Vérification de propriétés de sécurité sur des modèles B

Nicolas Stouls

Nicolas.Stouls@imag.fr

LSR - IMAG / Grenoble

Introduction

- **Motivations :**
  - *ACI Sécurité GECCOO*

    (Génération de code certifié pour des applications orientées objet)
  - *Application au domaine des cartes à puce*
- **Propriétés visées :**
  - *Cycle de vie, Atomicité et contrôle d'accès*
  - *Propriétés comportementales*
- **Etude de cas utilisée :**
  Spécification de l'applet **DEMONEY** [a]

[a] Porte monnaie électronique pour carte à puce.

LSR-IMAG

- Intro.
- Trvx sim.
- Gén. STES
- Prop. sécu.
- Vérif. sécu.
- Concl.

Méthode

1. **Extraction du STE :**
   L'outil GénéSyst *permet d'extraire un automate comportemental (***STE***) d'un modèle* B*.*

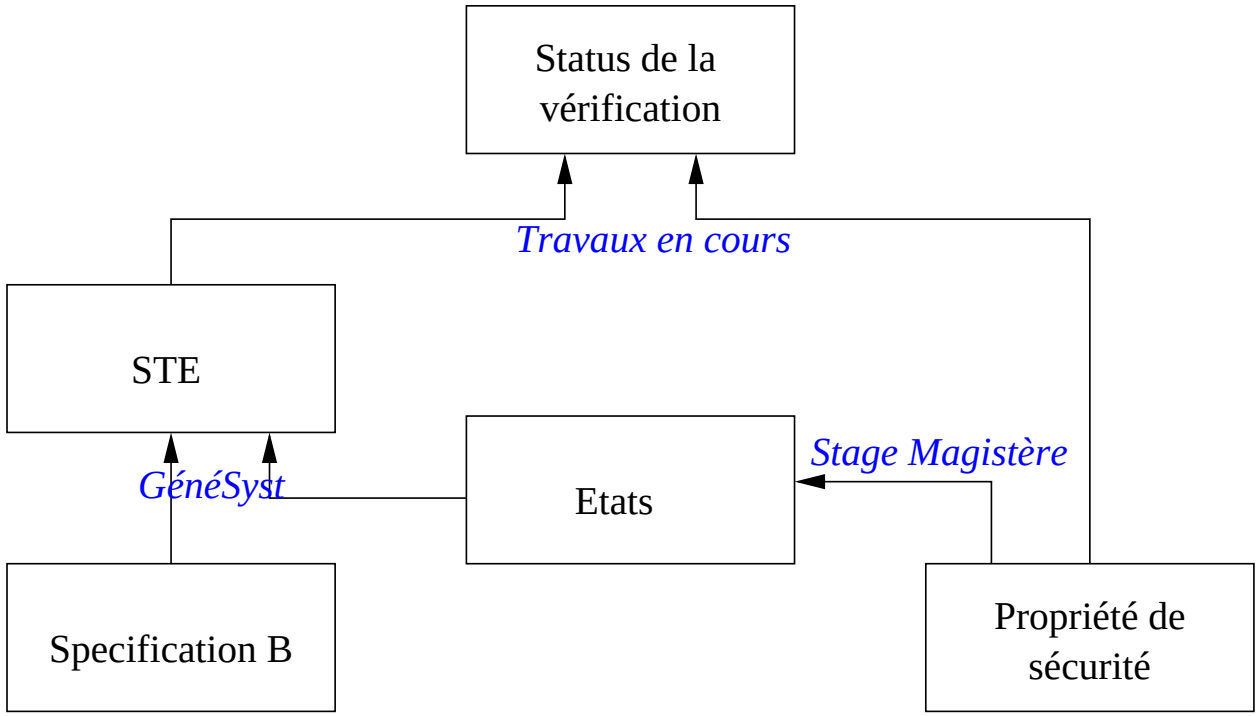
2. **Expression des propriétés :**
   Formule logique avec des prédicats particuliers.

