Geccoo

Génération de code certifié pour des applications orientées objet

Spécification, raffinement, preuve et détection d'erreurs

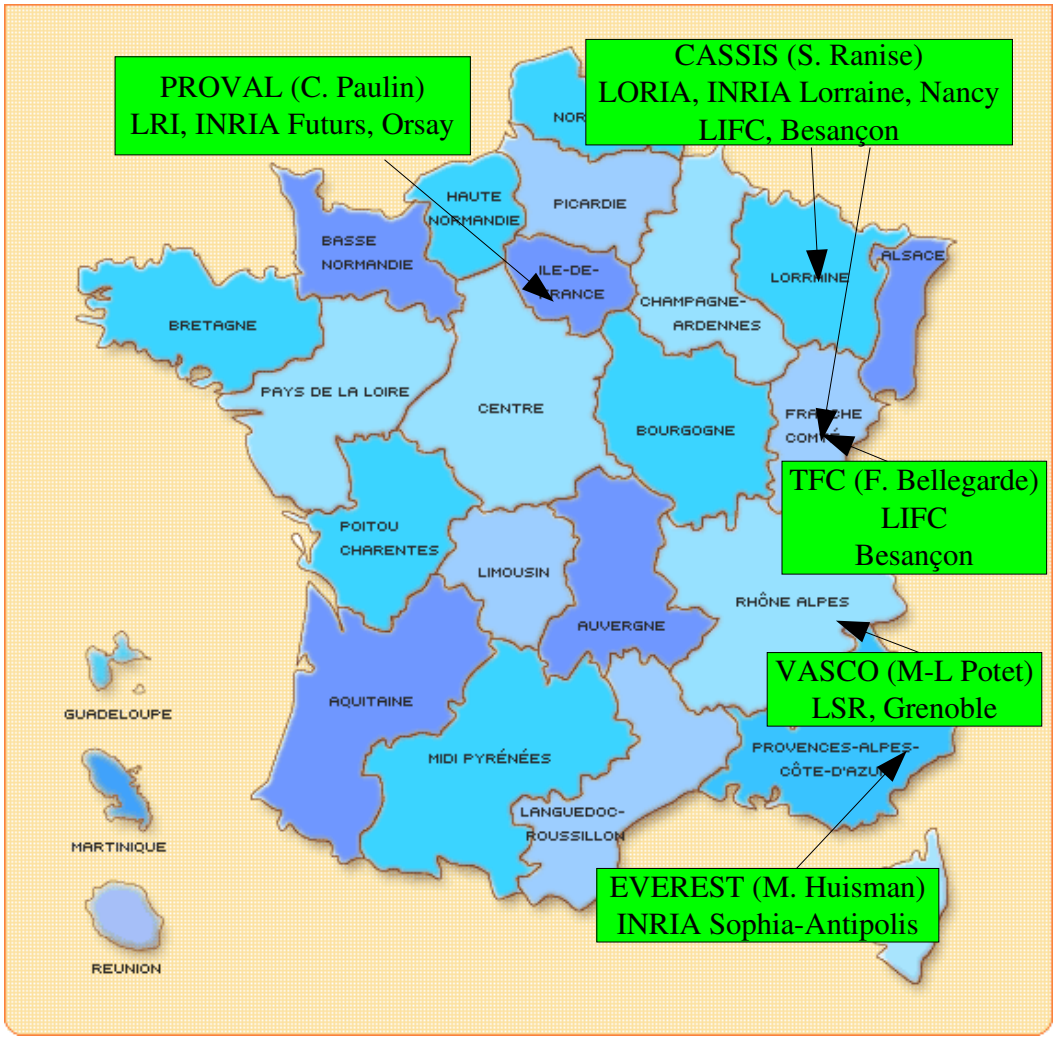
Julien Groslambert[1] Christine Paulin-Mohring[2]

[1]Université de Franche-Comté

[2]Université Paris Sud & INRIA Futurs

Novembre 2005 - PaRI-STIC - Bordeaux

Les équipes



Objectifs

Outils de développement de code orienté objet certifié

- Code critique
- Cartes à puce (JavaCard)/Terminaux

Couvrir la chaîne de développement

- Spécification des propriétés de sécurité;
- Traduction en terme de propriétés logiques;
- Développement modulaire : raffinement et composition;
- Génération et résolution automatique des obligations de preuve;
- Détection d'erreurs : simulation et test.

Objectifs

Outils de développement de code orienté objet certifié

- Code critique
- Cartes à puce (JavaCard)/Terminaux

Couvrir la chaîne de développement

- Spécification des propriétés de sécurité;
- Traduction en terme de propriétés logiques;
- Développement modulaire : raffinement et composition;
- Génération et résolution automatique des obligations de preuve;
- Détection d'erreurs : simulation et test.

Approche

Une approche formelle accessible aux programmeurs :

- Programmes JAVA annotés en JML.
- Automatisation des preuves.

Des modèles rigoureux de haut niveau pour raisonner :

- Systèmes d'évènements, méthode B.
- Assistant de preuve COQ.

Approche

Une approche formelle accessible aux programmeurs :

- Programmes JAVA annotés en JML.
- Automatisation des preuves.

Des modèles rigoureux de haut niveau pour raisonner :

- Systèmes d'évènements, méthode B.
- Assistant de preuve COQ.

Exemple de programme JAVA annoté en JML

```
class Purse {
    int balance;
//@ public invariant balance >= 0;
/*@ public behavior
  @  requires s >= 0;
  @  modifiable balance;
  @  ensures s<=\old(balance)
  @          && balance==\old(balance)-s;
  @  signals (NoCreditException)
  @          s > balance && balance == \old(balance);
  @*/
public void withdraw(int s) throws NoCreditException {
        if (balance >= s) { balance -= s; }
        else { throw new NoCreditException(); }
    }
```

Exemple de programme JAVA annoté en JML

```
class Purse {
    int balance;
//@ public invariant balance >= 0;
/*@ public behavior
  @  requires s >= 0;
  @  modifiable balance;
  @  ensures s<=\old(balance)
  @          && balance==\old(balance)-s;
  @  signals (NoCreditException)
  @          s > balance && balance == \old(balance);
  @*/
public void withdraw(int s) throws NoCreditException {
        if (balance >= s) { balance -= s; }
        else { throw new NoCreditException(); }
    }
```

