions : démarrée dans un état vérifiant la précondition, la postcondition est vérifiée à la fin de l'exécution
- Obtention des propriétés temporelles des substitutions (WDC* $\longrightarrow$ DC*)
- Bonus : introduction de nouvelles substitutions
- Basé sur [Siewe and Hung, 2001]
- Présenté dans [Colin et al., 2004]

Sémantique WDC* de B
Principe

- Idée : modéliser un calculateur/ordonnanceur en WDC*
- Hypothèses :
  - $Run_i \Rightarrow Wait_i$
  - $[\![\neg\,(Run_1 \wedge Run_2)]\!]^+$
  - $[\![(Wait_1 \vee Wait_2) \Rightarrow (Run_1 \vee Run_2)]\!]^+$
- Les substitutions sont exprimées en terme de longueur d'intervalle, et de requête de fonctionnement

Sémantique WDC* de B
Exemple

$$\mathcal{M}_{fin}(\text{delay } d \,|||\, x := E) \equiv [\![\neg\, Wait_1]\!]^+ \frown Unit \frown \lceil 1 \rceil^0 \wedge \ell = d \wedge \lceil Run_2 \rceil^1 \frown \lceil x = E_0 \rceil^0 \wedge \ell = 0$$

$$\lceil Run_2 \rceil^1 \frown \overbrace{[\![\neg\, Wait_1]\!]^+ \frown Unit \frown \lceil 1 \rceil^0 \wedge \ell = d}^{\lceil x = E_0 \rceil^0 \wedge \ell = 0}$$

$$\mathcal{M}_{fin}(\text{delay } d \,|||\, x := E) \equiv (\eta = 1 \wedge \ell = 0 \wedge \lceil Wait_1 \wedge x = E_0 \rceil) \frown [\![\neg\, Wait_1]\!]^+ \frown Unit \frown \lceil 1 \rceil^0 \wedge \ell = d$$

Sémantique WDC* de B

Conséquences

- Conservation des propriétés des substitutions : $\lceil [S]P \rceil^0 \frown \mathcal{M}_{fin}(S) \Rightarrow \mathcal{M}_{fin}(S) \frown \lceil P \rceil^0$
- Obtention des propriétés temporelles des substitutions : $\prod(\lceil [S]P \rceil^0 \frown \mathcal{M}_{fin}(S)) \Rightarrow \mathrm{dur}([S], P)$
- Bonus : obtention de nouvelles substitutions définies fonctionnellement et temporellement (délai, attente réactive, $[S \,|||\, T]$)
- Besoin supplémentaire : disposer d'un état à partir duquel calculer la formule de durées

Sémantique WDC* de B

Nouvelles substitutions (postcondition)

Postcondition :

- Découle du besoin d'un état à partir duquel calculer la formule temporelle
- En théorie, se rapproche d'un raffinement
- En pratique, focalisé sur les variables de l'opération, i.e. moins de variables que l'invariant

$$\begin{aligned} trm(S \rhd Post) &\equiv [x_0 := x]([S]Post) \\ prd_x(S \rhd Post) &\equiv [x_0, x := x, x']Post \end{aligned}$$