3. **Vérification des propriétés :**
   Syntaxique sur un **STE** *associé.*

LSR-IMAG

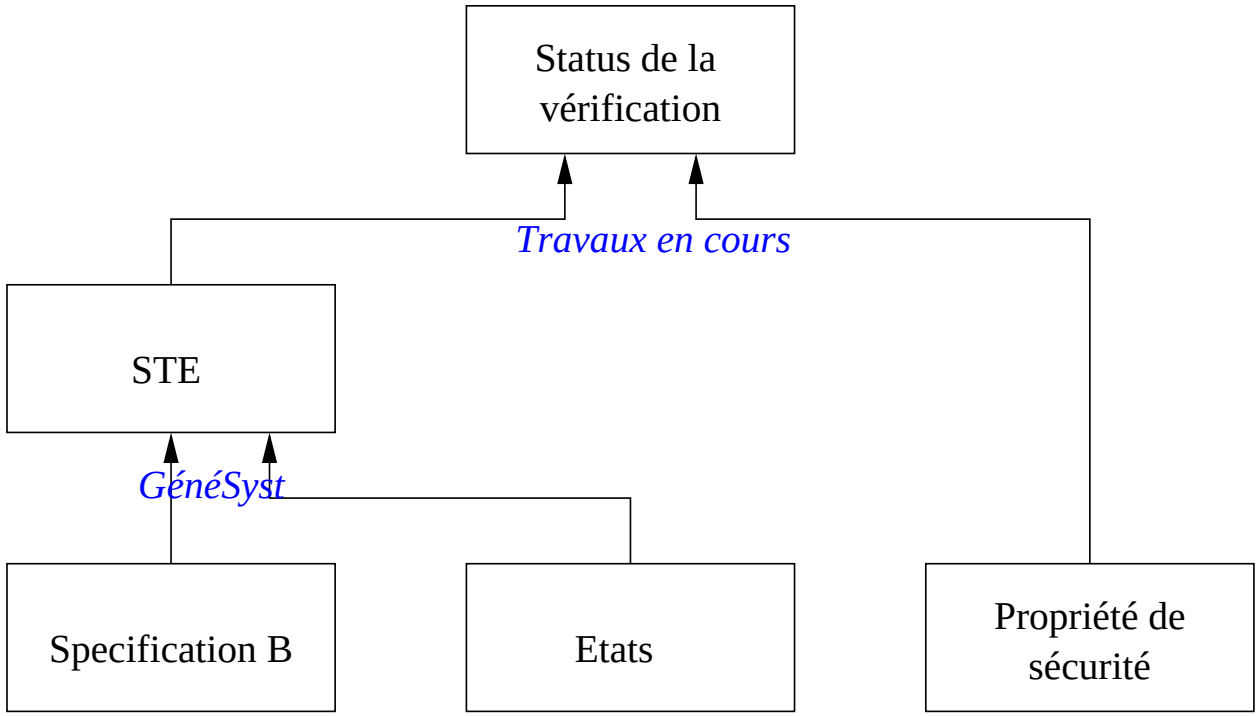
- Intro.
- Trvx sim.
- Gén. STES
- Prop. sécu.
- Vérif. sécu.
- Concl.

Notre Démarche



Première approche

Notre Démarche



Approche simplifiée

Travaux similaires

1. Travaux similaires
2. Génération de systèmes de transitions étiquetées
3. Expression de propriétés de sécurité
4. Vérification de propriétés de sécurité
5. Conclusion

Lamport - Software Engeneering'95

TLA *in Pictures*

- Définition de conditions liant l'interprétation d'un automate à une spécification TLA.
- Les automates définis sont plus abstraits que la spécification.
- Aucun algorithme d'extraction de l'automate n'est décrit.

Rekhis & Boudriga - WITS'04

Formal Verification of Intrusion Detection Systems Using **TLA+**.