Exemple de programme JAVA annoté en JML

```
class Purse {
    int balance;
//@ public invariant balance >= 0;
/*@ public behavior
  @  requires s >= 0;
  @  modifiable balance;
  @  ensures s<=\old(balance)
  @          && balance==\old(balance)-s;
  @  signals (NoCreditException)
  @          s > balance && balance == \old(balance);
  @*/
public void withdraw(int s) throws NoCreditException {
        if (balance >= s) { balance -= s; }
        else { throw new NoCreditException(); }
    }
```

Exemple de programme JAVA annoté en JML

```
class Purse {
    int balance;
//@ public invariant balance >= 0;
/*@ public behavior
  @  requires s >= 0;
  @  modifiable balance;
  @  ensures s<=\old(balance)
  @          && balance==\old(balance)-s;
  @  signals (NoCreditException)
  @          s > balance && balance == \old(balance);
  @*/
public void withdraw(int s) throws NoCreditException {
        if (balance >= s) { balance -= s; }
        else { throw new NoCreditException(); }
    }
```

Exemple de programme JAVA annoté en JML

```
class Purse {
    int balance;
//@ public invariant balance >= 0;
/*@ public behavior
  @  requires s >= 0;
  @  modifiable balance;
  @  ensures s<=\old(balance)
  @          && balance==\old(balance)-s;
  @  signals (NoCreditException)
  @          s > balance && balance == \old(balance);
  @*/
public void withdraw(int s) throws NoCreditException {
        if (balance >= s) { balance -= s; }
        else { throw new NoCreditException(); }
    }
```

Une application test Demoney

Porte-monnaie électronique

(Trusted Logic, projet Secsafe)
Analyse des propriétés de sécurité :

- Absence d'erreurs d'exécution : division par zéro, déréférencement du pointeur nul, accès en dehors des bornes d'un tableau. . .
- Authentification : différents niveaux d'identification requis pour chaque opération.
- Enchainement des opérations : l'opération d'initialisation d'une transaction doit être immédiatement suivie de l'opération de complétion de la transation; interruption par arrachage . . .

Une application test Demoney

Porte-monnaie électronique

(Trusted Logic, projet Secsafe)

Analyse des propriétés de sécurité :

- Absence d'erreurs d'exécution : division par zéro, déréférencement du pointeur nul, accès en dehors des bornes d'un tableau. . .
- Authentification : différents niveaux d'identification requis pour chaque opération.
- Enchainement des opérations : l'opération d'initialisation d'une transaction doit être immédiatement suivie de l'opération de complétion de la transation; interruption par arrachage . . .

Une application test Demoney

Porte-monnaie électronique

(Trusted Logic, projet Secsafe)
Analyse des propriétés de sécurité :

- Absence d'erreurs d'exécution : division par zéro, déréférencement du pointeur nul, accès en dehors des bornes d'un tableau. . .
- Authentification : différents niveaux d'identification requis pour chaque opération.
- Enchainement des opérations : l'opération d'initialisation d'une transaction doit être immédiatement suivie de l'opération de complétion de la transation; interruption par arrachage . . .

Une application test Demoney

Porte-monnaie électronique

(Trusted Logic, projet Secsafe)
Analyse des propriétés de sécurité :

- Absence d'erreurs d'exécution : division par zéro, déréférencement du pointeur nul, accès en dehors des bornes d'un tableau. . .
- Authentification : différents niveaux d'identification requis pour chaque opération.
- Enchainement des opérations : l'opération d'initialisation d'une transaction doit être immédiatement suivie de l'opération de complétion de la transation; interruption par arrachage . . .

Une application test Demoney

Porte-monnaie électronique

(Trusted Logic, projet Secsafe)
Analyse des propriétés de sécurité :

- Absence d'erreurs d'exécution : division par zéro, déréférencement du pointeur nul, accès en dehors des bornes d'un tableau. . .
- Authentification : différents niveaux d'identification requis pour chaque opération.
- Enchainement des opérations : l'opération d'initialisation d'une transaction doit être immédiatement suivie de l'opération de complétion de la transation; interruption par arrachage . . .

Principales difficultés

- Établir les modèles et raisonner dessus.
- Propager les contraintes de sécurité jusqu'au code.
- Résoudre les obligations.

Modélisation (1)

Génésyst : Analyse du comportement d'un système B événementiel

N. Stouls, D. Bert, S. Hamdan, H. Mohamed, X. Morselli.
Vasco, LSR, IMAG, Grenoble

L'initialisation de la transaction doit être immédiatement suivie de sa complétion.

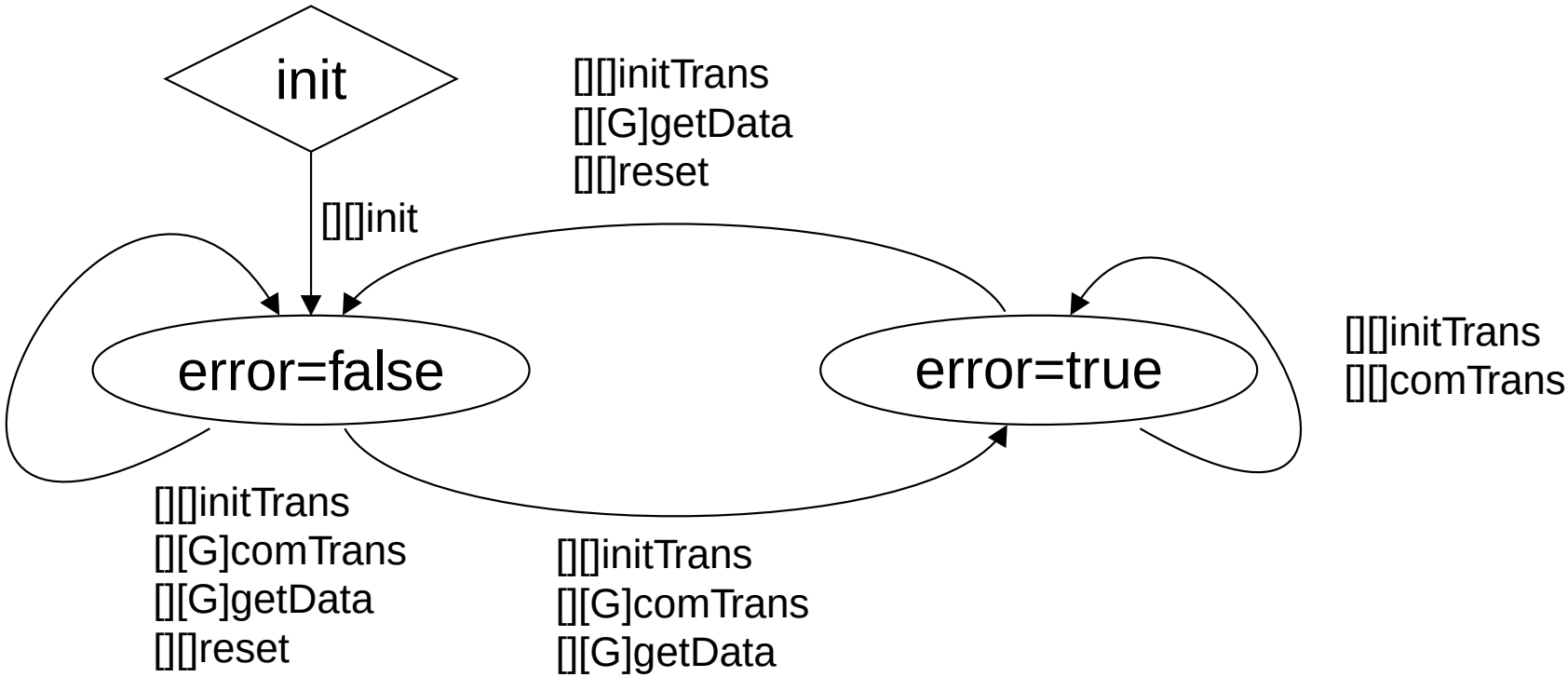
init	error,engaged := false,false
reset	error,engaged := false,false
getData	**if** engaged **then** error,engaged := true,false **else** error := false
initTrans	**if** engaged **then** error,engaged := true,false
	else error,engaged := false,true + error,engaged := true,false
comTrans	**if** engaged **then** error,engaged := false,false **else** error := true