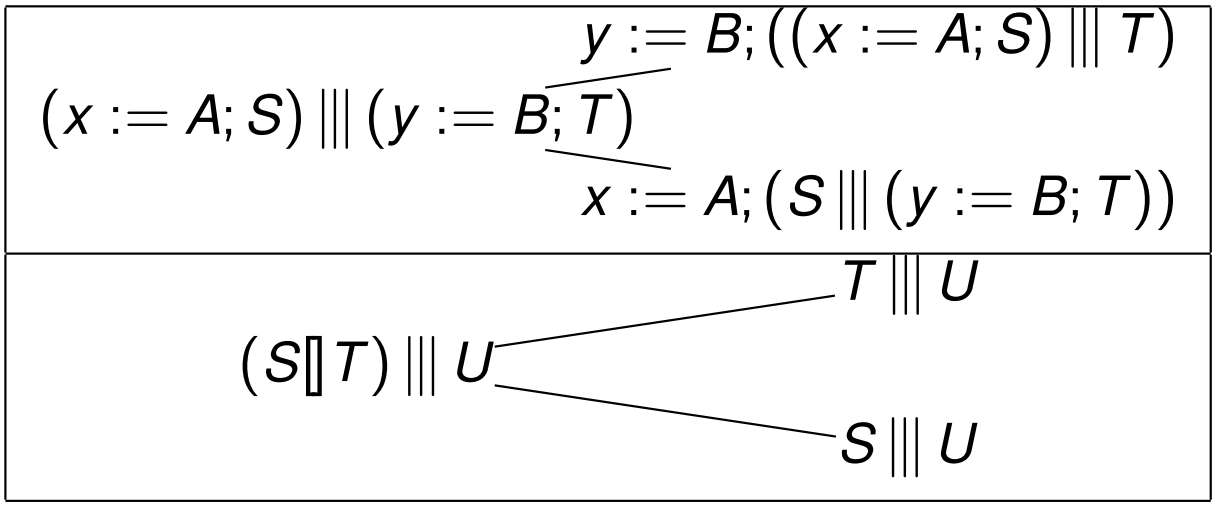
Sémantique WDC* de B

Nouvelles substitutions

- Délai : delay 0.01
  - Fonctionnellement : $[\text{delay } 0.01]P \equiv P$
  - Temporellement : $\ell = 0.01$
- Attente réactive : await $(x = 0)$
  - Fonctionnellement : $[\text{await } (x = 0)]P \equiv (x = 0 \Rightarrow P)$
  - Temporellement : $[\![x \neq 0 \wedge P]\!]^*$
- Concurrence : $S \,|||\, T$
  - Fonctionnellement : calcul d'entrelacement
  - Temporellement : $dur(S, Post) \wedge dur(T, Post)$

Sémantique WDC* de B
Calcul d'entrelacement

Définition de l'entrelacement (non défini par Siewe et Hung)



- Variables partagées
- Les contraintes de B sur les variables restent valides

$\Rightarrow$ Le partage de variables ne se fait que dans la machine où elles sont définies

Machines B temporisées
Exemple

x ⟵Exemple =
PRE
x ≥1
THEN
(delay 2 ; x :=x-2) |||(delay 1 ; x :=x+1 ; delay 1) ;
POST
x ≥0
TIMING
□(⟦x ≥0 ⟧)
END

$x \geq 1 \Rightarrow [\text{delay } 1; x := x+1; \text{delay } 1; x := x-2](x \geq 0)$

$\Rightarrow \quad x \geq 1 \Rightarrow x+1-2 \geq 0$

Machines B temporisées
Exemple

x ⟵Exemple =
PRE
x ≥1
THEN
(delay 2 ; x :=x-2) |||(delay 1 ; x :=x+1 ; delay 1) ;
POST
x ≥0
TIMING
□(⟦x ≥0 ⟧)
END

$$\left(\begin{array}{l} \ell = 2 \wedge \llbracket x - 2 \geq 0 \rrbracket \\ \wedge (\ell = 1 \wedge \llbracket x + 1 \geq 0 \rrbracket)^\frown (\ell = 1 \wedge \llbracket x \geq 0 \rrbracket) \end{array} \right) \Rightarrow \Box(\llbracket x \geq 0 \rrbracket)$$

Machines B temporisées
Exemple (modularité)

x ⟵Exemple =
PRE
x ≥1
THEN
(delay 2 ; x :=x-2) |||(delay 1 ; x :=appel(x)) ;
POST
x ≥0
TIMING
□(⟦x ≥0 ⟧)
END

Machines B temporisées
Exemple (modularité)

- Supposons que `appel(x)` vérifie $[\![x \geq 1]\!] \wedge \ell = 1$
- L'appel d'opération est remplacé par la formule temporelle que l'opération vérifie
- Obligation de preuve temporelle :

$$\left(\begin{array}{l} \ell = 2 \wedge [\![x - 2 \geq 0]\!] \\ \wedge (\ell = 1 \wedge [\![[x := \mathit{appel}(x)](x \geq 0)]\!]) \frown ([\![x \geq 1]\!] \wedge \ell = 1) \end{array} \right) \Rightarrow \Box([\![x \geq 0]\!])$$

Machines B temporisées
Exemple (raffinement)

$x \longleftarrow \text{Exemple} =$

PRE

$x \geq y$

$\wedge\ y \geq 1$

THEN

(delay 2 ; x :=x-2 ; y := y + x)

$|||$(delay 1 ; x := x+1 ; delay 1) ;

POST

$x \geq 0$

$\wedge\ y \geq x$

END

Machines B temporisées
Exemple (modularité)

- La formule temporelle de l'opération raffinée implique la formule temporelle de l'opération abstraite

$$\left(\begin{array}{l} \ell = 2 \wedge [\![x - 2 \geq 0 \wedge y \geq 0]\!] \\ \wedge (\ell = 1 \wedge [\![x + 1 \geq 0 \wedge y \geq x + 1]\!]) ^\frown (\ell = 1 \wedge [\![x \geq 0 \wedge y \geq x]\!]) \end{array}\right)$$

