Travaux sur DEMONEY

Nicolas Stouls [a] *Marie-Laure Potet*

nicolas.stouls@imag.fr marie-laure.potet@imag.fr

LSR-IMAG - Grenoble

---

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

Introduction

1. Introduction
2. Construction d'un système de transitions étiqueté
3. Etude de cas **DEMONEY**
4. Conclusion

Motivations

- Notre approche

  Utilisation de la méthode B *pour exprimer et raffiner les spécifications jusqu'à leur implantation.*

  Vérification de propriété à partir d'automates générés par l'outil GénéSyst*.*

- Type de propriétés visées

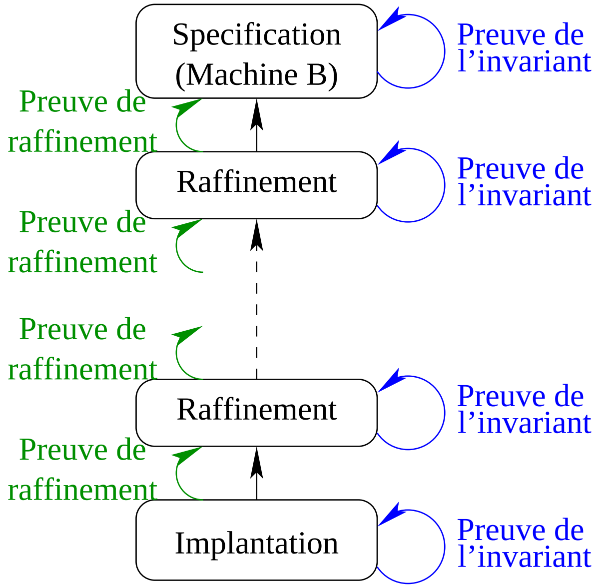
  Comportementales (cycle de vie, contrôle d'accès, atomicité, . . .)

- Pourquoi **DEMONEY**

  Etude de cas commune à GECCOO permettant de comparer les différentes méthodes.

La méthode B

- Langage basé sur l'affectation
- Invariant
- Modularité
- Développement par raffinement
- Génération de code *(C/C++/ADA)*



Ex : *Métro sans conducteur Météor :*

86000 lignes d'ADA produites à partir de B

env. 1000 composants B

27800 OP.

Autre exemples en cours : *Métro de New York, Navette de Roissy*

Exemple de machine B

MACHINE $Demoney$

SETS $StatusType = \{ISO\_Error, ISO\_Ok\}$

VARIABLES $StatusWord, Personalized$

INVARIANT

$StatusWord \in StatusType \ \wedge \ Personalized \in \mathsf{BOOL}$

INITIALISATION

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

OPERATIONS

$StoreData$ = CHOICE $StatusWord := ISO\_Ok \ || \ Personalized := \mathsf{true}$
OR $StatusWord := ISO\_Error$
END;

$GetData$ = $StatusWord := ISO\_Ok$;

$InitializeTransaction$ = IF $Personalized = \mathsf{false}$
THEN $StatusWord := ISO\_Error$
ELSE $StatusWord :\in StatusType$
END;

