

# Étude de l'apport des méthodes formelles déductives pour les développements de sécurité

Éric Jaeger, UPMC/LIP6/SPI & ANSSI/ACE/LRP

Soutenance de thèse - 8 mars 2010



ANSSI



# Plan

- 1 Pièges
- 2 Raffinement
- 3 Plongement
- 4 Conclusions

# Sécurité et systèmes critiques

Le développement de systèmes est une activité complexe, et les défauts résiduels sont fréquents dans les versions déployées ; mais pour les **systèmes critiques** certains défauts sont inacceptables

Aux plus hauts niveaux d'assurance, la certification de sûreté ou de sécurité recommande ou impose une **validation formelle**

Nous nous intéressons dans la suite à l'utilisation des méthodes formelles (déductives) pour la **sécurité**

- ▶ Quelles différences avec la sûreté de fonctionnement ?
- ▶ Quel est le niveau de confiance ? Quelles sont les limites ?
- ▶ Comment améliorer le niveau d'assurance ?

# Approches formelles

Les approches formelles permettent de mener des **analyses mathématiques** des programmes, et plus généralement des systèmes

- ▶ Le programme est-il cohérent ?
- ▶ Le programme termine-t-il toujours et sans erreurs ?
- ▶ Le programme fait-il ce qu'il doit faire ?

Les principes d'analyse sont justifiés en **correction** et/ou **complétude**

- ▶ Toutes les erreurs (d'un certain type) sont-elles détectées ?
- ▶ Toutes les alertes correspondent-elles à une erreur ?

Les méthodes formelles **déductives** permettent de **prouver la conformité** d'une implémentation à une spécification ; ce sont les plus utilisées en certification de sécurité (B, COQ, FOCALIZE, etc.)

# Démarche de la thèse

Nous considérons le développement formel d'un système sécurisé

- ▶ Identifier et illustrer les **problèmes** qui peuvent se poser
- ▶ Comprendre les **causes** profondes
- ▶ Proposer des **recommandations**

L'analyse considère systématiquement les différentes étapes du cycle de vie et les outils formels utilisés

Pour les illustrations, nous adoptons le point de vue d'un attaquant cherchant à mettre en défaut un objectif de sécurité

- ▶ **Développeur malicieux** voulant faire certifier un système piégé
- ▶ **Utilisateur** voulant contourner la sécurité prouvée

# Malveillance et méthodes formelles

Un attaquant peut tenter d'abuser de différentes situations

- ▶ Spécifications incohérentes ou insuffisantes
- ▶ Mauvaise compréhension de la portée de la preuve (invariant)
- ▶ **Paradoxe du Raffinement**
- ▶ Manipulations de types vides
- ▶ Difficultés à traduire certains concepts de sécurité (*hash*)
- ▶ Violation d'hypothèses explicites ou implicites (typage)
- ▶ **Incohérences dans la théorie**
- ▶ **Erreurs dans les outils formels**

Analyses partiellement applicables en sûreté, mettant en évidence des particularité en **sécurité**; des recommandations pour les développeurs ou évaluateurs sont proposées pour limiter les risques

# Plan

- 1 Pièges
- 2 Raffinement**
- 3 Plongement
- 4 Conclusions

# Paradoxe du raffinement

Certains des pièges identifiés reposent sur le Paradoxe du Raffinement, *i.e.* une **mauvaise compréhension du raffinement** conduisant à ne pas envisager toutes les implémentations possibles

## Exemple (Paradoxe du Raffinement en B)

- ▶ *Combien de raffinements cette spécification admet-elle ?*

**MACHINE** *bool\_choice*

**OPERATIONS**  $b \leftarrow \text{getbool} \triangleq b := \top \parallel b := \perp$

# Paradoxe du raffinement

Certains des pièges identifiés reposent sur le Paradoxe du Raffinement, *i.e.* une **mauvaise compréhension du raffinement** conduisant à ne pas envisager toutes les implémentations possibles

