# Réduction paramétrée de spécifications formées d'automates communicants : algorithmes polynomiaux pour la réduction de modèles

Soutenance de thèse de
Sébastien LABBÉ

CEA, LIST / Université Pierre et Marie Curie – Paris VI

| | | |
|---|---|---|
| Jury : | M. Jean-Claude FERNANDEZ | Rapporteur |
| | M. Yves LE TRAON | Rapporteur |
| | M. Marc AIGUIER | Examinateur |
| | Mme. Thérèse HARDIN | Examinateur |
| | M. Marc POUZET | Directeur de thèse |
| | M. Jean-Pierre GALLOIS | Encadrant de thèse |

# Plan

# Contexte (1/2)

- Sciences et technologies industrielles de pointe

# Contexte (1/2)

- Sciences et technologies industrielles de pointe

# Contexte (1/2)

- Sciences et technologies industrielles de pointe



# Contexte (1/2)

- Sciences et technologies industrielles de pointe




# Contexte (1/2)

- Sciences et technologies industrielles de pointe




  - Enjeux économiques et humains élevés
  - Part croissante des logiciels dans les systèmes

# Contexte (1/2)

- Sciences et technologies industrielles de pointe



  - Enjeux économiques et humains élevés
  - Part croissante des logiciels dans les systèmes

- Minimiser les risques d'échec pour une expérience

# Contexte (1/2)

- Sciences et technologies industrielles de pointe



  - Enjeux économiques et humains élevés
  - Part croissante des logiciels dans les systèmes

- Minimiser les risques d'échec pour une expérience
- Étude de la sûreté de fonctionnement des systèmes
  - Fiabilité
  - Disponibilité
  - Maintenabilité
  - Sécurité

# Contexte (2/2)

## Méthodes formelles

- Méthodes fondées mathématiquement
- Langages de spécification formellement définis
  - Syntaxe, sémantique

# Contexte (2/2)

## Méthodes formelles

- Méthodes fondées mathématiquement
- Langages de spécification formellement définis
  - Syntaxe, sémantique
- Outils formels
  - Exemples : outils de raffinement, de génération de code
  - Cadres théoriques
    - Exemple : Théorie des treillis pour les analyses statiques
  - Algorithmes
    - Correction, complétude, complexité

# Contexte (2/2)

## Méthodes formelles

- Méthodes fondées mathématiquement
- Langages de spécification formellement définis
  - Syntaxe, sémantique
- Outils formels
  - Exemples : outils de raffinement, de génération de code
  - Cadres théoriques
    - Exemple : Théorie des treillis pour les analyses statiques
  - Algorithmes
    - Correction, complétude, complexité
- Validation
  - Test, simulation
- Vérification
  - Preuve de propriétés

# Problématique

## Méthodes formelles (suite)

- Complexité importante
  - Examen de toutes les exécutions possibles
- Explosion combinatoire
  - Larges domaines numériques
  - Parallélisme

# Problématique

## Méthodes formelles (suite)

- Complexité importante
  - Examen de toutes les exécutions possibles
- Explosion combinatoire
  - Larges domaines numériques
  - Parallélisme

## Exemple : l'approche AGATHA

- Analyse, validation de systèmes constitués d'automates communicants
- Exécution symbolique
  - Représentation compacte de l'ensemble des comportements du système
- Spécifications industrielles, systèmes réactifs

# Solution proposée

- Problèmes NP-complets
- Abstraire ou *partitionner* les problèmes

# Solution proposée

- Problèmes NP-complets
- Abstraire ou *partitionner* les problèmes
- Traitement en amont d'une analyse complexe
- Analyses formelles souvent "bridées" par un critère
  - Exemple : propriété à prouver (preuve de propriétés)

# Solution proposée

- Problèmes NP-complets
- Abstraire ou *partitionner* les problèmes
- Traitement en amont d'une analyse complexe
- Analyses formelles souvent "bridées" par un critère
  - Exemple : propriété à prouver (preuve de propriétés)
- Réduction paramétrée de spécifications formées d'automates communicants
  - *Slicing* de programmes
  - Construction de compilateurs (analyses de flot de données)
  - Analyse de dépendances dans les spécifications
  - Algorithmes polynomiaux
- Spécification réduite, potentiellement moins sujette à l'explosion combinatoire

# Plan

# Plan

# Introduction au “slicing” de programmes

- Analyse statique introduite par Weiser [Weiser, ICSE'81]
  - **Objectif initial** : débogage et compréhension de programmes
  - **Entrées** : un programme exprimé dans un langage séquentiel et impératif et un *critère*
  - **Sortie** : une *tranche* qui contient tous les éléments du programme, pertinents par rapport au critère.
- D'un problème général, en extraire un plus spécifique, focalisé sur quelque(s) point(s) d'intérêt.

