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

Vérification de propriétés de sécurité sur des modèles B

Nicolas Stouls [a]

Nicolas.Stouls@imag.fr

LSR - IMAG / Grenoble

[a] Travail supporté par une bourse BDI financée par ST-Microelectronics et le CNRS.

Travail réalisé dans le cadre de l'ACI sécurité GECCOO

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.

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.*

Génération de systèmes de transitions étiquetées

1. Génération de systèmes de transitions étiquetées
2. Expression de propriétés de sécurité
3. Vérification de propriétés de sécurité
4. 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;

$GetData$ = ...

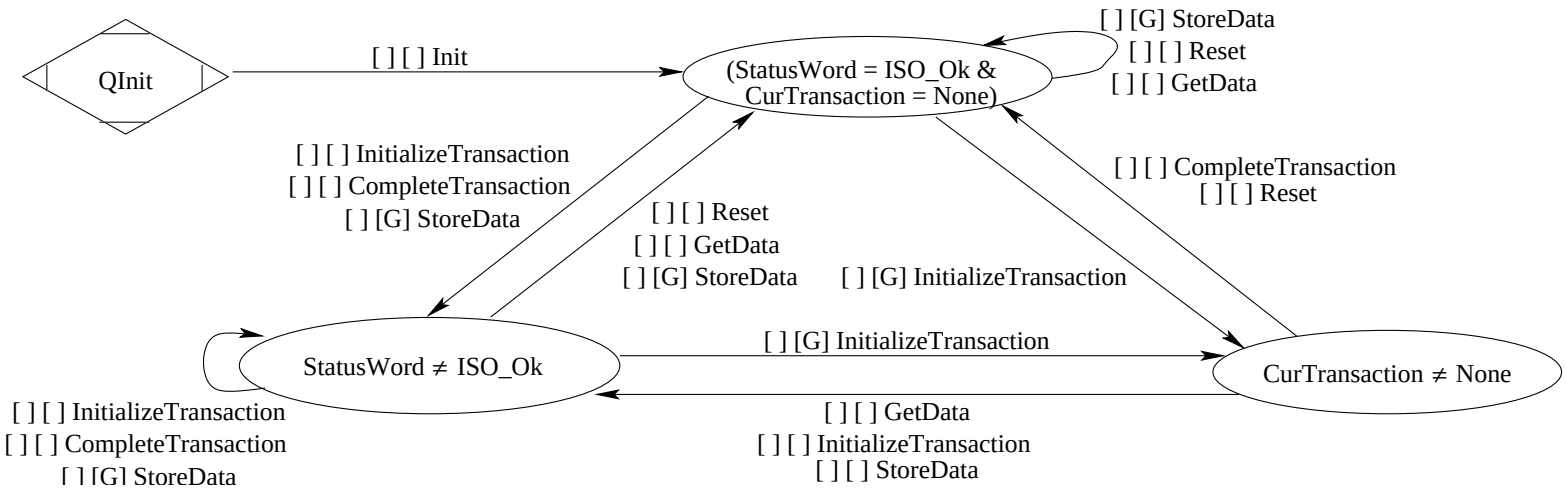
$Reset$ = ...

$InitializeTransaction$ = ...

$CompleteTransaction$ = ...