## Exemple (Paradoxe du Raffinement en B)

- ▶ *Combien de raffinements cette spécification admet-elle ?*

**MACHINE** *bool\_choice*

**OPERATIONS**  $b \leftarrow \text{getbool} \triangleq b := \top \parallel b := \perp$

- ▶ *Beaucoup, dont celle-ci*

**MACHINE** *covert\_channel*

**REFINES** *bool\_choice*

**VARIABLES** *secret*

**INVARIANT**  $\text{secret} \in \mathbb{N}$

**OPERATIONS**  $b \leftarrow \text{getbool} \triangleq b := (\text{secret} \% 2 = 0); \text{secret} := \text{secret} / 2$

# Qu'est-ce que le raffinement ?

Nous cherchons à capturer la **démarche de développement** dans une méthode formelle déductive, par exemple B, COQ ou FOCALIZE, et pas seulement à analyser le raffinement de B ou Z

Une relation de **similitude** entre des **descriptions** du système

- ▶ Spécification : abstraite, déclarative, non déterministe
- ▶ Implémentation : concrète, impérative, déterministe

Plusieurs **constituants** sous-jacents identifiés

- ▶ *Data-refinement* (changer la représentation des données)
- ▶ *Choice-refinement* (réduire le non-déterminisme)
- ▶ *Completion-refinement* (étendre le domaine de définition)
- ▶ *Decomposition-refinement* (décomposer en modules)

# Propriétés du raffinement

Quelques propriétés sont naturellement attendues de cette relation

- ▶ Préservation des bonnes propriétés (conformité par similitude) tout en restant **aveugle** à certains aspects des descriptions
- ▶ Réflexivité, transitivité (nombre arbitraire d'étapes)
- ▶ Monotonie (développements modulaires)

Orientations retenues suite à cette analyse

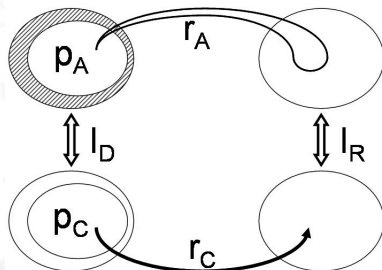
- ▶ Les **descriptions** sont des relations (vision extensionnelle)
- ▶ Les différents **constituants** sont étudiés séparément
- ▶ Les **paramètres** du raffinement sont explicités

Sur cette base, nous proposons une formalisation générique du raffinement (mécanisée en COQ)

# Une vision générique du raffinement

$$(p_A, r_A) \rightsquigarrow (p_C, r_C) [I_D, I_R] \triangleq I_D\{p_A\} \subseteq p_C \wedge p_A \triangleleft (r_C \circ I_D) \subseteq I_R \circ r_A$$

- ▶  $(p_A, r_A)$  et  $(p_C, r_C)$  sont des **descriptions**
- ▶  $I_D$  et  $I_R$  sont des interprétations **paramètres**



# Analyse du raffinement

Cette formalisation permet quelques observations intéressantes

- ▶ Caractérisation des interprétations
- ▶ Impacts de la décomposition et conditions de monotonie
- ▶ Pré-ordre du raffinement vs cycle de développement
- ▶ Présentations alternatives de systèmes standard (ECC)
- ▶ Notion de raffinement partiel (*Retrenchment*)

En particulier concernant l'exploitation du **Paradoxe du Raffinement**

- ▶ Dépendances cachées via des interprétations non fonctionnelles
- ▶ Cela contredit l'**intuition d'une réduction du non-déterminisme**
- ▶ **Non spécifique à B**, reproductible en COQ, FOCALIZE, etc.