# Introduction au "slicing" de programmes

- Analyse statique introduite par Weiser [Weiser, ICSE'81]
  - **Objectif initial** : débogage et compréhension de programmes
  - **Entrées** : un programme exprimé dans un langage séquentiel et impératif et un *critère*
  - **Sortie** : une *tranche* qui contient tous les éléments du programme, pertinents par rapport au critère.
- D'un problème général, en extraire un plus spécifique, focalisé sur quelque(s) point(s) d'intérêt.
- Depuis, de nombreuses extensions...
  - Constructions plus complexes (e.g. tableaux, pointeurs, procédures, concurrence)
  - Formalismes plus récents (e.g. langages synchrones, spécifications Z).
- ... pour des applications dans la simulation [Millett et al., STTT'00], la génération de cas de test [Bozga et al., STTT'03], le model checking [Dwyer et al., TACAS'06].

# Exemple de réduction paramétrée

## Exemple de programme impératif

```
1   somme := 0
2   prod := 1
3   a := 1
4   b := lireEntier()
5   tant que (a<=b){
6     somme := somme+a
7     prod := prod*a
8     a := a+1
9   }
10  imprimer(somme)
11  imprimer(prod)
```

# Exemple de réduction paramétrée

## Exemple de programme impératif

```
1   somme := 0
2   prod := 1
3   a := 1
4   b := lireEntier()
5   tant que (a<=b){
6     somme := somme+a
7     prod := prod*a
8     a := a+1
9   }
10  imprimer(somme)
11  imprimer(prod)
```

11 lignes de code
Variables : $\{a, b, somme, prod\}$

## Tranche par rapport au critère {ligne 11}

```
1   somme := 0
2   prod := 1
3   a := 1
4   b := lireEntier()
5   tant que (a<=b){
6     somme := somme+a
7     prod := prod*a
8     a := a+1
9   }
10  imprimer(somme)
11  imprimer(prod)
```

8 lignes de code
Variables : $\{a, b, prod\}$

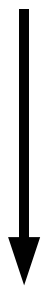
# Plan

## Automates : IOSTS

*Input/Output Symbolic Transitions Systems*

Dénotent les comportements du système spécifié

- Ensemble d'états $S$, un état initial, ensemble de transitions $T$
- Interactions avec l'environnement : communications sur des *canaux* (ensemble $C$)
- Actions internes : affectation de valeurs à des *variables attributs* (ensemble $V$)



[x>0]
a
x := x+y

$a$ est de la forme

$$\tau \mid c? \mid c?x \mid c! \mid c!t$$

où $c \in C$, $x \in V$ et $t$ un terme

# Spécifications formées d'automates communicants

## Automates : IOSTS

*Input/Output Symbolic Transitions Systems*

Dénotent les comportements du système spécifié

- Ensemble d'états $S$, un état initial, ensemble de transitions $T$
- Interactions avec l'environnement : communications sur des *canaux* (ensemble $C$)
- Actions internes : affectation de valeurs à des *variables attributs* (ensemble $V$)