$CompleteTransaction$ = $StatusWord :\in StatusType$

END

Construction d'un système de transitions étiqueté

1. Introduction
2. Construction d'un système de transitions étiqueté
3. Etude de cas **DEMONEY**
4. Conclusion

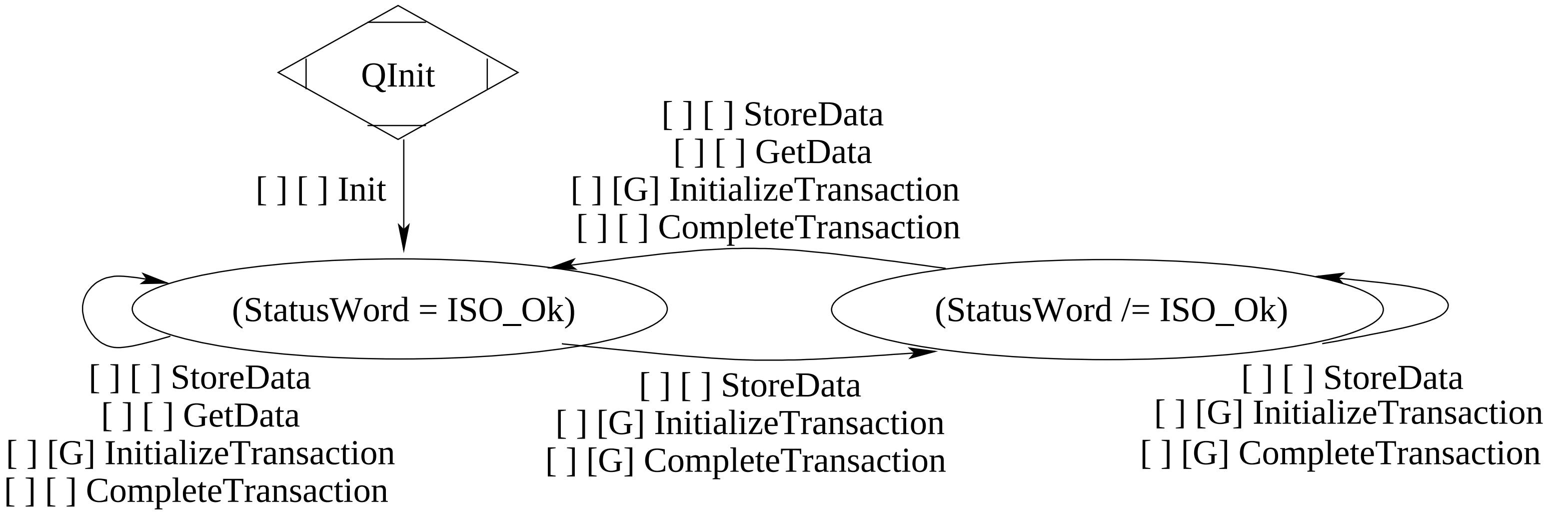
- Visualisation de l'aspect comportemental selon différents points de vue

- Représentation d'automates possiblement infinis

- Exhibation de comportements en lien avec des propriétés de sécurité
  aspects comportementaux (Protocole, droits d'accès, . . .)