END

Principe de GénéSyst



Exemple tiré de **DEMONEY**

- *Etats fournis par l'utilisateur*

  Etats = Prédicat portant sur les variables du système

$$I \Rightarrow K$$

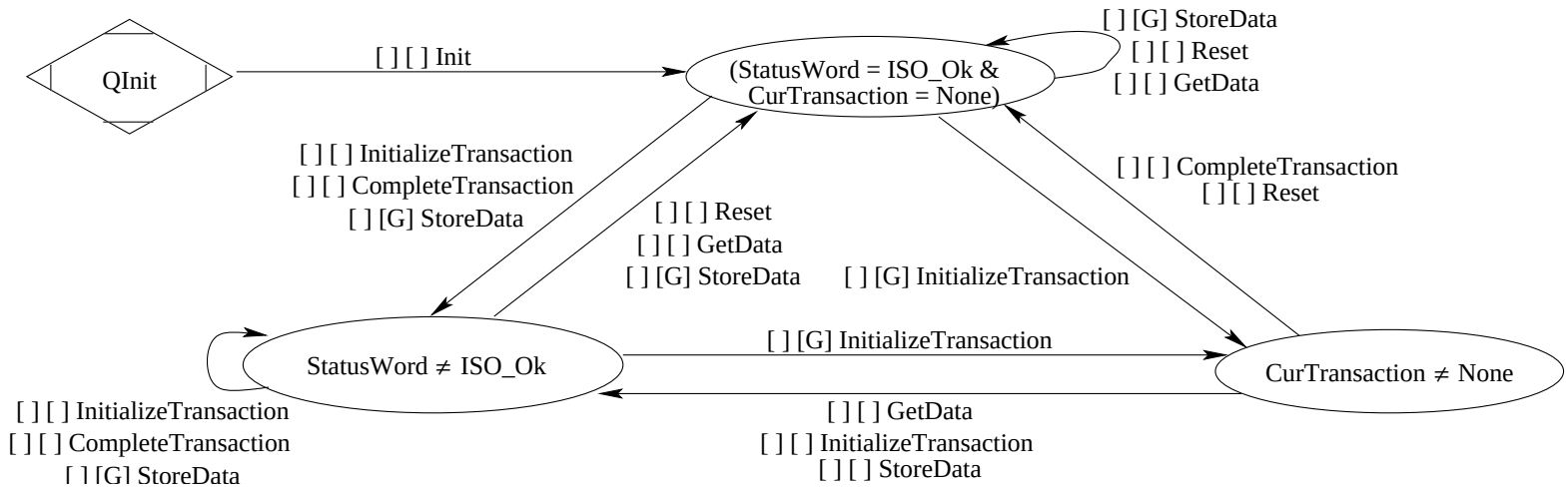
Avec I l'invariant et K la disjonction des états.

- *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 :

$$true,\ false \text{ et } conditionnee$$

- Calcul des conditions :

	Obligations de preuve	D
(1)	$\forall x \cdot (E \Rightarrow Garde(e))$	$true$
(2)	$\forall x \cdot (E \Rightarrow \neg Garde(e))$	$false$
(3)	$\exists x \cdot (E \wedge Garde(e))$	$Garde(e)$
	Obligations de preuve	A
(4)	$\forall x \cdot (E \wedge Garde(e) \Rightarrow \neg[Action(e)]\neg F)$	$true$
(5)	$\forall x \cdot (E \wedge Garde(e) \Rightarrow [Action(e)]\neg F)$	$false$
(6)	$\exists x \cdot (E \wedge Garde(e) \wedge \neg[Action(e)]\neg F)$	$\neg[Action(e)]\neg F$

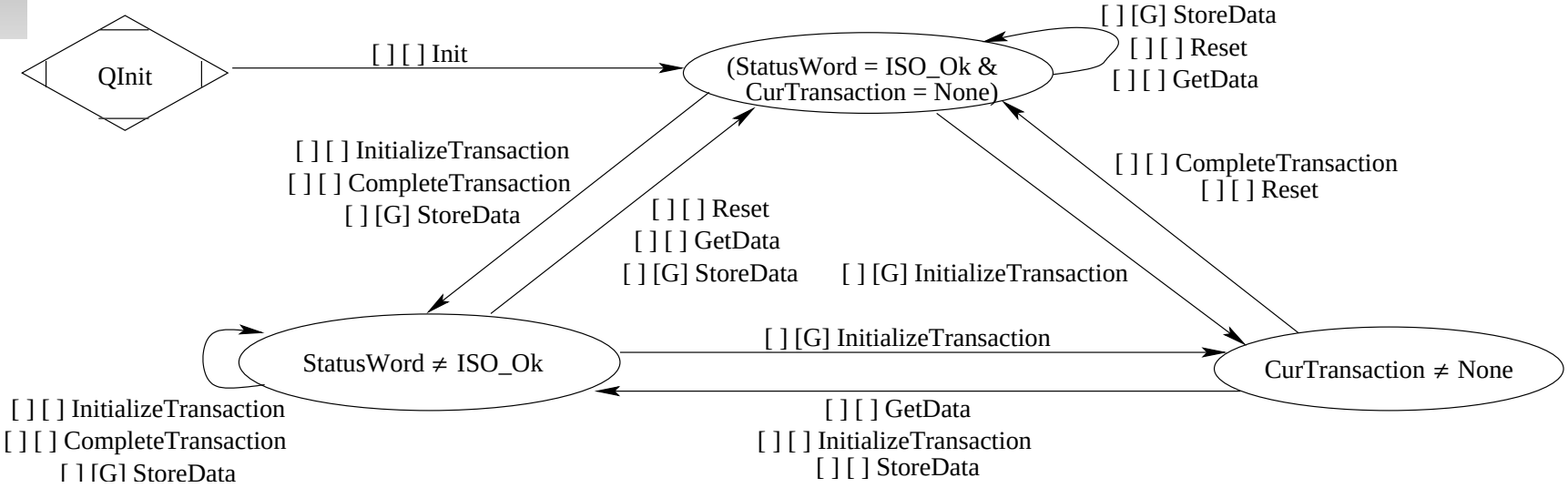
Résolution des obligations de preuve

Deux cas :