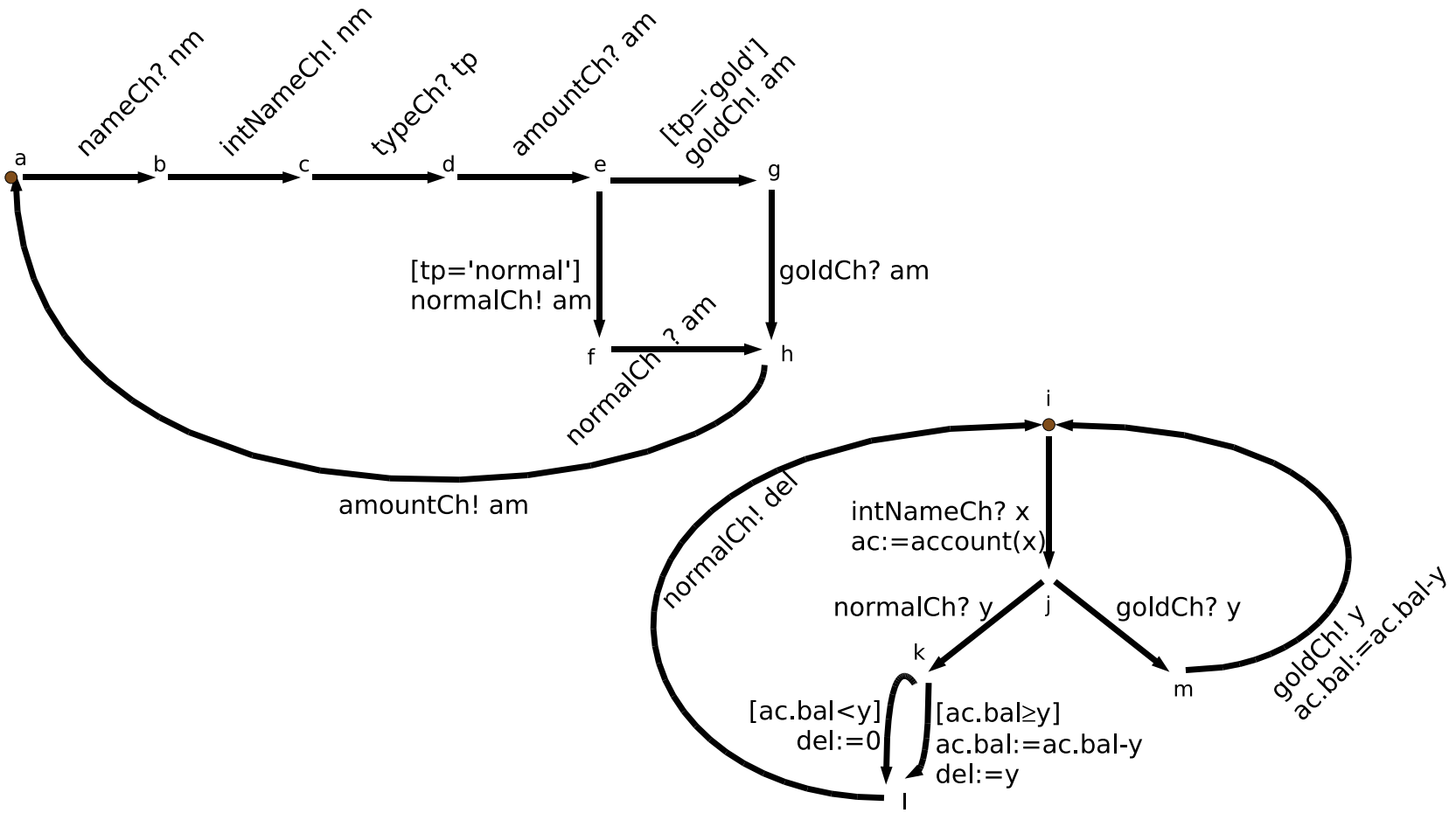
## Spécification formée d'IOSTS

Un ensemble d'automates.

- Ensembles des variables attributs disjoints 2 à 2
- Communications par rendez-vous binaire
- Composition parallèle par entrelacements

# Exemple de spécification

- Distributeur de billets



# Plan

# Dépendances dans les programmes

## Relations de dépendances : relations binaires

- Dépendances de **données** et de **contrôle**
  - Construction de compilateurs [Allen Kennedy, 2002]
    - Transformation, optimisation (déplacement de code, parallélisation...) [Ferrante et al., TOPLAS'87]
  - Analyse de programmes
    - Réduction paramétrée [Thèse Krinke, 2003]

# Dépendances dans les programmes

## Relations de dépendances : relations binaires

- Dépendances de **données** et de **contrôle**
  - Construction de compilateurs [Allen Kennedy, 2002]
    - Transformation, optimisation (déplacement de code, parallélisation...) [Ferrante et al., TOPLAS'87]
  - Analyse de programmes
    - Réduction paramétrée [Thèse Krinke, 2003]
- Calcul des dépendances de données [Muchnick, 1997]
  - Analyse de flot de données "propagation des définitions"

# Dépendances dans les programmes

## Relations de dépendances : relations binaires

- Dépendances de **données** et de **contrôle**
  - Construction de compilateurs [Allen Kennedy, 2002]
    - Transformation, optimisation (déplacement de code, parallélisation...) [Ferrante et al., TOPLAS'87]
  - Analyse de programmes
    - Réduction paramétrée [Thèse Krinke, 2003]
- Calcul des dépendances de données [Muchnick, 1997]
  - Analyse de flot de données "propagation des définitions"
- Calcul des dépendances de contrôle
  - Analyse de flot de contrôle "calcul des post-dominateurs" [Muchnick, 1997]
  - Analyse symbolique du flot de contrôle [Ranganath et al., ESOP'05]

# Dépendances dans les spécifications

## Spécificités

- Flot de contrôle non-structuré
  - Boucles à entrées multiples, indéterminisme...
- Système réactifs
  - Pas d'hypothèse sur des point(s) de sortie
  - Interagissent continûment
- Communications par canaux
  - Flots additionnels de contrôle et de données

# Dépendances dans les spécifications

## Spécificités

- Flot de contrôle non-structuré
  - Boucles à entrées multiples, indéterminisme...
- Système réactifs
  - Pas d'hypothèse sur des point(s) de sortie
  - Interagissent continûment
- Communications par canaux
  - Flots additionnels de contrôle et de données

## Solutions

- Relations spécifiques de dépendance de **contrôle** et de **données** (intra-automates)
- Relation additionnelle de dépendance de **communication** (inter-automates)