- Aide au raffinement
  génération automatique d'invariant

Principe de GénéSyst



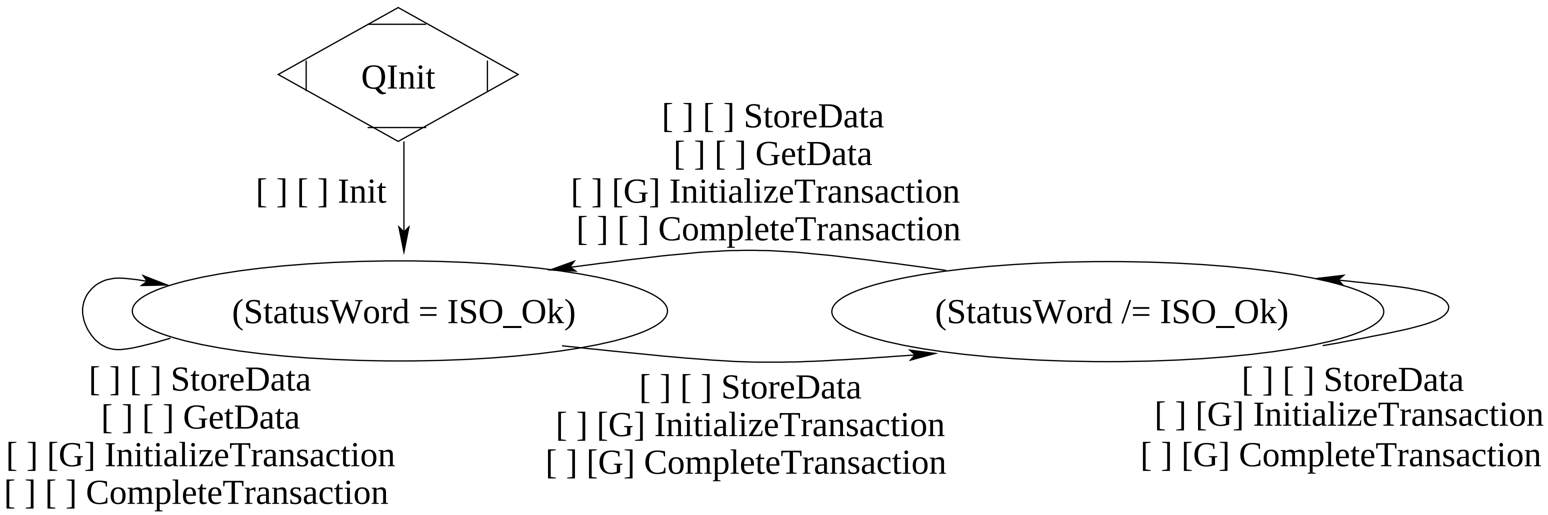
Exemple tiré de **DEMONEY**

- Etats fournis par l'utilisateur

  Etat = Prédicat portant sur les variables du système

$$I \Rightarrow E_0 \;\vee\; E_1 \;\vee\; \ldots \;\vee\; E_n$$

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 :

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

- Définition de D et A :

$$I \ \wedge \ E \Rightarrow (D \Leftrightarrow Garde(Ev))$$

$$I \ \wedge \ E \ \wedge \ Garde(Ev) \Rightarrow (A \Leftrightarrow \neg[Ev]\neg F))$$

- Algorithme :

D vaut true ?	A vaut true ?
Sinon D vaut false ?	Sinon A vaut false ?
Sinon D est $conditionnée$.	Sinon A est $conditionnée$.

$$E \xrightarrow{[D][A]Ev} F$$

Exemple de raffinement B

REFINEMENT $Demoney\_R1$

REFINES $Demoney$

SETS $TransacType = \{Credit, Debit, None\}$

VARIABLES $StatusWord, CurTransaction, ChannelIsSecured, Personalized$

INVARIANT

$CurTransaction \in TransacType \ \wedge \ ChannelIsSecured \in \mathsf{BOOL} \ \wedge$

$((StatusWord \neq ISO\_Ok) \Rightarrow CurTransaction = None) \ \wedge \ \ldots$

INITIALISATION

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

OPERATIONS

$StoreData$ = IF $CurTransaction = None \ \wedge \ ChannelIsSecured = \mathsf{true}$
$\wedge \ Personalized = \mathsf{false}$
THEN $Personalized := \mathsf{true} \ || \ StatusWord := ISO\_Ok$
ELSE $StatusWord := ISO\_Error \ || \ CurTransaction := None$
END;

$GetData$ = ...

$InitializeTransaction$ = ...

$CompleteTransaction$ = ...