- Extension du modèle TLA pour la prise en compte des ensembles.
- Exemples de specification de scénarios mettant en oeuvre des IDS pour modéliser et détecter différent types d'attaques.

Abrial & Mussat - B'98

Introducing Dynamic Constraints in B.

- Introduction de VARIANT, DYNAMICS et MODALITIES.
- Introduction des modalités LEADSTO et UNTIL.
  $\Rightarrow$ Modalités non composables.
  $\Rightarrow$ Obligations de preuves nécessitent une borne finie d'événements déclenchés dans les modalités.

Exemple de limitation :

INITIALISATION $x := 1 || V := 100$

OPERATIONS

$ev_1 \ \hat{=} \ x > 0 \Longrightarrow x := x + 1 \ || \ V := V - 1$

$ev_2 \ \hat{=} \ x = 2 \Longrightarrow x := -1 \ || \ V := 0$

MODALITIES

SELECT $x = 0$ UNTIL $x = -1$ WHILE ev_1 OR ev_2 VARIANT V END

Bellegarde, Chouali & Julliand - ZB'02

Verification of Dynamic Constraints for B *Event Systems under Fairness Assumptions.*

- Introduction d'une logique **PLTL**.
- Hypothèses d'équités forte (ou faible par implication).
- Vérification par Model-Checking.

Ruíz Barradas & Bert - IFM'02

Specification and Proof of Liveness Properties under Fairness Assumptions in B *Event Systems.*

- Utilisation de la logique **UNITY**.
- Hypothèses d'équités faible ou de progrès minimum.
- Vérification par preuve.

Génération de systèmes de transitions étiquetées

1. Travaux similaires
2. Génération de systèmes de transitions étiquetées
3. Expression de propriétés de sécurité
4. Vérification de propriétés de sécurité
5. Conclusion

Exemple de machine B

MACHINE $Demoney$

SETS $TransacType = \{Credit, Debit, None\}; StatusType = \{ISO\_Error, ISO\_Ok\}$

VARIABLES $StatusWord, CurTransaction, ChannelIsSecured, Personalized$

INVARIANT

$StatusWord \in StatusType \ \wedge \ CurTransaction \in TransacType \ \wedge$
$ChannelIsSecured \in \mathsf{BOOL} \ \wedge \ Personalized \in \mathsf{BOOL} \ \wedge$
$((StatusWord \neq ISO\_Ok) \Rightarrow CurTransaction = None) \ \wedge \ \ldots$

INITIALISATION

$StatusWord := ISO\_Ok \ || \ ChannelIsSecured := \mathsf{false} \ || \ \ldots$

OPERATIONS

$StoreData$ = IF $CurTransaction = None$
THEN $Personalized :\in \mathsf{BOOL} \ || \ StatusWord :\in StatusType$
ELSE $StatusWord := ISO\_Error \ || \ CurTransaction := None$
END;

$GetData$ = ...