Modélisation (1)

Génésyst : Analyse du comportement d'un système B événementiel

N. Stouls, D. Bert, S. Hamdan, H. Mohamed, X. Morselli.

Vasco, LSR, IMAG, Grenoble

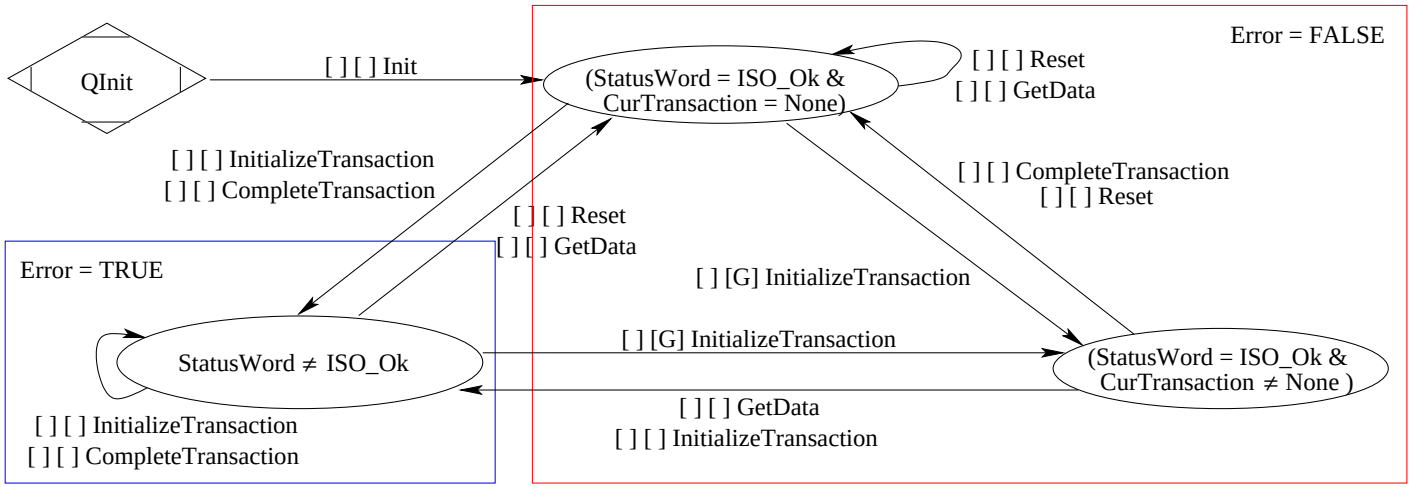


Modélisation (1)