- $\Rightarrow$

$$\left(\begin{array}{l} \ell = 2 \wedge [\![x - 2 \geq 0]\!] \\ \wedge (\ell = 1 \wedge [\![x + 1 \geq 0]\!]) ^\frown (\ell = 1 \wedge [\![x \geq 0]\!]) \end{array}\right)$$

- Plutôt vers les dernières étapes de raffinement

- Permet de développer des systèmes possédant des contraintes temporelles fortes ($\equiv$ exprimables par DC*)
- S'appuie sur la sémantique en plus faible précondition sans l'invalider
- Est automatisable
- Exploite partiellement la modularité
- Laisse la possibilité de raffinement vers les dernières étapes, plus concrètes

$\Rightarrow$ Comment vérifier les formules temporelles produites ?

Plan

Vérification de DC
Approches existantes

- Approche par vérification de modèle [Pandya, 2001] (limité à certaines classes de formules)
- Approche par traduction en système de contraintes linéaires [Enslev and Nielsen, 2005] (limité par le solveur desdites contraintes, et à des bornes finies du temps)
- Approche axiomatique qui suit la sémantique [Skakkebæk, 1994] (efficacité limitée par les clauses existentielles)
- Approche axiomatique qui adapte le système de preuve
  - En Isabelle/HOL ($\Rightarrow$ réimplémentation *totale*)
  - Inclut un outil de décision arithmétique
  - N'est plus maintenu

$\Rightarrow$ notre idée : adapter le système de preuves dans un outil générique en ayant le maximum de réutilisabilité
[Colin et al., 2003].

Coq
Rappel(s)

- Basé sur le calcul des constructions inductives, un lambda-calcul typé doté de :
  - Types dépendants
  - Constructions de types
  - Types polymorphes
  - Mécanisme de définition inductive des types, avec un système de réduction pour rendre les calculs plus efficaces
- Dispose de moyens de redéfinitions syntaxiques
- Permet d'accueillir virtuellement n'importe quelle logique à n'importe quel ordre
- Est activement maintenu et utilisé industriellement (Trusted Logic, Dassault, Philips, France Telecom)

$\Rightarrow$ pourquoi ne pas l'utiliser pour DC ?

Implémentation de DC* en Coq

Modalité de la variable ℓ

Problème :

$$\frac{\ell = 3 \vdash \ell = 1^\frown \ell = 2}{\ell = 3 \vdash 3 = 1^\frown 3 = 2}$$

Solutions :

- Isabelle/HOL : une autre définition de l'égalité
- Coq : rendre ℓ prédicatif

$$\ell_= \ell_< :\vdash \ell_= x \Rightarrow \ell_< y \Rightarrow x < y$$

Implémentation de DC* en Coq

Inférence de □

Problème :

$$\frac{P}{\Box P}$$

Solutions :

- Isabelle/HOL : manipulation des hypothèses
- Coq : «nécessiter» les axiomes

$$\Box(P \Rightarrow Q \Rightarrow P \wedge Q)$$

Problème : $\int 1 = \ell$

Solutions :

- Isabelle/HOL : similaire à la solution pour ℓ (modification de l'égalité)
- Coq : similaire à la solution pour ℓ (prédicativité)

Implémentation de DC* en Coq

Exploitation des fonctionnalités de Coq

- Définition de l'opérateur d'itération :

$$P^k = \underbrace{P^\frown \ldots ^\frown P}_{k \text{ fois}} \quad P^* = \bigvee_{n \in \mathbb{N}} P^n$$

- En Coq :

```
Fixpoint repetition (p:Prop) (n:nat) {struct n} :Prop :=
match n with
 | 0 => (length_eq 0%R)
 | (S k) => chop (repetition p k) p
end.

Fixpoint repetition_closure
    (p:Prop) (n:nat) {struct n} :Prop :=
match n with
 | 0 => repetition p 0
 | (S k ) => (repetition p n) \/ (repetition_closure p k)
end.
```

Implémentation de DC* en Coq
Conclusion

- Un outil pour prouver des formules de DC*
- Le plongement léger réutilise l'existant (logique de base, bibliothèque sur les réels)
- Ne se limite à aucune sous-classe de DC*

Plan

Résultats

- Au niveau de l'extension de B :
  - Validation de contraintes temporelles
  - Nouvelles substitutions démontrées correctes fonctionnellement : attente, concurrence
  - Le mécanisme de validation fonctionnelle ne change pas
  - Raffinement temporel $\equiv$ implication logique
- Au niveau de DC :
  - Plongement léger dans un outil de preuve : étendre en faisant le moins d'adaptations possibles
  - Appréhender les modalités