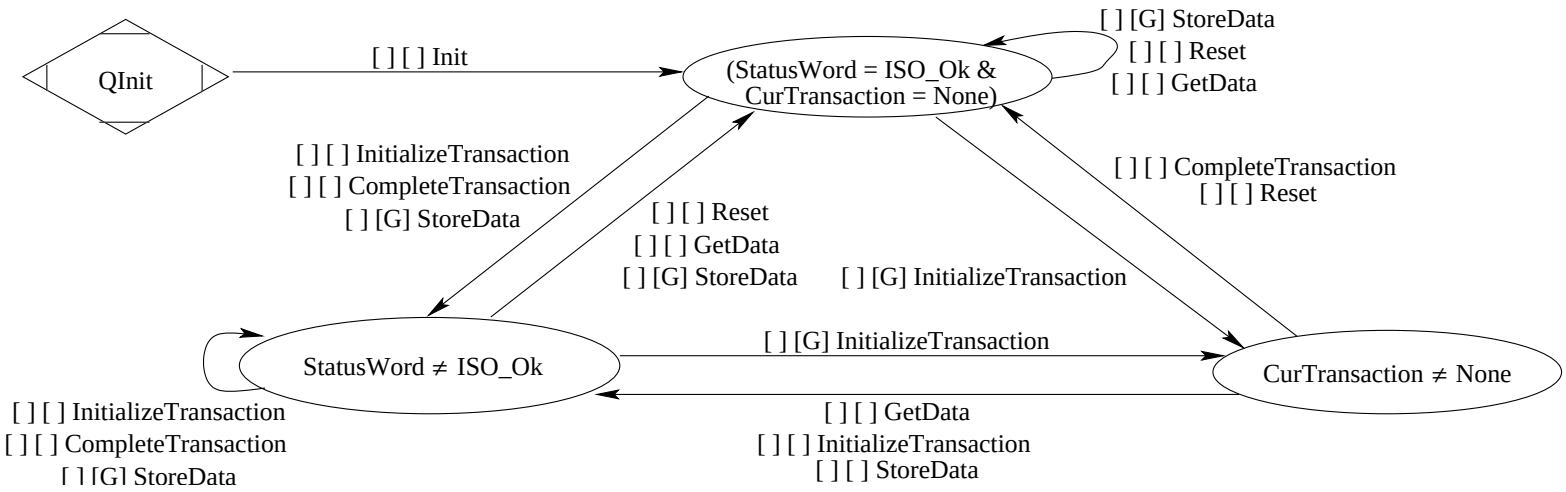
$Reset$ = ...

$InitializeTransaction$ = ...

$CompleteTransaction$ = ...

END

Principe de GénéSyst



Exemple tiré de **DEMONEY**

- *Etats fournis par l'utilisateur*

  Etats = Prédicat portant sur les variables du système

$$I \Rightarrow K$$

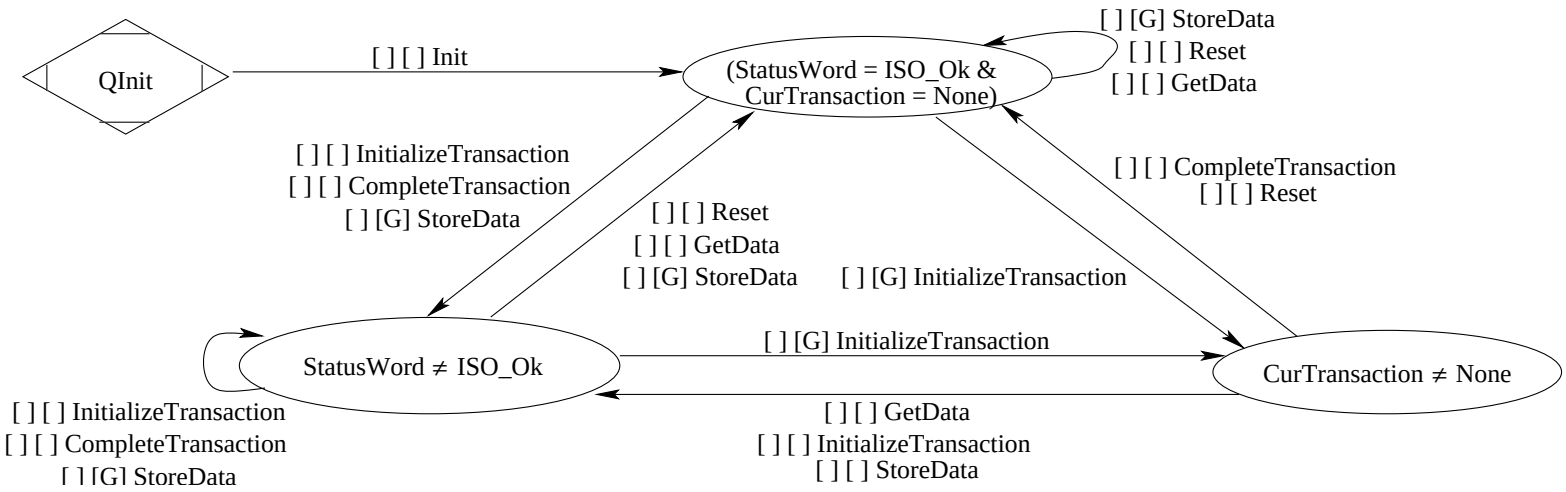
Avec I l'invariant et K la disjonction des états.