Génésyst : Analyse du comportement d'un système B événementiel

N. Stouls, D. Bert, S. Hamdan, H. Mohamed, X. Morselli.

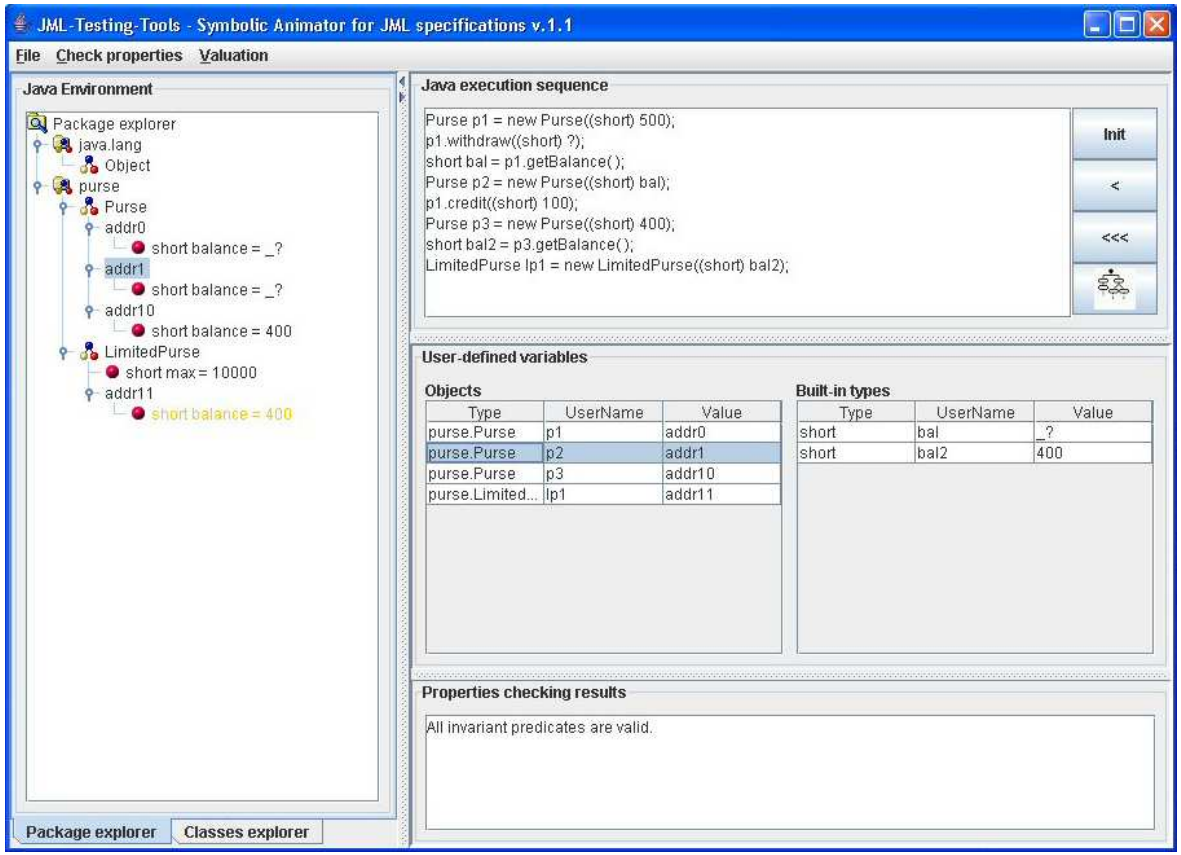
Vasco, LSR, IMAG, Grenoble



Modélisation (2)

JML Testing-Tools Animation et test de spécifications JML

F. Dadeau, F. Bouquet, B. Legeard
TFC, LIFC, Besançon.



Génération d'obligations de preuve

Jack : L. Burdy, J. Charles, J-L Lanet, M. Pavlova.
Gemplus-Everest, INRIA Sophia-Antipolis.
Krakatoa : C. Marché, C. Paulin, (X. Urbain).
ProVal, LRI, INRIA Futurs.

Entrée : Programme Java annoté en JML

Génération automatique : Modélisation (B ou Coq) du programme (classes, mémoire. . .)

Analyse statique Interprétation du programme et de sa spécification comme un programme impératif avec assertions

Génération d'obligations de preuve à prouver interactivement ou automatiquement Coq, B, Simplify, haRVey. . .

Génération d'obligations de preuve

Jack : L. Burdy, J. Charles, J-L Lanet, M. Pavlova.
Gemplus-Everest, INRIA Sophia-Antipolis.
Krakatoa : C. Marché, C. Paulin, (X. Urbain).
ProVal, LRI, INRIA Futurs.

Entrée : Programme Java annoté en JML

Génération automatique : Modélisation (B ou COQ) du programme (classes, mémoire. . .)

Analyse statique Interprétation du programme et de sa spécification comme un programme impératif avec assertions

Génération d'obligations de preuve à prouver interactivement ou automatiquement COQ, B, SIMPLIFY, haRVey. . .

Génération d'obligations de preuve

Jack : L. Burdy, J. Charles, J-L Lanet, M. Pavlova.
Gemplus-Everest, INRIA Sophia-Antipolis.
Krakatoa : C. Marché, C. Paulin, (X. Urbain).
ProVal, LRI, INRIA Futurs.

Entrée : Programme Java annoté en JML

Génération automatique : Modélisation (B ou COQ) du programme (classes, mémoire. . .)

Analyse statique Interprétation du programme et de sa spécification comme un programme impératif avec assertions

Génération d'obligations de preuve à prouver interactivement ou automatiquement COQ, B, SIMPLIFY, haRVey. . .

Génération d'obligations de preuve

Jack : L. Burdy, J. Charles, J-L Lanet, M. Pavlova.
Gemplus-Everest, INRIA Sophia-Antipolis.
Krakatoa : C. Marché, C. Paulin, (X. Urbain).
ProVal, LRI, INRIA Futurs.

Entrée : Programme Java annoté en JML

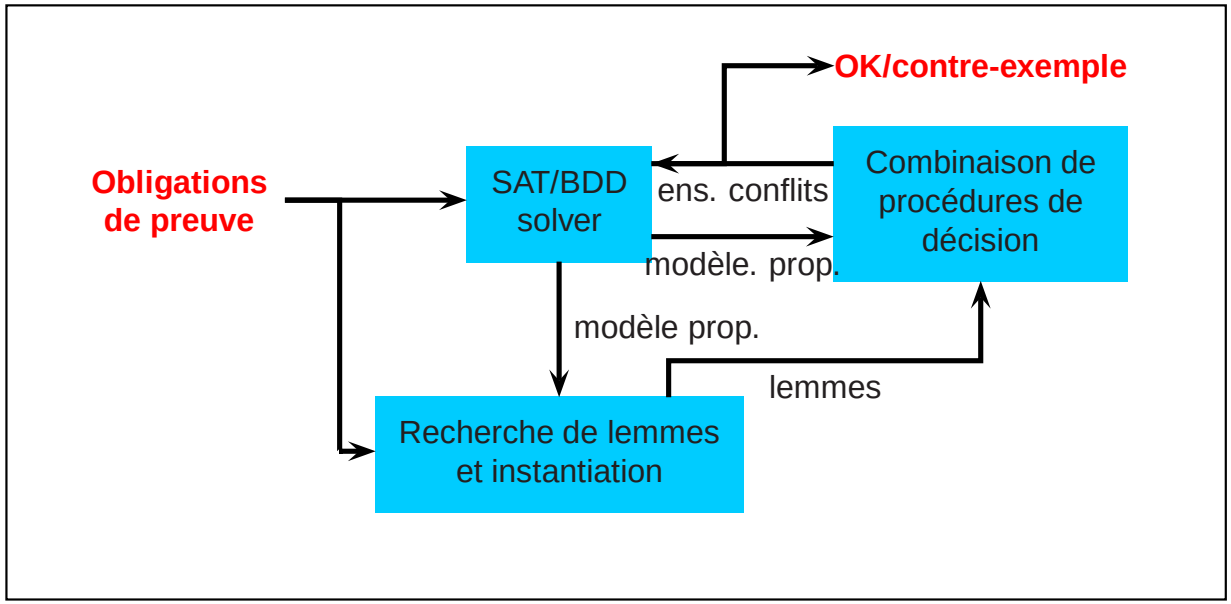
Génération automatique : Modélisation (B ou COQ) du programme (classes, mémoire. . .)

Analyse statique Interprétation du programme et de sa spécification comme un programme impératif avec assertions

Génération d'obligations de preuve à prouver interactivement ou automatiquement COQ, B, SIMPLIFY, haRVey. . .