# Plan

- 1 Pièges
- 2 Raffinement
- 3 Plongement**
- 4 Conclusions

# Maîtrise de la théorie et des outils formels ?

Des incohérences dans la théorie, des erreurs dans les outils formels, ou certaines options peuvent être exploitées avec des effets désastreux

## Exemple (Typage et preuve en B)

*Si le typage supplémentaire n'est pas activé dans le prouveur, un paradoxe existe*

# Maîtrise de la théorie et des outils formels ?

Des incohérences dans la théorie, des erreurs dans les outils formels, ou certaines options peuvent être exploitées avec des effets désastreux

## Exemple (Typage et preuve en B)

*Si le typage supplémentaire n'est pas activé dans le prouveur, un paradoxe existe*

## Exemple (Traduction de COQ vers OCAML)

- ▶ **let rec**  $\omega = S(\omega)$  est un valeur de  $\mathbb{N}$  en OCAML mais pas en COQ
- ▶ La traduction de **Inductive**  $E : \mathbf{Set} := \mathbf{Cons} : E \rightarrow E$  n'est pas vide
- ▶ **Inductive**  $\mathbf{Empty} := .$  est traduit par **type**  $\mathbf{empty} = \mathbf{unit}$ ;

# Validation de la théorie et des outils

La validation d'une théorie ou d'outils est un problème abstrait et difficile, mais nécessaire : un paradoxe connu du développeur malicieux mais pas de l'évaluateur permet de faire certifier n'importe quoi

Une approche par **plongement profond** permet de

- ▶ Valider la théorie (besoin de précision)
- ▶ Développer des outils certifiés, ici un prouveur
- ▶ Démontrer de nouveaux résultats (bonus)

Nous plongeons la logique du B **telle quelle** en COQ ; B est fréquemment utilisé, notamment en sécurité, et bien documenté

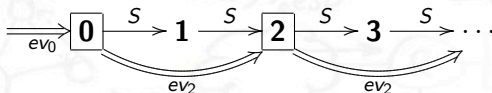
À noter que d'autres plongements de B existent (*Chartier, Bodeveix-Filali-Muñoz, Colin-Mariano, Berkani-Dubois-Faivre-Falampin*) avec des objectifs différents

# Notion de plongement profond

Prenons l'exemple des entiers naturels et du prédicats "être pair"

**Inductive**  $\mathbb{N} : \mathbf{Set} \triangleq 0 : \mathbb{N} \mid S : \mathbb{N} \rightarrow \mathbb{N}$

**Inductive**  $even : \mathbb{N} \rightarrow \mathbf{Prop} \triangleq ev_0 : even\ 0 \mid ev_2 : \forall (n : \mathbb{N}), even\ n \rightarrow even\ S(S\ n)$



$\mathbb{N}$  est une construction **syntaxique** et **even** un prédicat **marquant** des éléments de  $\mathbb{N}$ ; le principe est le même pour un plongement profond

- ▶ Structure syntaxique des propositions d'une logique
- ▶ Prédicat "être prouvable" reproduisant les règles d'inférence et marquant les propositions vraies

# Quelques constats sur le B-BOOK

Le plongement met en évidence quelques omissions ou problèmes

- ▶ Syntaxe non respectée, définitions avec captures accidentelles
- ▶ Besoin de clarification de définitions pourtant formelles
- ▶ Théorèmes du B-BOOK **non prouvables**

$$\not\vdash E_1 \mapsto F_1 = E_2 \mapsto F_2 \Rightarrow E_1 = E_2 \wedge F_1 = F_2$$

$$\not\vdash S_1 \subseteq S_2 \wedge T_1 \subseteq T_2 \Rightarrow S_1 \times T_1 \subseteq S_2 \times T_2$$

Quelques corrections proposées, notamment pour les règles d'inférence

Plusieurs de ces points sont **indéTECTABLES sans un plongement profond**, ce qui justifie l'effort ( $\approx 5$  kLoCoQ)

# Un prouveur prouvé

Le plongement permet de développer des outils certifiés, par exemple ici un **prouveur prouvé** extractible en OCAML