- Preuve automatique :
  - *Les OP (3) et (6) ne sont pas prouvées.*
  - *Le système est minimal si tous les* D, A *vallent* $true$ *ou* $false$*.*
- Preuve interactive :
  - *Les OP (3) et (6) n'ont pas a être prouvées.*
  - *Le système est minimal.*

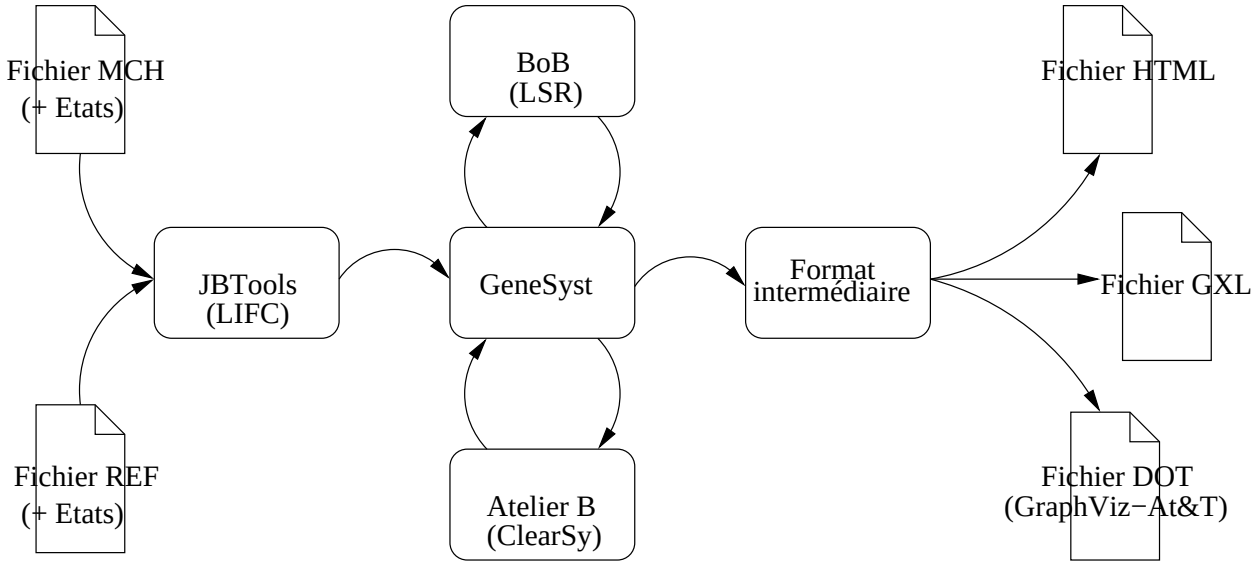
Exemple de STE (Rappel)

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



Exemple issu de **DEMONEY**

(Système minimal car conditions prouvées interactivement)



- 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. Génération de systèmes de transitions étiquetées
2. Expression de propriétés de sécurité
3. Vérification de propriétés de sécurité
4. Conclusion

Principes de l'approche

- Expression de propriétés de type :
  - *Cycle de vie ;*
  - *Atomicité ;*
  - *Contrôle d'accès.*
- Définition de prédicats sur transitions simples :
  - $Enabled$;
  - $AlwaysEnabled$;
  - $Crossable$;
  - $AlwaysCrossable$.

Logique utilisée

Logique d'expression des propriétés :

- $F ::= \forall_{ev} E \cdot (Fq \Rightarrow F) \mid \exists_{ev} E \cdot (Fq \ \wedge \ F) \mid Atome$
- $Fq ::= Fq \ \wedge \ Fq \mid E = E \mid E \neq E \mid E \in EnsEv$
- $E ::=$ un nom d'événement
- $EnsEv ::= Interface(\mathcal{S}) \mid E_1, ..., E_n$
- $Atome ::= \neg Atome \mid Enabled(PredB, E) \mid AlwaysEnabled(PredB, E) \mid Crossable(PredB, E, PredB) \mid AlwaysCrossable(PredB, E, PredB)$
- $PredB ::=$ prédicats tels que défi nis en B

Avec prédicats B basés sur les variables du modèle $\mathcal{S}$.