Résolution des obligations de preuve

haRVey : D. Déharbe, P. Fontaine, S. Ranise.
Cassis, LORIA, INRIA Lorraine.
haRVey vérifie la satisfiabilité de formules de la logique du premier ordre par combinaison de raisonnement propositionnel et de procédures de décision pour des types de données.



et un challenge pour les outils

Résumé

- Adaptation, enrichissement de techniques existantes au cas JAVA/JML.
- Importance de l'automatisation des annotations, des preuves.
- La confrontation à l'étude de cas Demoney est un challenge pour les outils.

Problématique

Vérification formelle de la **conformité** entre:

- Cahier des charges.
- Code source de l'implémentation.

L'approche proposée dans GECCOO.

- Exprimer des **propriétés sécurité** à partir du cahier des charges
- Traduire les propriétés dans un **format permettant la vérification formelle** sur le code Java.
- Vérifier les propriétés sur le **code Java**.

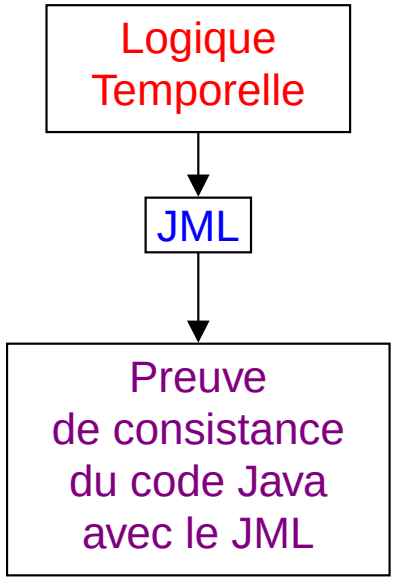
Problématique

Vérification formelle de la **conformité** entre:

- Cahier des charges.
- Code source de l'implémentation.

L'approche proposée dans GECCOO.

- Exprimer des **propriétés sécurité** à partir du cahier des charges
- Traduire les propriétés dans un **format permettant la vérification formelle** sur le code Java.
- Vérifier les propriétés sur le **code Java**.



Exprimer des propriétés de sécurité (1/2)

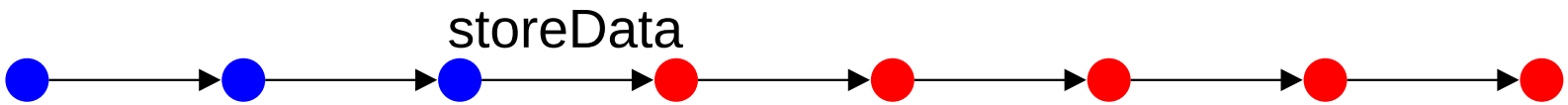
Langage de logique temporelle.

- Proposé par Huisman and Trentelman (AMAST'02).
- **Sémantique de traces** adaptée à Java/JML
- Prise en compte **terminaisons exceptionnelles**, appel de méthode..
- Expression de **sûretés** et de **vivacités**.

Exemple de propriétés temporelles(2/2)

Extraite de la specification **Demoney** (Trusted Logic).

after storeData() **normal always** storeData() **not enabled;**

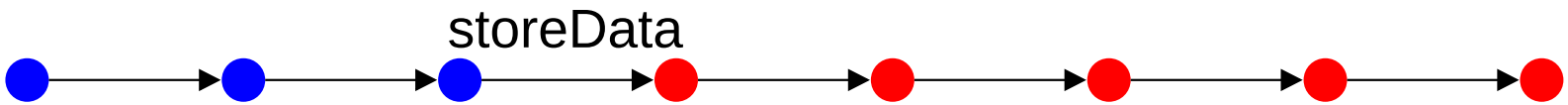


after initializeTransaction() **called**
always initializeTransaction() **not enabled**
unless completeTransaction() **called**;

Exemple de propriétés temporelles(2/2)

Extraite de la specification **Demoney** (Trusted Logic).

after storeData() **normal always** storeData() **not enabled;**

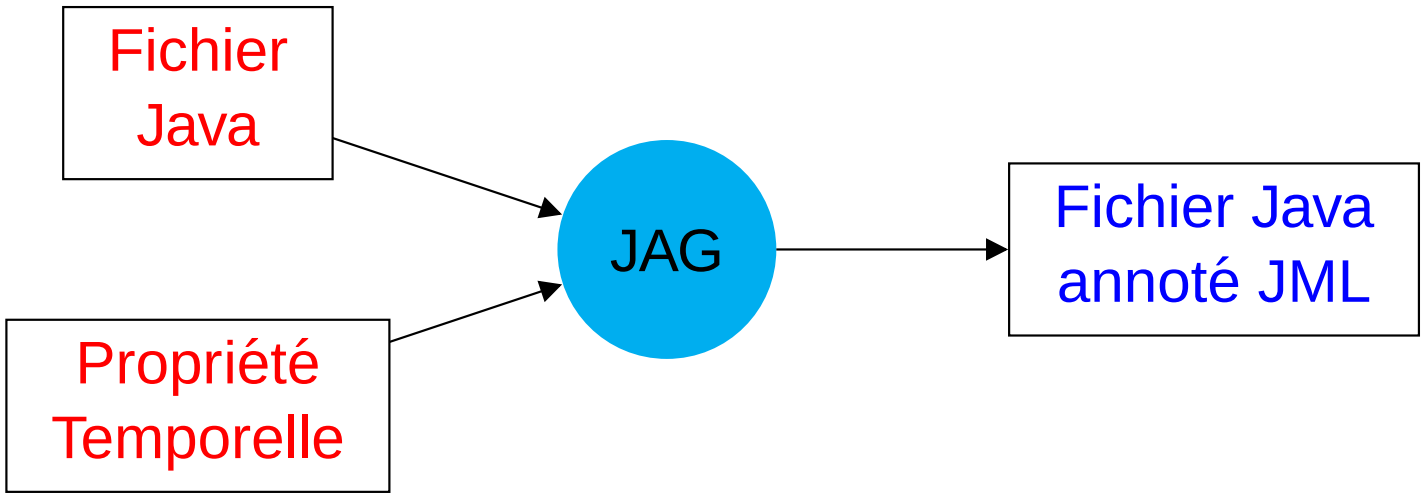


after initializeTransaction() **called**
always initializeTransaction() **not enabled**
unless completeTransaction() **called**;

Traduction des propriétés

Traduction automatique en annotations **JML standard**.