- *Invariant du système déjà prouvé*

$$I \Rightarrow [ev]I$$

Avec I l'invariant et ev un événement.

Exemple de STE



Exemple tiré de **DEMONEY**

- Transitions gardées : [ ][ ]
- Décomposition de la garde en 2 conditions :
  *Déclenchabilité (**D**) & Atteignabilité (**A**)*

Calcul de D et A

- Pour les conditions D et A, on s'intéresse à 3 valeurs particulières :

  $true$, $false$ et $conditionnee$

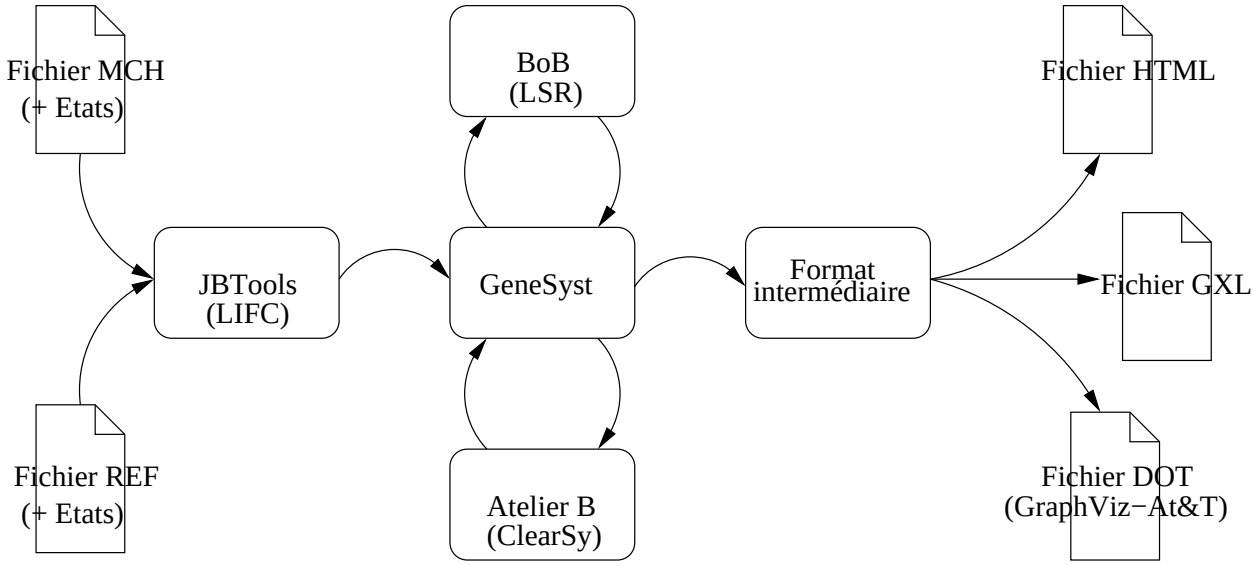
- Deux cas :
  - Preuve automatique :

    Le système est **minimal** *si tous les* D, A *vallent* $true$ *ou* $false$.

  - Preuve interactive :

    Le système est **minimal**.