# Dépendance de données (1/2)

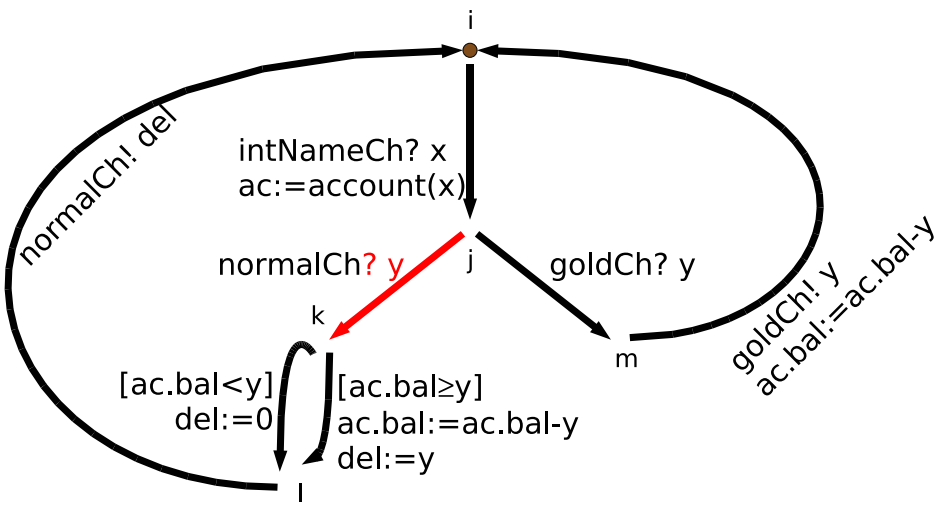
*Une transition utilise potentiellement une valeur définie par une autre transition*

- Requiert les notions de définition/utilisation d'une variable

# Dépendance de données (1/2)

*Une transition utilise potentiellement une valeur définie par une autre transition*

- Requiert les notions de définition/utilisation d'une variable :

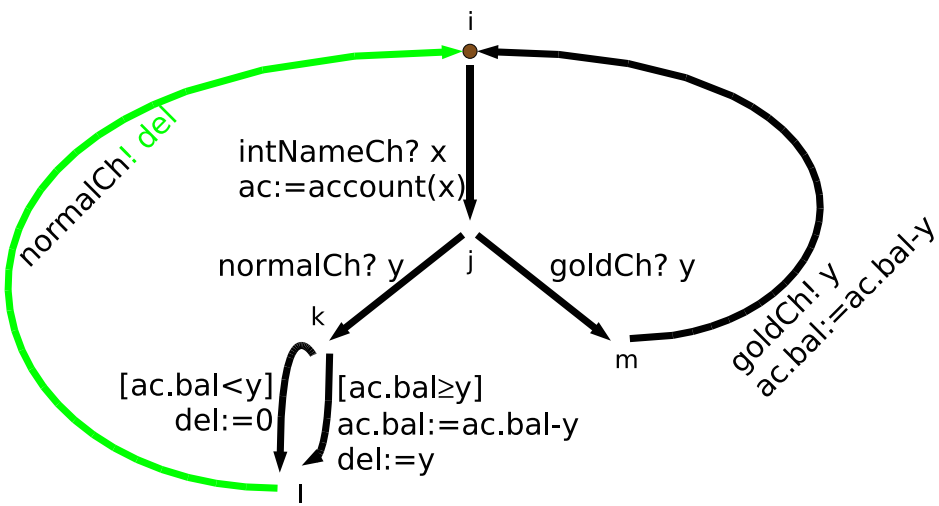



- $y$ est défini par $j \rightharpoonup k$

# Dépendance de données (1/2)

*Une transition utilise potentiellement une valeur définie par une autre transition*

- Requiert les notions de définition/utilisation d'une variable :

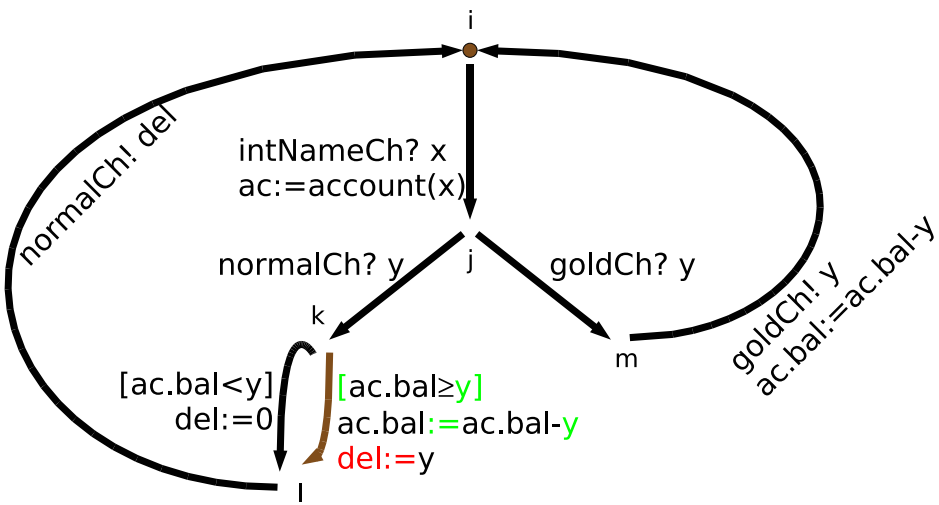



- $y$ est défini par $j \rightharpoonup k$
- $del$ est utilisé par $l \rightharpoonup i$