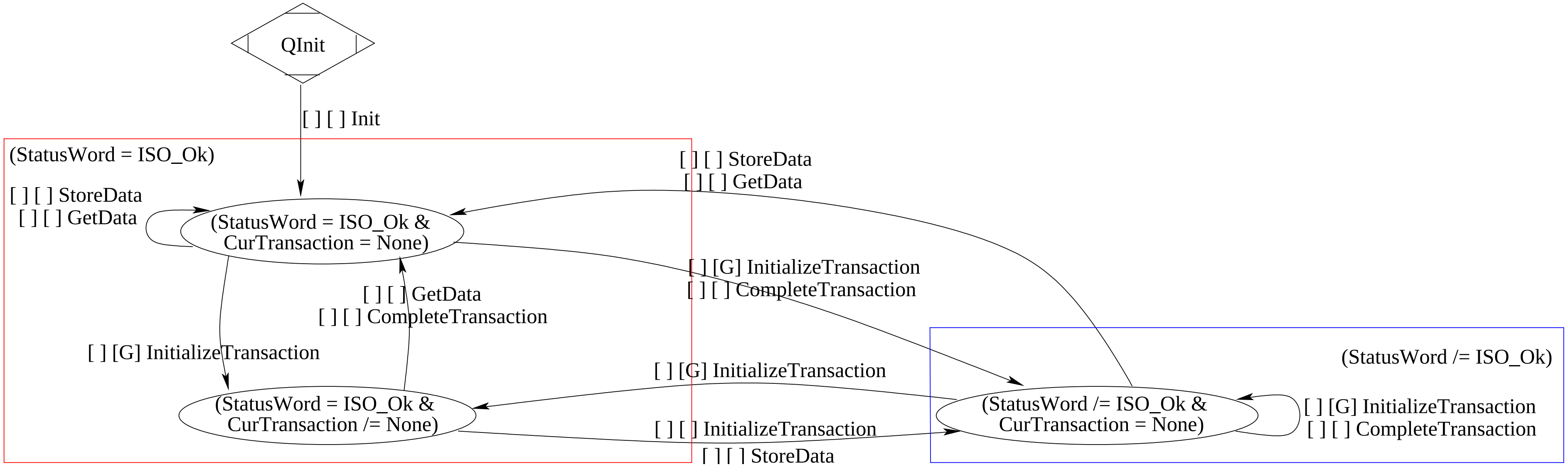
END

Représenter le raffinement

Construction pour le raffinement en conservant la structure de la spécification :

- Utiliser une représentation hiérarchisée,
- Affiner les conditions sur des transitions

Raffinement des données, du comportement et de la dynamique du système



Remarque : *Les variables de l'abstraction ne sont pas forcément conservées dans le raffinement.*

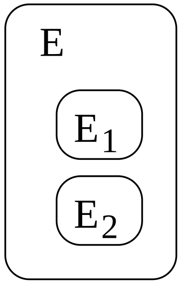
Représentation hiérarchique

- On associe un ensemble d'états $\{E_1, \ldots, E_n\}$ à chaque état E tel que :

$$\bigvee_{i=1}^{n} E_i \Leftrightarrow Proj_L(E)$$

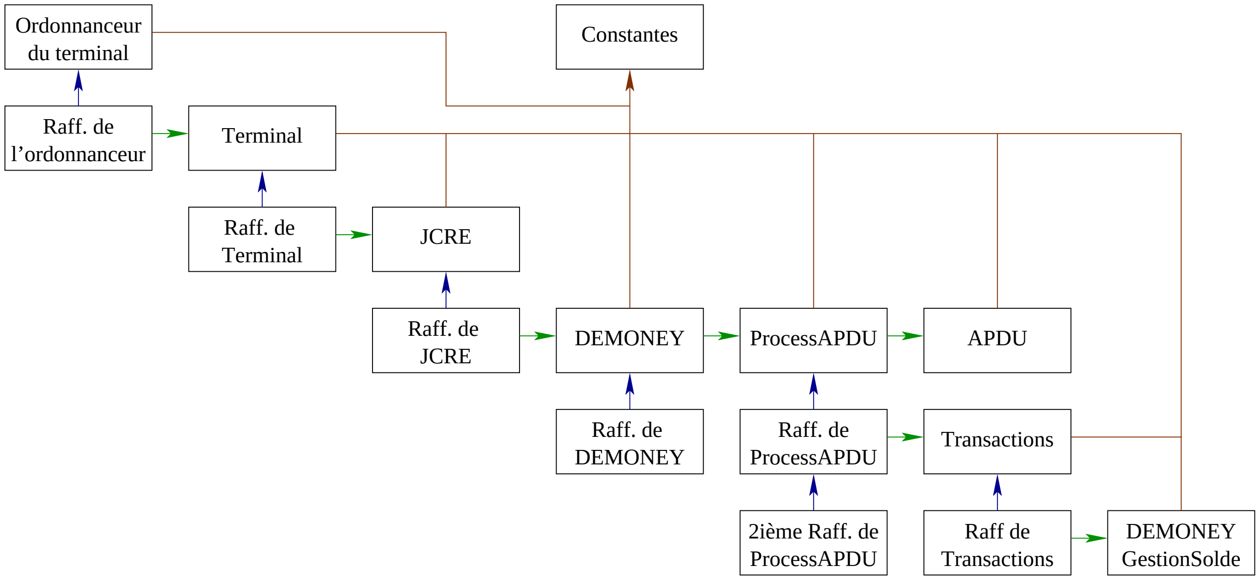
$$Proj_L(E) \mathrel{\hat{=}} \{y \mid \exists x \cdot (L \ \wedge \ E)\}$$

- Nous parlons du *sur-état* E et des *sous-états* E_i.