Conclusion

- Utiliser DC* pour étendre (conservativement) une méthode formelle :
  - Est faisable, et fondé théoriquement
  - Facilite l'expression de contraintes temporelles et de leurs preuves
- Il est possible d'utiliser un plongement léger dans un outil de preuve générique pour implémenter une logique modale

Perspectives

- Un outil pour la génération d'obligations de preuves
- Quelle logique temporelle pour eventB ?
- Plongement profond de (Meta ?)DC dans Coq
- Unifier :
  - Les travaux sur la composition concurrente de machines/modèles B
  - Les travaux qui intègrent le raffinement aux méthodes d'assumption/commitment
  - Dans un cadre de logique temporelle (MetaDC)

Questions ?

Merci de votre attention

Références I


AFADL2003 (2003).
Approches Formelles dans l'Assistance au Développement de Logiciels, IRISA Rennes – France. IRISA, IRISA.


Behm, P., Benoit, P., Faivre, A., and Meynadier, J.-M. (1999).
Meteor : A successful application of B in a large project.
In *World Congress on Formal Methods 1999*, number 1709 in Lecture Notes in Computer Science, pages 369–387. Springer Verlag.


Bodeveix, J.-P., Filali, M., and Rached, M. (2004).
Méthodes de spécification de systèmes temps réel en B.
In *FAC2004 (Formalisation des Activités Concurrentes)*, Toulouse. CERT-ONERA.
`http://www.cert.fr/feria/svf/FAC/2004/actes.html`.


Bouajjani, A., Lakhnech, Y., and Robbana, R. (1995).
From duration calculus to linear hybrid automata.
In Wolper, P., editor, *Proceedings of the 7th International Conference On Computer Aided Verification*, volume 939, pages 196–210, Liege, Belgium. 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.

Références II


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.


Colin, S., Poirriez, V., and Mariano, G. (2003).
Thoughts about the implementation of the duration calculus with coq.
In *4th International Workshop on the Implementation of Logics*, volume Technical report ULCS-03-018. University of Liverpool.
http://www.csc.liv.ac.uk/research/techreports/.


Enslev, J. and Nielsen, A.-S. (2005).
Bounded model construction for duration calculus.
Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby. Supervised by Assoc. prof. Martin Franzle and Assoc. prof. Michael R. Hansen.


Hammad, A., Julliand, J., Mountassir, H., and Ossami, D. O. (2003).
Expression en B et raffinement des systèmes réactifs temps réel.
In [AFADL2003, 2003], pages 211–226.

Références III

Lano, K. and Dick, J. (1996).
Development of concurrent systems in B AMN.
In Jifeng, H., Cooke, J., and Wallis, P., editors, *BCS-FACS 7th Refinement Workshop*, Electronic Workshops in Computing. Springer-Verlag.

Marcano, R., Colin, S., and Mariano, G. (2004).
A formal framework for uml modelling with timed constraints : Application to railway control systems.
In *SVERTS : Specification and Validation of UML models for Real Time and Embedded Systems*, Lisbon, Portugal.
(in conjunction with 7th International Conference on the Unified Modeling Language, UML 2004).

Pandya, P. K. (2001).
Specifying and deciding quantified discrete-time duration calculus formulae using dcvalid.
In *RT-TOOLS'2001*, Aalborg. (affiliated with CONCUR 2001).
Technical report TCS-00-PKP-1, Tata Institute of Fundamental Research, Mumbai, 2000.

Références IV

Rocheteau, J., Colin, S., Mariano, G., and Poirriez, V. (2004).

Évaluation de l'extensibilité de phox : B/phox un assistant de preuves pour b.

In *JFLA*, pages 139–153.

http://pauillac.inria.fr/jfla/2004/.

Schneider, S. and Treharne, H. (2002).

Communicating B machines.

In [ZB02, 2002], pages 416–435.

Siewe, F. and Hung, D. (2001).

Deriving real-time programs from duration calculus specifications.

In *11th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME 2001)*, volume LNCS 2144, pages 92–97, Livingston-Edinburgh, Scotland. Springer-Verlag.

(Technical Report 222, UNU-IIST, P.O. Box 3058, Macau, December 2000).

Skakkebæk, J. U. (1994).

A Verification Assistant for a Real-Time Logic.

Phd-thesis, Department of Computer Science, Technical University of Denmark.

Also available as Technical Report ID-TR : 1994-150.

Références V

ZB02 (2002).
ZB'2002 – Formal Specification and Development in Z and B, volume 2272 of *Lecture Notes in Computer Science (Springer-Verlag)*, Grenoble, France. LSR-IMAG.