# Dépendance de données (1/2)

*Une transition utilise potentiellement une valeur définie par une autre transition*

- Requiert les notions de définition/utilisation d'une variable :




- $y$ est défini par $j \rightharpoonup k$
- *del* est utilisé par $l \rightharpoonup i$
- $y$ est utilisé et *del* est défini par $k \rightharpoonup l$

# Dépendance de données (2/2)

## Dépendance de données ($tr_i \xrightarrow{dd} tr_j$)

$tr_j$ est *données-dépendant* de $tr_i$

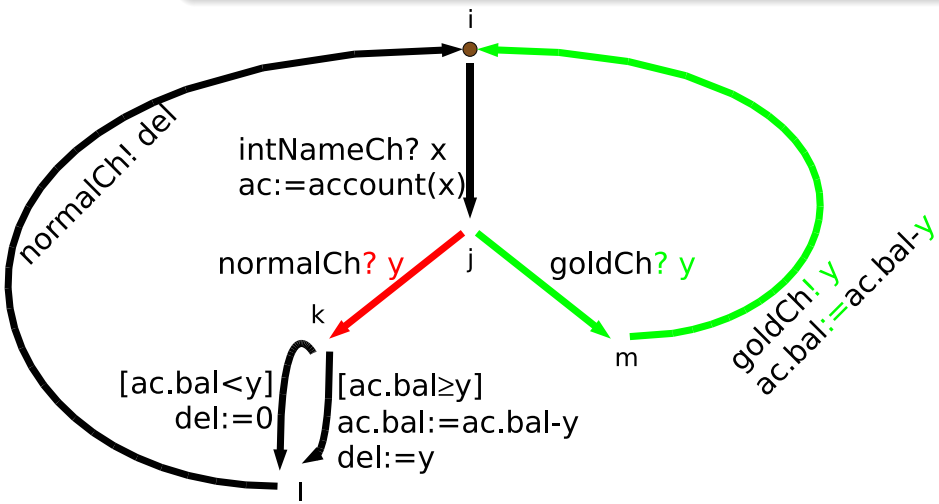
- $x$ est défini par $tr_i$ et il existe un chemin de $tr_i$ à $tr_j$ sur lequel $x$ n'est pas redéfini
- et
  - $x$ est utilisé par la garde de $tr_j$
  - ou $x$ n'est pas défini par l'action de $tr_j$ et $x$ est utilisé par $tr_j$

# Dépendance de données (2/2)

## Dépendance de données ($tr_i \xrightarrow{dd} tr_j$)

$tr_j$ est *données-dépendant* de $tr_i$

- $x$ est défini par $tr_i$ et il existe un chemin de $tr_i$ à $tr_j$ sur lequel $x$ n'est pas redéfini
- et
  - $x$ est utilisé par la garde de $tr_j$
  - ou $x$ n'est pas défini par l'action de $tr_j$ et $x$ est utilisé par $tr_j$




$m \rightharpoonup i$ est données-dépendant de $j \rightharpoonup m$ et non de $j \rightharpoonup k$

# Dépendance de contrôle (1/2)

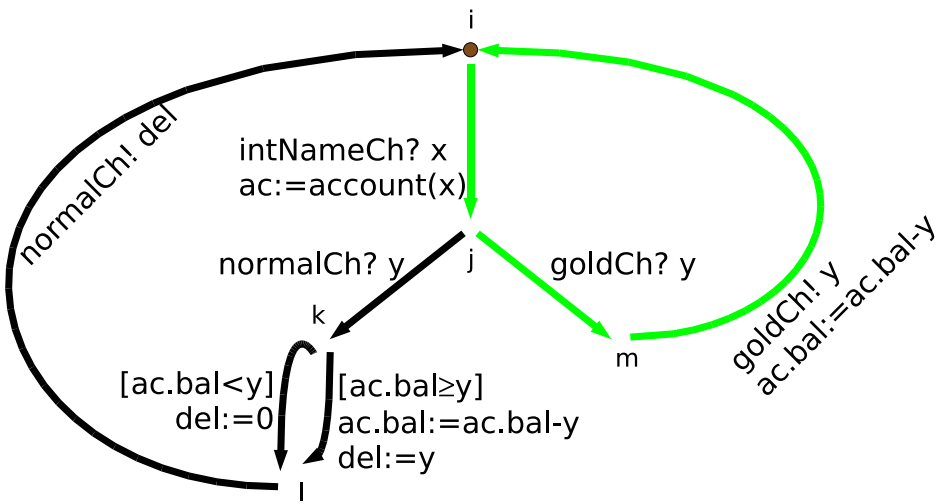
*Une transition détermine potentiellement l'exécution ou non d'une autre transition*