- ▶ Une **tactique** est une fonction sur les séquents (syntaxiques) qui émule l'application d'une règle d'inférence ou d'un théorème  
$$T(\Gamma, P) = [(\Gamma_1, P_1), \dots, (\Gamma_n, P_n)]$$
- ▶ Une tactique par règle d'inférence garantit la **complétude**
- ▶ Une preuve de correction par tactique garantit la **correction**  
$$\mathbf{Fold}_{\wedge}^{\vdash} T(\Gamma, P) \Rightarrow \Gamma \vdash P$$

Prouveur peu utilisable en l'état (peu d'automatisation, pas d'IHM), éventuellement intégrable dans des environnements existants

À noter que le prédicat de prouvabilité définit aussi un **langage des termes de preuve B** (permettant la certification)

# De nouveaux résultats pour le B

Les nouveaux résultats permettent le remplacement de sous-termes équivalents dont les variables libres sont **liées par le contexte**

$$\frac{\Gamma \vdash E_1 \doteq E_2}{\Gamma \vdash [x \triangleleft E_1]E \doteq [x \triangleleft E_2]E} \quad \Gamma \perp_E E_1 \doteq E_2$$

Ces résultats de **congruence** correspondent à de la **normalisation forte** ; ils justifient le dépliage de définitions conditionnelles (division), permettent une meilleure automatisation de la preuve, *etc.*

**Applicabilité** justifiée par l'isomorphisme de *Curry-Howard*

- ▶ Pas d'enrichissement de la logique de B
- ▶ La preuve en Coq du résultat est en fait un programme fabriquant le terme de preuve B correspondant

# Preuve des résultats de congruence

Théorie des substitutions parallèles infinies ( $\approx 1.5$  kLoCoQ en v3)

- ▶ Émulation des opérations, par exemple  $\uparrow_h T = [[M_h^\uparrow]]_h T$
- ▶ Composition  $[[m_1 \odot_h m_2]] T = [[m_1]]([[m_2]] T)$
- ▶ **Quasi-induction** sur  $\mathbb{M} \triangleq \mathbb{I} \rightarrow \mathbb{E}$   
 $P(t) \Rightarrow (\forall m, P([[m]]t) \Rightarrow \forall i e, P([[m \oplus (i \leftarrow e)]]t)) \Rightarrow \forall m, P([[m]]t)$
- ▶ Résultat **sémantique**  $\Gamma \dot{\vdash} p \Rightarrow (\forall i, m \ i = i \vee i \setminus (\Gamma \dot{\vdash} p)) \Rightarrow \Gamma \dot{\vdash} [[m]]p$

Cela rappelle les substitutions explicites. . .

Les résultats de congruence dérivent de la preuve par induction de

$$\Gamma \dot{\vdash} e_1 \dot{=} e_2 \Rightarrow \Gamma \perp_0 e_1 \dot{=} e_2 \Rightarrow$$

$$\forall t \ i \ m, (\forall (n' \ i' : \mathbb{N}), m \ (n'+1, i') = (n+1, i') \vee (n'+1, i') \setminus (\Gamma \dot{\vdash} e_1 \dot{=} e_2)) \Rightarrow$$

**match  $t$  with**

$$| \text{Trm\_of\_Prd } p' \rightarrow \Gamma \dot{\vdash} [[m]]([i \triangleleft e_1]p' \dot{=} [i \triangleleft e_2]p')$$

$$| \text{Trm\_of\_Exp } e' \rightarrow \Gamma \dot{\vdash} [[m]]([i \triangleleft e_1]e' \dot{=} [i \triangleleft e_2]e')$$

# Mécanisation des langages

Analyses des représentations à la *de Bruijn* ( $\approx 25$  kLoCoQ)