- **Equivalence** sémantique entre les formules temporelles et les annotations générées.
- **Implantation** dans l'outil JAG.
- **Traçabilité** des annotations générées.



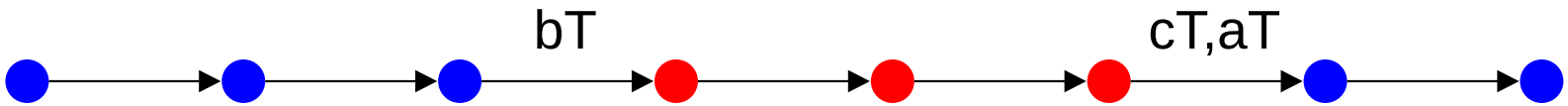
Traduction des propriétés - exemple (1/2)

Système de transactions - trois méthodes

- beginTransaction()
- commitTransaction()
- abortTransaction()

Propriété: *Lorsqu'une transaction est en cours, un verrou doit être en place.*

after beginTransaction **called**
always {lock == true}
unless commitTransaction **called**, abortTransaction **called**.



Traduction des propriétés - exemple (2/2)

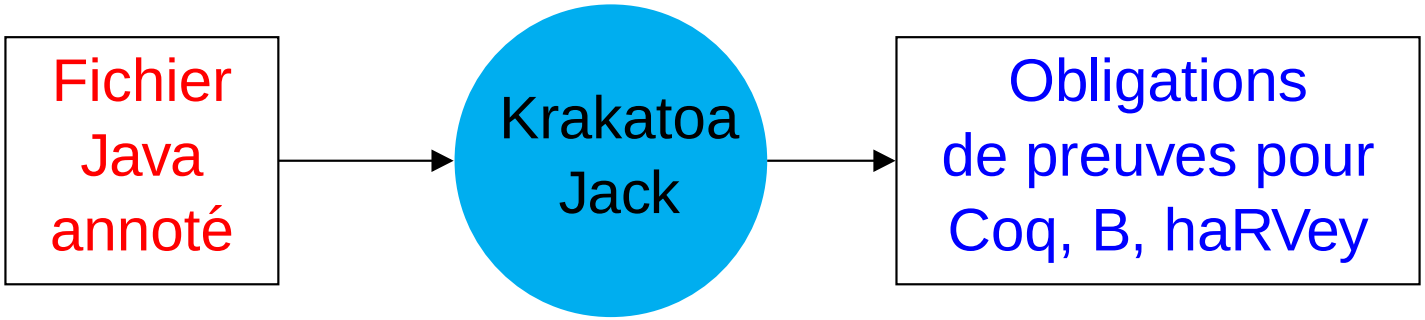
JAG génère le fichier Java/JML suivant

```
//@ ghost boolean bt =false;
//@ invariant bt ==> lock == true;
void beginTransaction(){
//@ set bt = true;
...}
 void commitTransaction(){
 //@ set bt = false;
 ...}
 void abortTransaction(){
 //@ set bt = false;
 ...}
```

Vérification des propriétés

Vérification de la **conformité** des annotations JML générées avec le **code Java** de l'implémentation.

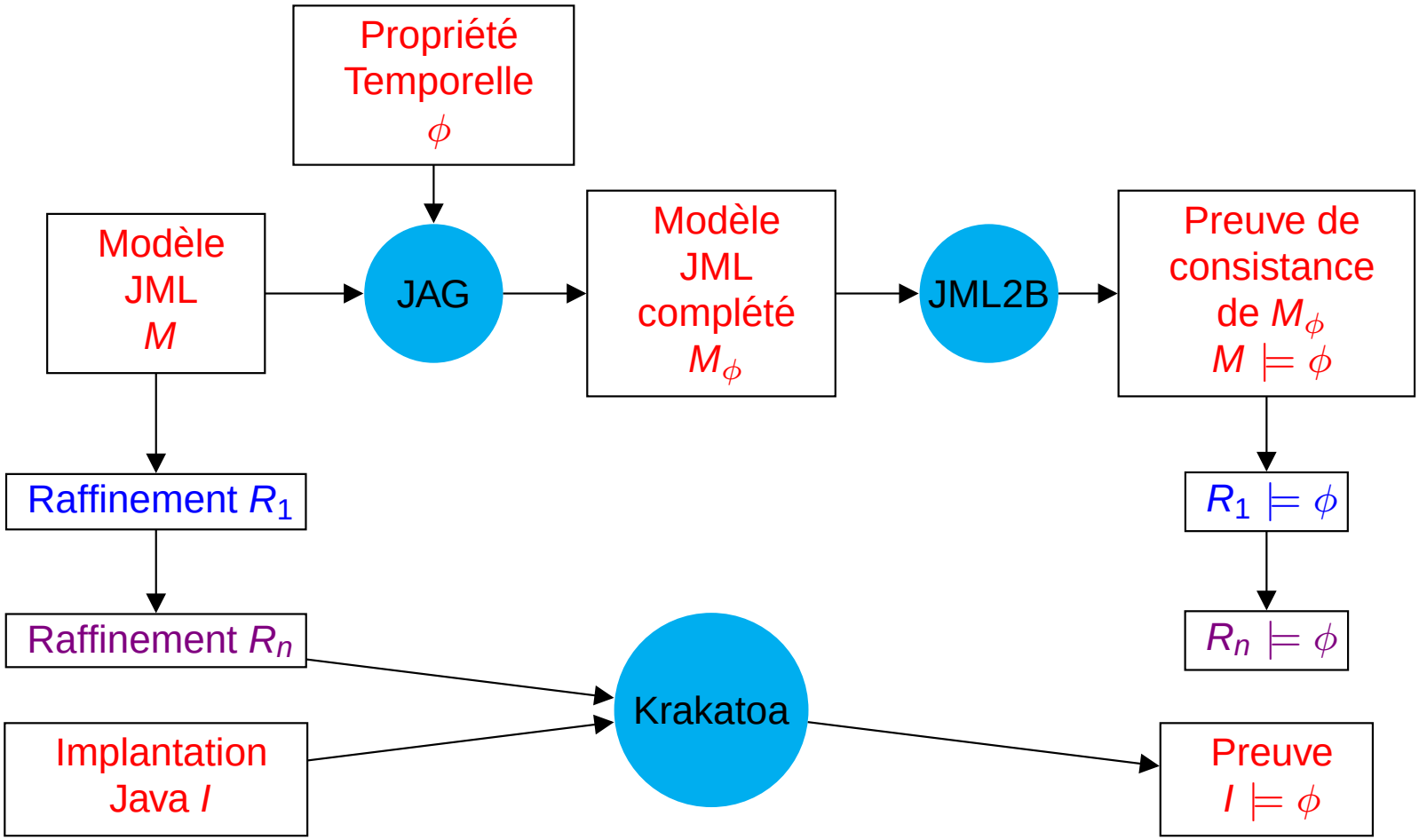
- Outils de génération d'obligations de preuve
  - Krakatoa (LRI)
  - Jack (Everest - INRIA Sophia)
- Sortie vers prouveurs automatiques et interactifs.