- Requiert la notion de chemin maximal

# Dépendance de contrôle (1/2)

*Une transition détermine potentiellement l'exécution ou non d'une autre transition*

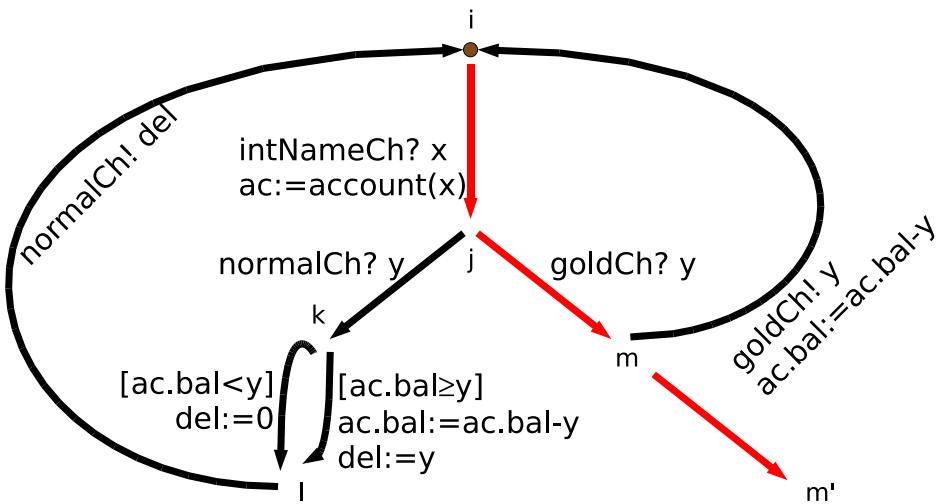
- Requiert la notion de chemin maximal : un chemin qui ne peut être prolongé par aucune transition
  (chemin infini ou chemin sans successeur)



# Dépendance de contrôle (1/2)

*Une transition détermine potentiellement l'exécution ou non d'une autre transition*

- Requiert la notion de chemin maximal : un chemin qui ne peut être prolongé par aucune transition
  (chemin infini ou chemin sans successeur)



# Dépendance de contrôle (2/2)

## Dépendance de contrôle ($tr_i \xrightarrow{cd} tr_j$)

$tr_j$ est contrôle-dépendant de $tr_i$

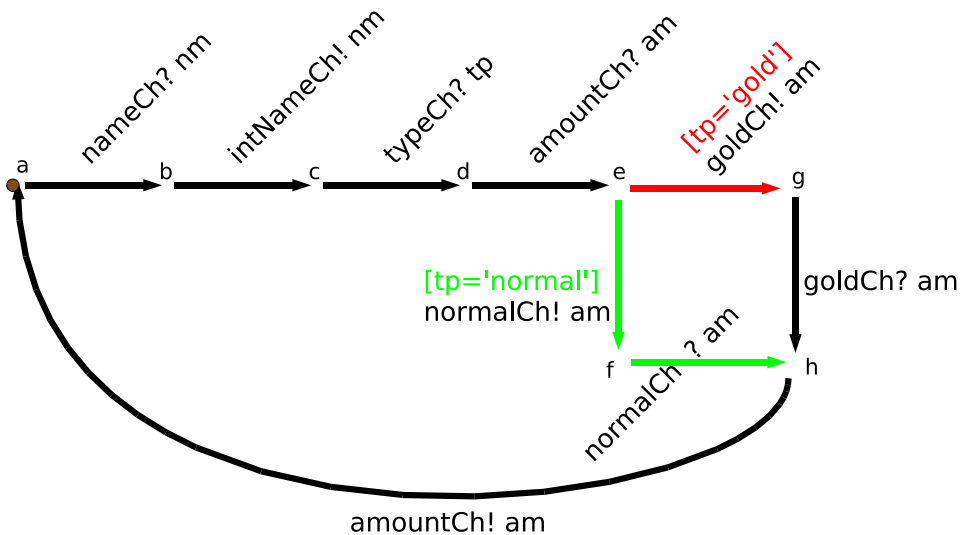
- $tr_i$ a une garde non-triviale
- $source_{tr_j}$ apparaît sur tous les chemins maximaux depuis $tr_i$
- Il existe $tr_k$ tel que $source_{tr_k} = source_{tr_i}$, et un chemin maximal depuis $tr_k$ sur lequel $source_{tr_j}$ n'apparaît pas.

# Dépendance de contrôle (2/2)

## Dépendance de contrôle ($tr_i \xrightarrow{cd} tr_j$)

$tr_j$ est contrôle-dépendant de $tr_i$

- $tr_i$ a une garde non-triviale
- $source_{tr_j}$ apparaît sur tous les chemins maximaux depuis $tr_i$
- Il existe $tr_k$ tel que $source_{tr_k} = source_{tr_i}$, et un chemin maximal depuis $tr_k$ sur lequel $source_{tr_j}$ n'apparaît pas.