GénéSyst



- Diffusé sous la licence *CeCILL* (GNU français)
- [PS04] ML. Potet et N. Stouls, *Explicitation du contrôle de développement B événementiel*, AFADL'04.
- Disponible en téléchargement à l'adresse :

http://www-lsr.imag.fr/Les.Personnes/Nicolas.Stouls/

Expression de propriétés de sécurité

1. Travaux similaires
2. Génération de systèmes de transitions étiquetées
3. Expression de propriétés de sécurité
4. Vérification de propriétés de sécurité
5. Conclusion

Prédicats de base

Si p_1 et p_2 sont 2 prédicats et ev un événement du système $\mathcal{S}$ alors :

ev peut être déclenché depuis p_1 :

$$Enabled(p_1, ev) \quad \hat{=} \quad \exists x \cdot (I \ \wedge \ p_1 \ \wedge \ Garde(ev))$$

ev est toujours déclenchable depuis p_1 :

$$AlwaysEnabled(p_1, ev) \quad \hat{=} \quad \forall x \cdot (I \ \wedge \ p_1 \Rightarrow Garde(ev))$$

ev peut aller en p_2 depuis p_1 :

$$Crossable(p_1, ev, p_2) \quad \hat{=} \quad \exists x \cdot (I \ \wedge \ p_1 \ \wedge \ \neg[ev]\neg p_2)$$

ev mène toujours en p_2 depuis p_1 :

$$AlwaysCrossable(p_1, ev, p_2) \quad \hat{=} \quad \forall x \cdot (I \ \wedge \ p_1 \Rightarrow [ev]p_2)$$

avec x les variables de $\mathcal{S}$ et I son invariant.

Exemple de propriétés simples

- Système défensif :

$$\forall_{ev} e \cdot (e \in interface(\mathcal{S}) \Rightarrow AlwaysEnabled(true, e))$$

- Unicité de la personnalisation :

$$\forall_{ev} e \cdot (e \in interface(\mathcal{S}) \Rightarrow \neg Crossable(Personalized = true, e, Personalized = false))$$

- Existence de la personnalisation :

$$Crossable(Personalized = false, StoreData, Personalized = true)$$

Vérification de propriétés de sécurité

1. Travaux similaires
2. Génération de systèmes de transitions étiquetées
3. Expression de propriétés de sécurité
4. Vérification de propriétés de sécurité
5. Conclusion

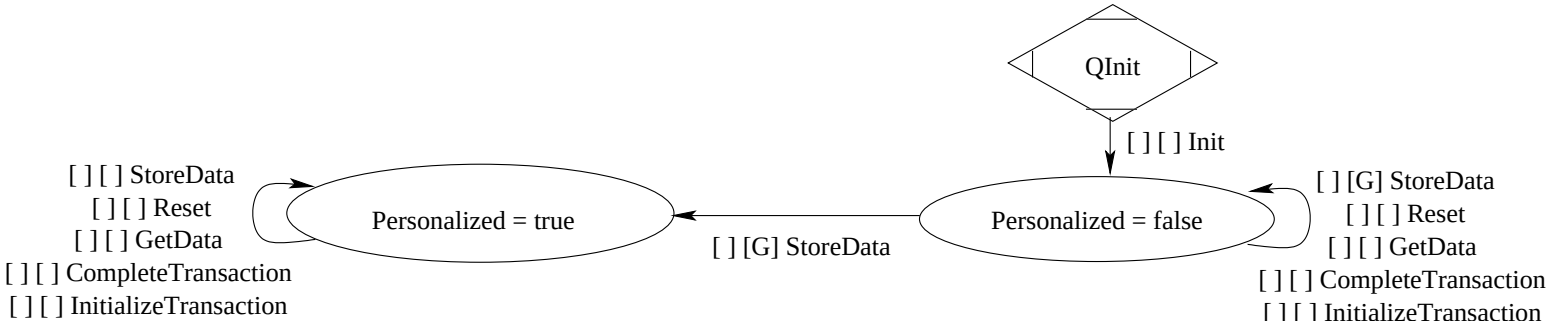
Principes de la vérification

- Vérification syntaxique des prédicats proposés.
- Règles de vérification *suffi santes*.
- Deux cas sont à différencier :
  - STE minimal,
  - STE non minimal.