Contraintes de GénéSyst

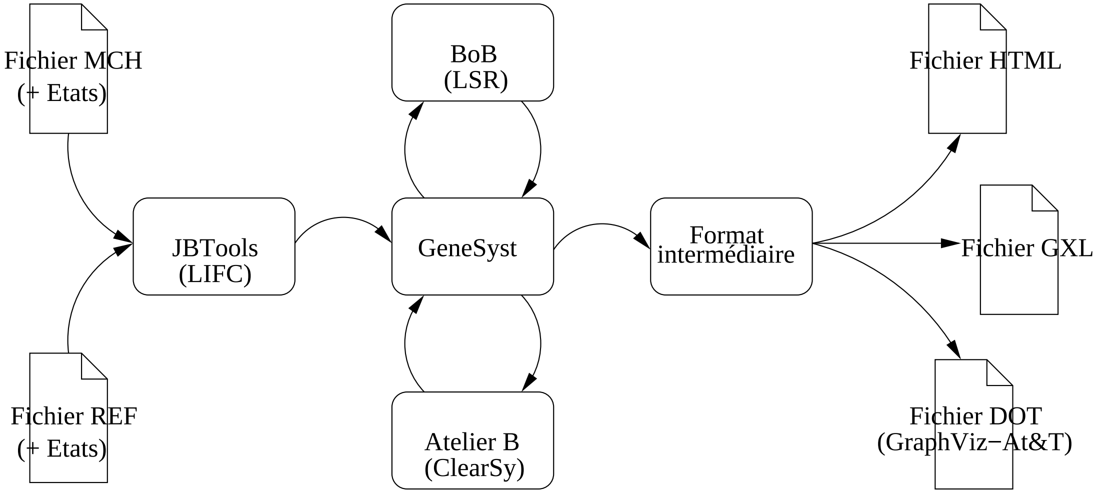
- Un seul niveau de raffinement *(Nouvel outil à exploiter)*
- Problème de la modularité *(Niveau de granularité de la vue)*



Vue d'ensemble du modèle de DEMONEY réalisé

- Restriction au B événementiel.

GénéSyst



- Diffusé sous la licence *CeCILL* (GNU français)
- Publications :
  - [PS04] ML. Potet et N. Stouls, AFADL'04,
    Explicitation du contrôle de développement B événementiel.
  - [BPS04] D. Bert, ML. Potet et N. Stouls, ZB'05,
    GénéSyst*: a Tool to Reason about Behavioral Aspects of* B *Event Specifications. Application to Security Properties*.

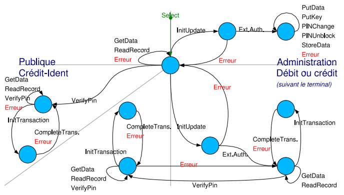
Etude de cas DEMONEY

Etude de cas DEMONEY

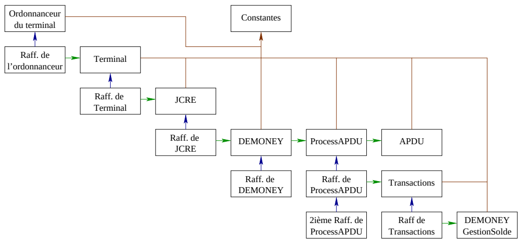
- Porte monnaie électronique pour carte à puce
- Développé par Trusted Logic
- Proche d'un porte monnaie électronique réel
- Spécification informelle publique
- Permet de comparer les différentes approches pour prouver du code carte à puces

Travaux réalisés sur DEMONEY

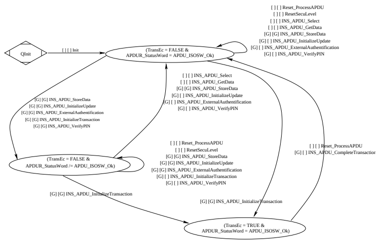
- Extraction de propriétés de sécurité
- Description des propriétés comportementales sous forme de STE