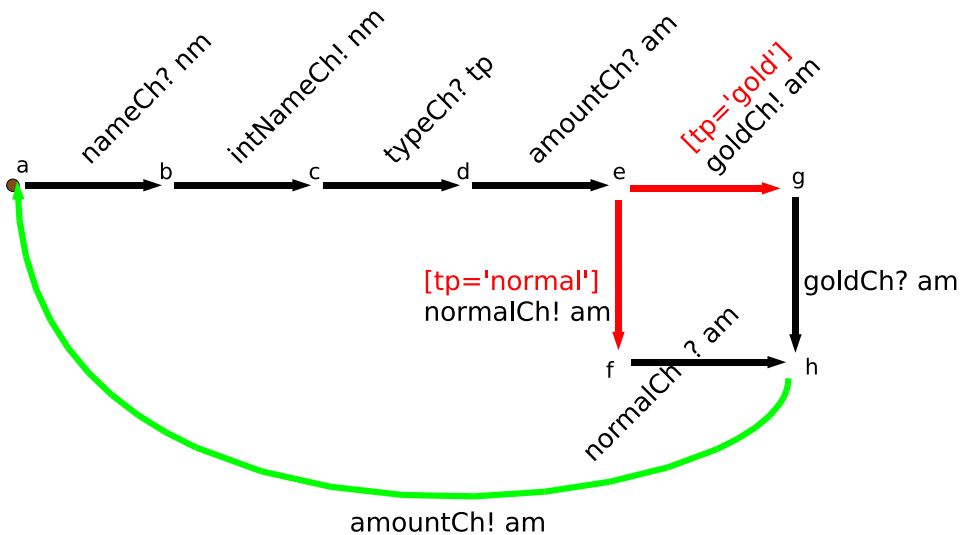

- $f \rightharpoonup h$
  contrôle-dépendant de
  $e \rightharpoonup f$ et non de $e \rightharpoonup g$

# Dépendance de contrôle (2/2)

## Dépendance de contrôle ($tr_i \xrightarrow{cd} tr_j$)

$tr_j$ est contrôle-dépendant de $tr_i$

- $tr_i$ a une garde non-triviale
- $source_{tr_j}$ apparaît sur tous les chemins maximaux depuis $tr_i$
- Il existe $tr_k$ tel que $source_{tr_k} = source_{tr_i}$, et un chemin maximal depuis $tr_k$ sur lequel $source_{tr_j}$ n'apparaît pas.




- $f \rightharpoonup h$ contrôle-dépendant de $e \rightharpoonup f$ et non de $e \rightharpoonup g$
- $h \rightharpoonup a$ n'est contrôle-dépendant ni de $e \rightharpoonup f$ ni de $e \rightharpoonup g$

# Dépendance de communication

*Transfert de données et de contrôle via les communications*

## Dépendance de communication ($tr_i \overset{com}{\leftrightarrow} tr_j$)

$tr_i$ et $tr_j$ mutuellement *communication-dépendants*

- $tr_i$ et $tr_j$ dans des automates distincts
- Il existe un canal $c$ tel que les actions de $tr_i$ et $tr_j$ sont de la forme
  - $c?$ et $c!$
  - ou $c?x$ et $c!t$

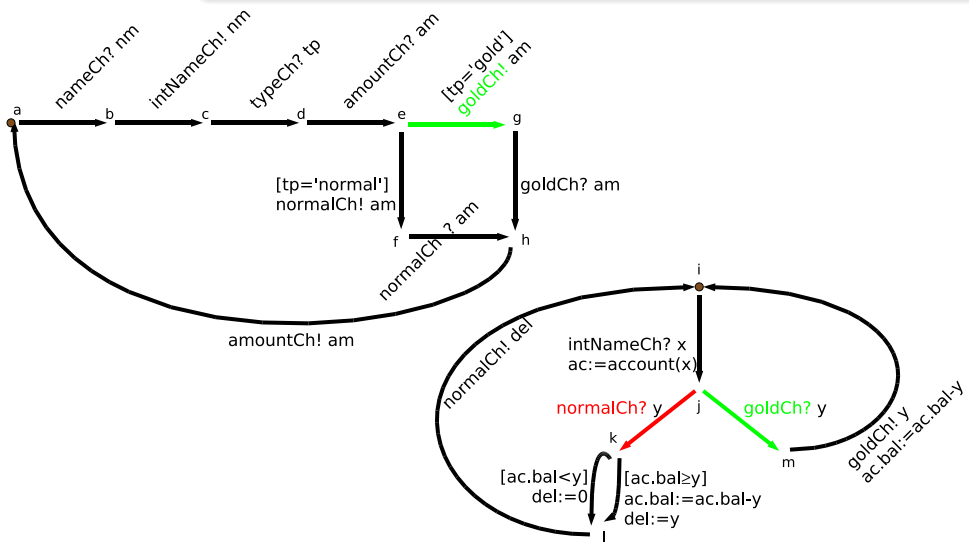
# Dépendance de communication

*Transfert de données et de contrôle via les communications*

## Dépendance de communication ($tr_i \overset{com}{\leftrightarrow} tr_j$)

$tr_i$ et $tr_j$ mutuellement *communication-dépendants*

- $tr_i$ et $tr_j$ dans des automates distincts
- Il existe un canal $c$ tel que les actions de $tr_i$ et $tr_j$ sont de la forme
  - $c?$ et $c!$
  - ou $c?x$ et $c!t$