Règles de vérification

Prédicat	STE minimal	STE non minimal
$Enabled(q_1, e)$	$D \not\equiv false$	$D \equiv true$
$\neg Enabled(q_1, e)$	$D \equiv false$	$D \equiv false$
$AlwaysEnabled(q_1, e)$	$D \equiv true$	$D \equiv true$
$\neg AlwaysEnabled(q_1, e)$	$D \not\equiv true$	$D \equiv false$
$Crossable(q_1, e, q_2)$	$A \not\equiv false$	$A \equiv true$
$\neg Crossable(q_1, e, q_2)$	$A \equiv false$	$A \equiv false$
$AlwaysCrossable(q_1, e, q_2)$	$A \equiv true \ \wedge$ $\forall q_3 \cdot (q_3 \in Q_{\mathcal{S}} \wedge$ $q_3 \not\equiv q_2 \Rightarrow$ $(q_1, e, q_3) \notin W_{\mathcal{S}})$	$A \equiv true \ \wedge$ $\forall q_3 \cdot (q_3 \in Q_{\mathcal{S}} \wedge$ $q_3 \not\equiv q_2 \Rightarrow$ $(q_1, e, q_3) \notin W_{\mathcal{S}})$
$\neg AlwaysCrossable(q_1, e, q_2)$	$A \not\equiv true$	$A \equiv false$

Exemples de vérification



Système défensif :

$$\forall ev \cdot (ev \in interface(\mathcal{S}) \Rightarrow AlwaysEnabled(true, ev)) \Leftrightarrow$$

$$AlwaysEnabled(true, StoreData) \ \wedge$$

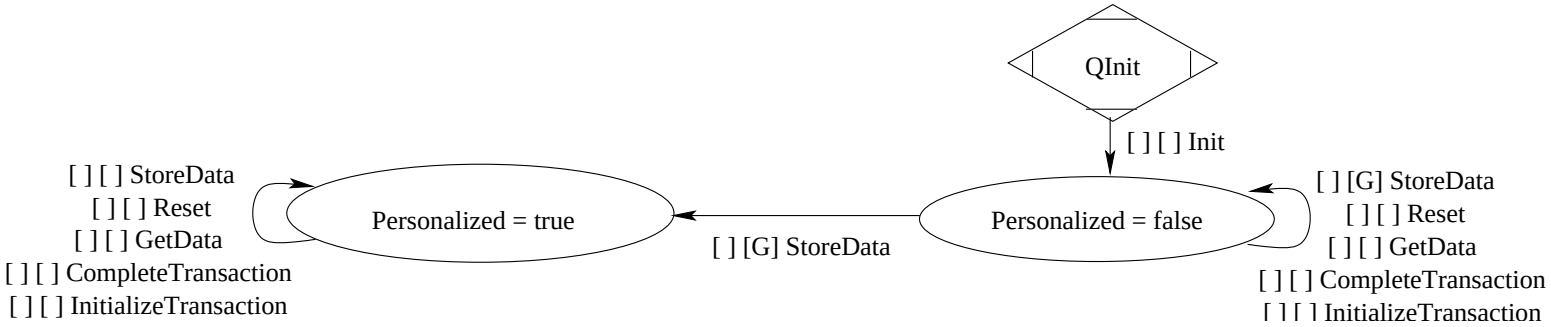
$$AlwaysEnabled(true, GetData) \ \wedge$$

$$AlwaysEnabled(true, Reset) \ \wedge$$

$$AlwaysEnabled(true, InitializeTransaction) \ \wedge$$

$$AlwaysEnabled(true, CompleteTransaction)$$

Exemples de vérification



Unicité de la personnalisation :

$$\forall ev \cdot (ev \in interface(\mathcal{S}) \Rightarrow$$

$$\neg Crossable(Personalized = true, ev, Personalized = false)) \Leftrightarrow$$

$$\neg Crossable(Personalized = true, StoreData, Personalized = false)) \ \wedge$$

$$\neg Crossable(Personalized = true, GetData, Personalized = false)) \ \wedge$$

$$\neg Crossable(Personalized = true, Reset, Personalized = false)) \ \wedge$$

$$\neg Crossable(Personalized = true, InitializeTransaction, Personalized = false)) \ \wedge$$

$$\neg Crossable(Personalized = true, CompleteTransaction, Personalized = false))$$