- Modélisation en B



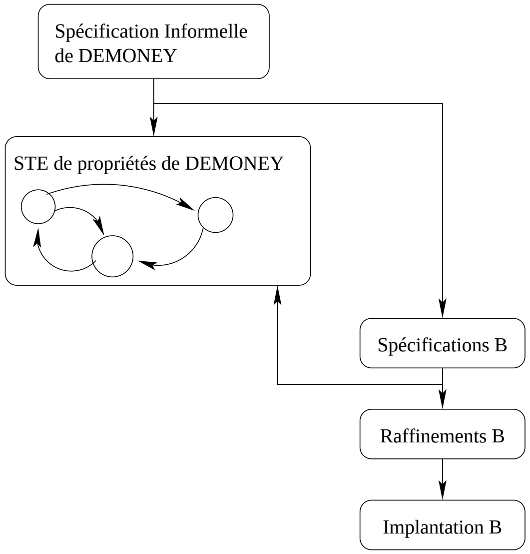
- Extraction par GénéSyst des différentes propriétés



- Vérification des propriétés obtenues par rapport à celles attendues

Comparaison des approches du LRI et de LSR

LSR



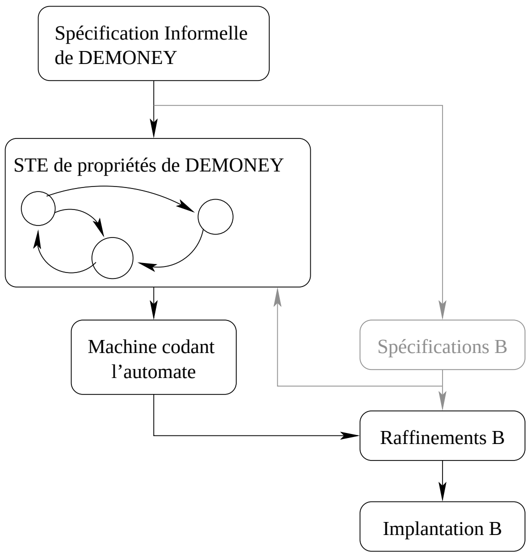
Vérification informelle des propriétés

Approche par raffinement

LRI

Comparaison des approches du LRI et de LSR

LSR



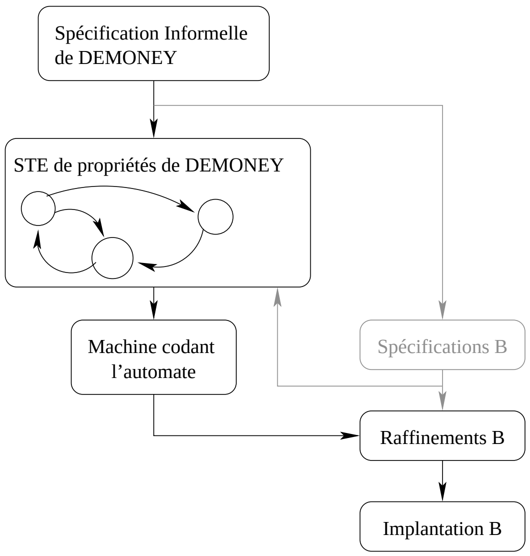
La spécification raffine l'automate et le code raffine la spécification.

Approche par raffinement

LRI

Comparaison des approches du LRI et de LSR

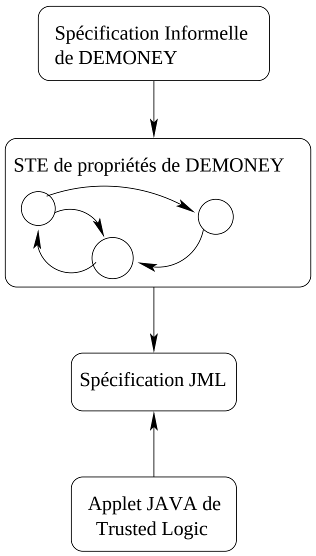
LSR



La spécification raffine l'automate et le code raffine la spécification.

Approche par raffinement

LRI



Le code respecte les pré/post issus de l'automate.

Approche par annotation du code

LSR-IMAG

Conclusion

1. Introduction
2. Construction d’un système de transitions étiqueté
3. Etude de cas **DEMONEY**
4. Conclusion

Perspectives

- Prendre en compte la modularité
  - Visualisation à différents niveaux de granularité
  - Construction du comportement d'un système à base d'opérations
- Intégration du prouveur Barvey
- Générer les états en fonction des propriétés attendues
- Spécifier des contraintes de sécurité par automate
- Produire des spécifications pré/post

LSR-IMAG

Avez vous des questions ?