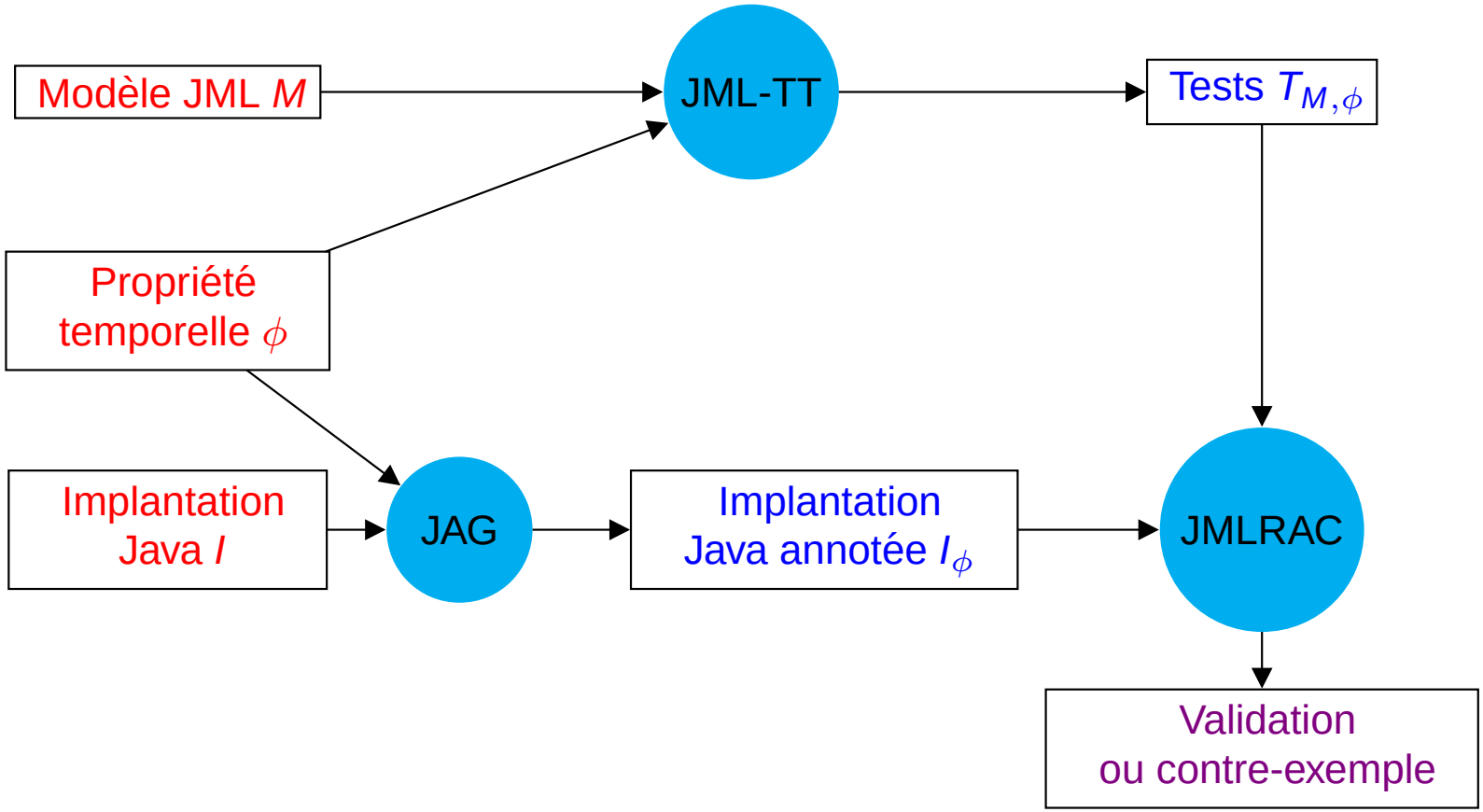
Problématiques actuelles

1. **Difficulté d'exprimer les propriétés temporelles directement sur l'implantation**
   - Propriétés temporelles de haut niveau
   - Implantation de bas niveau.
   - **Proposition**: Raffinement.
2. **Echec des obligations de preuve**
   - Soit erreur d'implantation.
   - Soit implantation correcte mais échec des heuristiques.
   - **Proposition**: Génération automatique de tests pour valider ou invalider l'implantation.