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.

Quantificateurs et égalité d'événements

Si ev est un événement du système $\mathcal{S}$ alors :

$\forall_{ev}$ porte sur l'ensemble des événements e_i de $\mathcal{S}$:

$$\forall_{ev} e \cdot (e \in Interface(\mathcal{S}) \Rightarrow P(e)) \quad \hat{=} \quad P(e_1) \ \wedge \ P(e_2) \ \wedge \ \ldots P(e_n)$$

$\exists_{ev}$ porte sur l'ensemble des événements e_i de $\mathcal{S}$:

$$\exists_{ev} e \cdot (e \in Interface(\mathcal{S}) \wedge P(e)) \quad \hat{=} \quad P(e_1) \ \vee \ P(e_2) \ \vee \ \ldots P(e_n)$$

La comparaison des événements se fait sur leur nom :

$$e_1 = e_2 \quad \hat{=} \quad \text{nom}(e_1) \equiv \text{nom}(e_2)$$

$$e_1 \neq e_2 \quad \hat{=} \quad \text{nom}(e_1) \not\equiv \text{nom}(e_2)$$

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. Génération de systèmes de transitions étiquetées
2. Expression de propriétés de sécurité
3. Vérification de propriétés de sécurité
4. Conclusion

Principes de la vérification

- Vérification syntaxique des prédicats proposés.
- Règles de vérification *suffisantes*.
- Prédicats des propriétés $\neq$ Etats du système
- Implications pour conversion :
  Etat du système / Prédicat de la propriété.
- Deux cas sont à différencier :
  - STE minimal,
  - STE non minimal.

Règles d'implication

Soient p_1, p_2 et p_3 3 prédicats et ev un événement, alors :

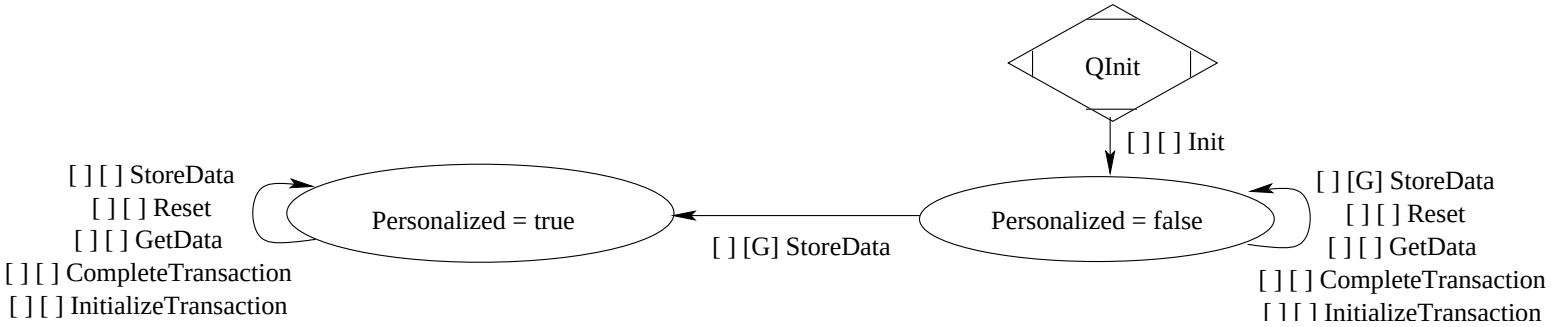
1. Si $p_1 \Rightarrow p_3$ et $Enabled(p_1, ev)$ alors $Enabled(p_3, ev)$
2. Si $p_3 \Rightarrow p_1$ et $AlwaysEnabled(p_1, ev)$ alors $AlwaysEnabled(p_3, ev)$
3. Si $p_1 \Rightarrow p_3$ et $Crossable(p_1, ev, p_2)$ alors $Crossable(p_3, ev, p_2)$
4. Si $p_2 \Rightarrow p_3$ et $Crossable(p_1, ev, p_2)$ alors $Crossable(p_1, ev, p_3)$
5. Si $p_3 \Rightarrow p_1$ et $AlwaysCrossable(p_1, ev, p_2)$ alors $AlwaysCrossable(p_3, ev, p_2)$
6. Si $p_2 \Rightarrow p_3$ et $AlwaysCrossable(p_1, ev, p_2)$ alors $AlwaysCrossable(p_1, ev, p_3)$