Conclusion

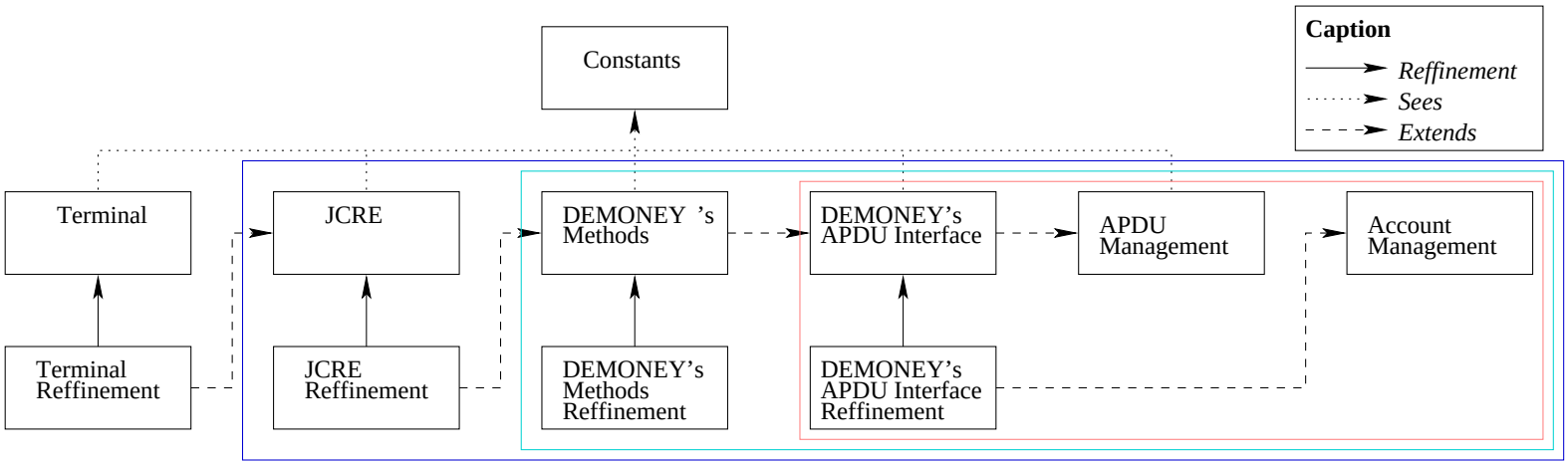
1. Travaux similaires
2. Génération de systèmes de transitions étiquetées
3. Expression de propriétés de sécurité
4. Vérification de propriétés de sécurité
5. Conclusion

Bilan des travaux réalisés

- GénéSyst : Extraction du contrôle d'une spécification B.
- Introduction de prédicats pour exprimer des propriétés de sécurité.
- Méthode de vérification de ces prédicats.
- Application au cas d'étude **DEMONEY**.
- Propriétés ramenées à des propriétés de sûreté.

Perspectives

- Prédicats plus évolués ($next$, $followed$, $until$, $leadsto$, . . .).
- Contrôle d'accès (Droits positifs ou négatifs).
- Modèles modulaires.
- Modèles à base d'opérations (Problème des paramètres).
- Prise en compte des raffi nements.



LSR-IMAG

- Intro.
- Trvx sim.
- Gén. STES
- Prop. sécu.
- Vérif. sécu.
- Concl.

Avez vous des questions ?