- ▶ Représentations fonctionnelles émulant la notation naturelle
- ▶ Induction structurelle et induction sémantique
- ▶ Indices ou niveaux, gestion fine du **contexte** (calcul vs preuve)
- ▶ Adjonction de **sortes** ou de **types** aux identifiants de variables

Synthèse sous la forme d'un  $\lambda$ -calcul simplement typé **sans contexte**

$$\Sigma \triangleq (\Phi) \mid \Sigma \blacktriangleright \Sigma \quad \text{et} \quad \Lambda \triangleq \delta(\Sigma, \mathbb{N}) \mid \underline{\lambda}(\Sigma, \Lambda) \mid \Lambda @ \Lambda$$

$$l \rightarrow_{\beta} l' \Rightarrow [\downarrow l_e]_{\mu}^{\sigma} l \rightarrow_{\beta} [\downarrow l_e]_{\mu}^{\sigma} l' \quad \text{et} \quad l_e \rightarrow_{\beta} l'_e \Rightarrow [\downarrow l_e]_m^{\sigma} l \rightarrow_{\beta}^* [\downarrow l'_e]_m^{\sigma} l'$$

$$l \rightarrow_{\beta} l_1 \Rightarrow l \rightarrow_{\beta} l_2 \Rightarrow \exists l', l_1 \rightarrow_{\beta}^* l' \wedge l_2 \rightarrow_{\beta}^* l' \quad (\text{confluence faible})$$

A noter aussi des discussions sur les représentations de l'application et la substitution vue comme une opération composite

# Plan

- 1 Pièges
- 2 Raffinement
- 3 Plongement
- 4 Conclusions**

# Conclusions “académiques”

L'utilisation des méthodes formelles déductives en sécurité **ne donne pas le même niveau de confiance** qu'en sûreté

- ▶ Il reste possible de certifier des systèmes piégés ou vulnérables
- ▶ Le système ne fait-il **que** ce qu'il doit faire ?
- ▶ Des recommandations pour améliorer la confiance

L'étude du raffinement permet d'expliquer certains phénomènes contre-intuitifs ; elle constitue une présentation alternative, **simple** et **générique** de problématiques complexes notamment en sécurité

La validation de théories formelles est **faisable** et **utile** ; il est possible d'optimiser l'investissement en réutilisant ces travaux pour

- ▶ Développer des outils formels certifiés
- ▶ Démontrer de nouveaux résultats

La thèse propose aussi une analyse de problématiques de mécanisation

# Quelques perspectives

Intégration de différentes approches formelles et analyses (preuves, dépendances, animations, *etc.*) dans un IDE

Poursuite des travaux sur le raffinement, avec l'introduction de la notion de condensation  $r_A \rightsquigarrow r_C [I_D, I_R] \triangleq r_C \circ I_D \subseteq I_R \circ r_A$  et l'étude de la propagation de certaines analyses (dépendance)

Poursuite de la validation du B (cohérence de la logique, raffinement au premier ordre, rôle du *type-checking*, obligations de preuve)

Intégration du plongement de B en COQ avec d'autres outils et environnements

Poursuite des travaux sur la mécanisation des langages (identifiants riches pour les variables, indices et niveaux de *de Bruijn*)

# Conclusions “métier”

Les méthodes formelles (déductives) sont **nécessaires** à la sécurité

- ▶ Élimination d'erreurs courantes et graves (débordements)
- ▶ Validation du bon fonctionnement (conformité)
- ▶ Analyse des politiques de sécurité, des protocoles

Mais elle ne sont pas **suffisantes**

Différents axes d'amélioration identifiés : faciliter la compréhension et l'utilisation, promouvoir les approches combinées, valider les théories et proposer des outils certifiés (ou certifiants)

Les méthodes formelles restent trop peu utilisées dans l'industrie – leur mise en œuvre est surtout justifiée par des exigences de certification ; elles donnent pourtant très tôt des **garanties de correction** et favorisent une **maîtrise réelle** du système développé