Raffinement



Génération automatique de tests



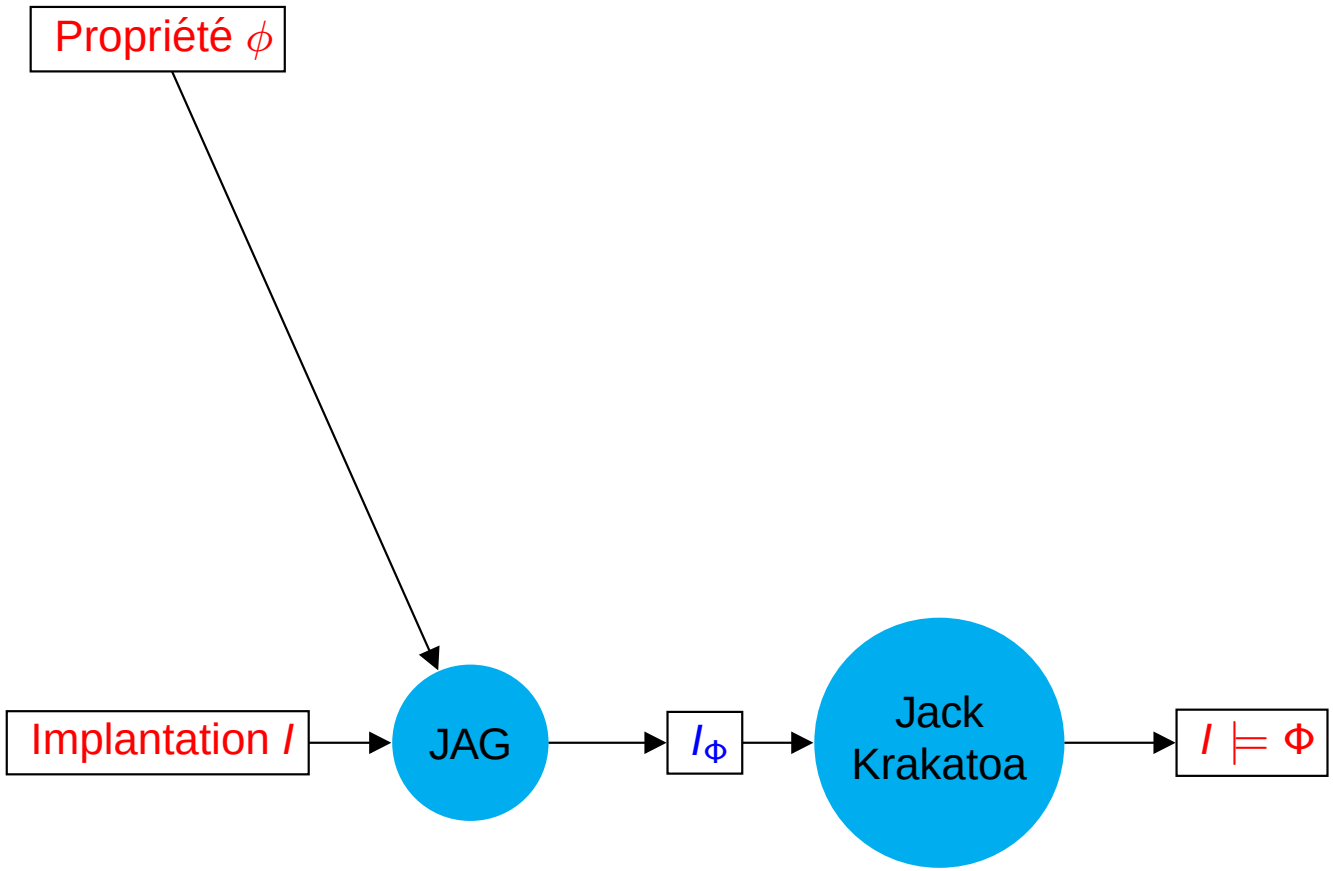
Résumé de la méthode

Propriété ϕ

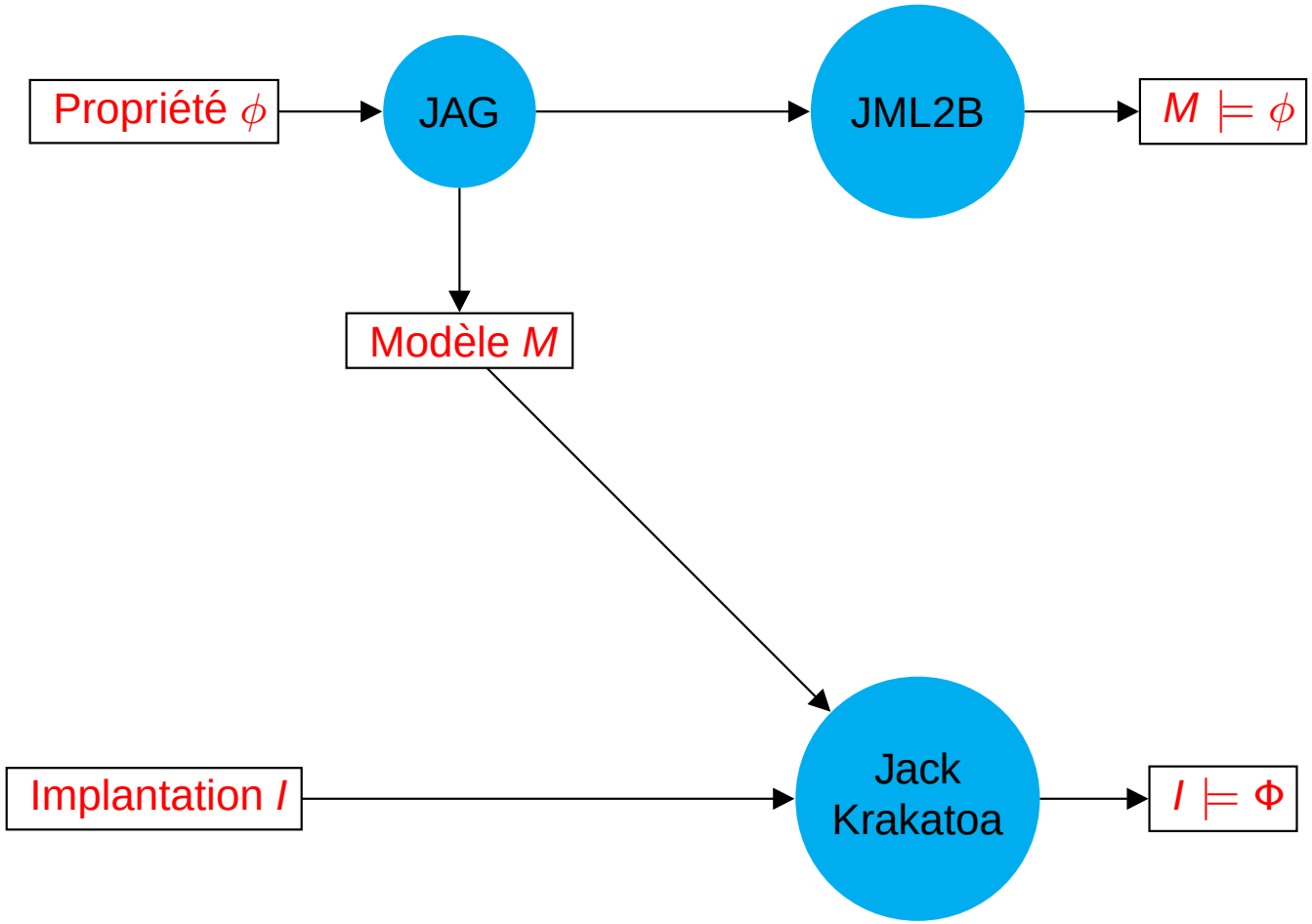
Implantation I

$I \models \Phi$

Résumé de la méthode



Résumé de la méthode



Résumé de la méthode