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

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



Système défensif :

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

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

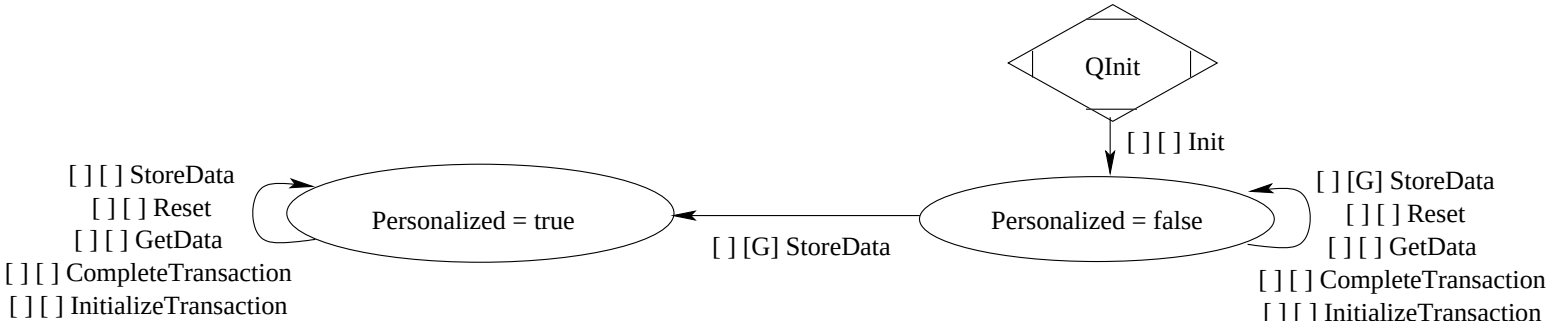
$$AlwaysEnabled(true, GetData) \ \wedge$$

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

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

$$AlwaysEnabled(true, CompleteTransaction)$$

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. Génération de systèmes de transitions étiquetées
2. Expression de propriétés de sécurité
3. Vérification de propriétés de sécurité
4. Conclusion

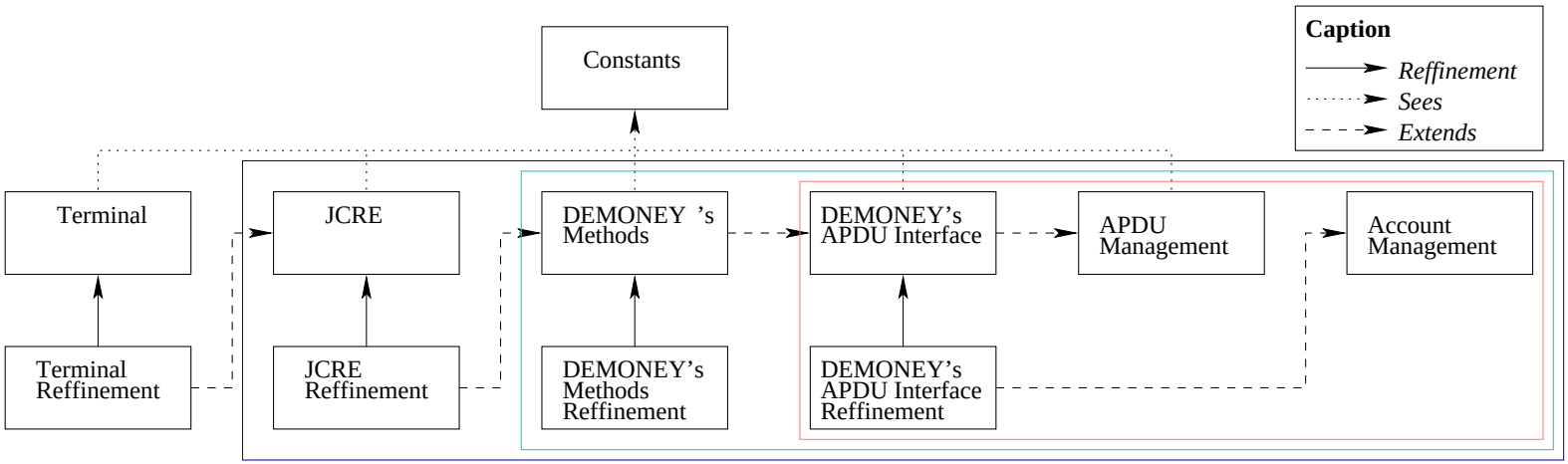
LSR-IMAG

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.



Avez vous des questions ?

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