$e \rightharpoonup g$ et $j \rightharpoonup m$ mutuellement communication-dépendants, mais pas $e \rightharpoonup g$ et $j \rightharpoonup k$

# Plan

# Réduction paramétrée de spécifications

## Tranche d'une spécification

- Une spécification
- Un critère de réduction
  - Transitions choisies dans la spécification

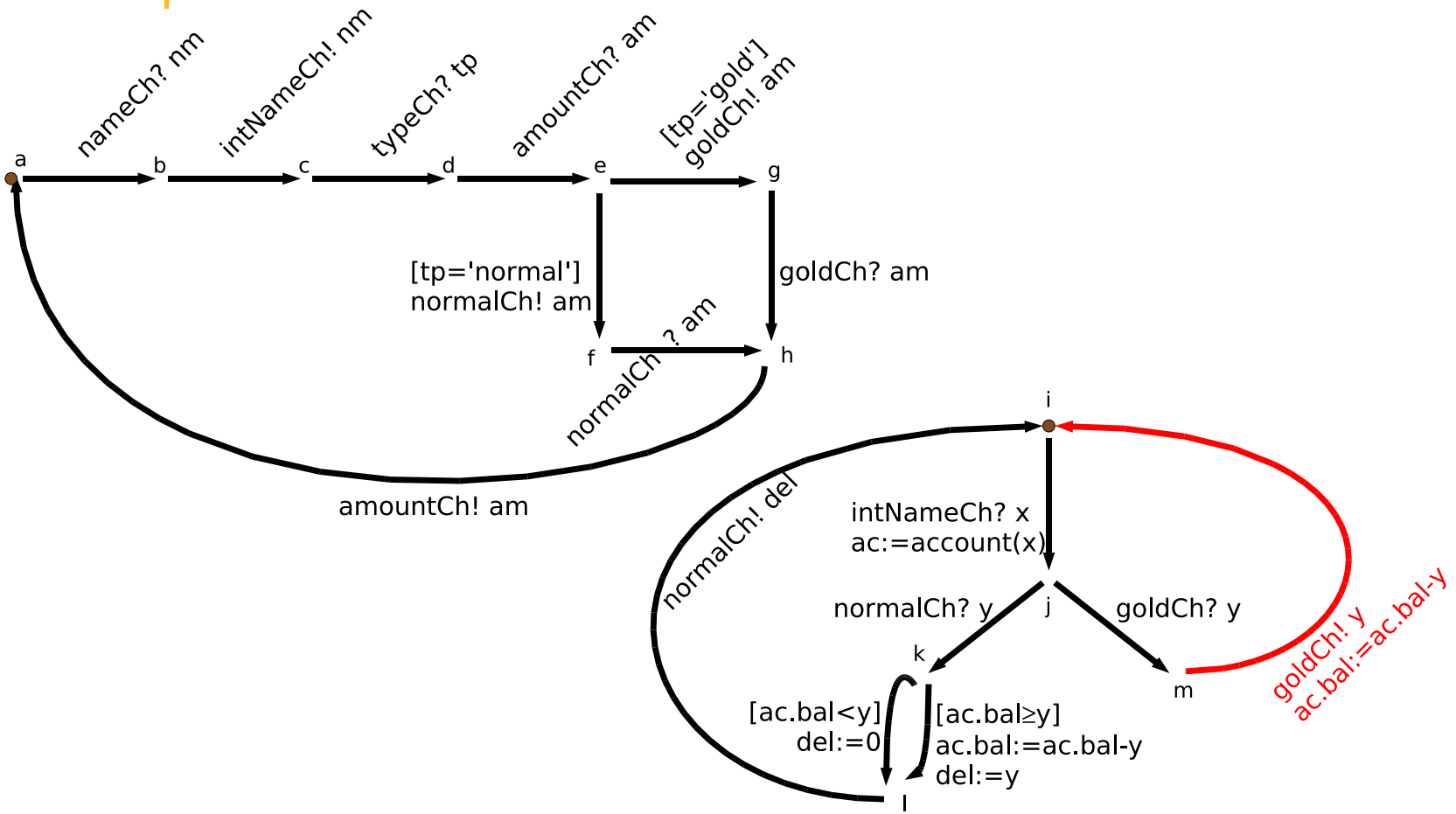
# Réduction paramétrée de spécifications



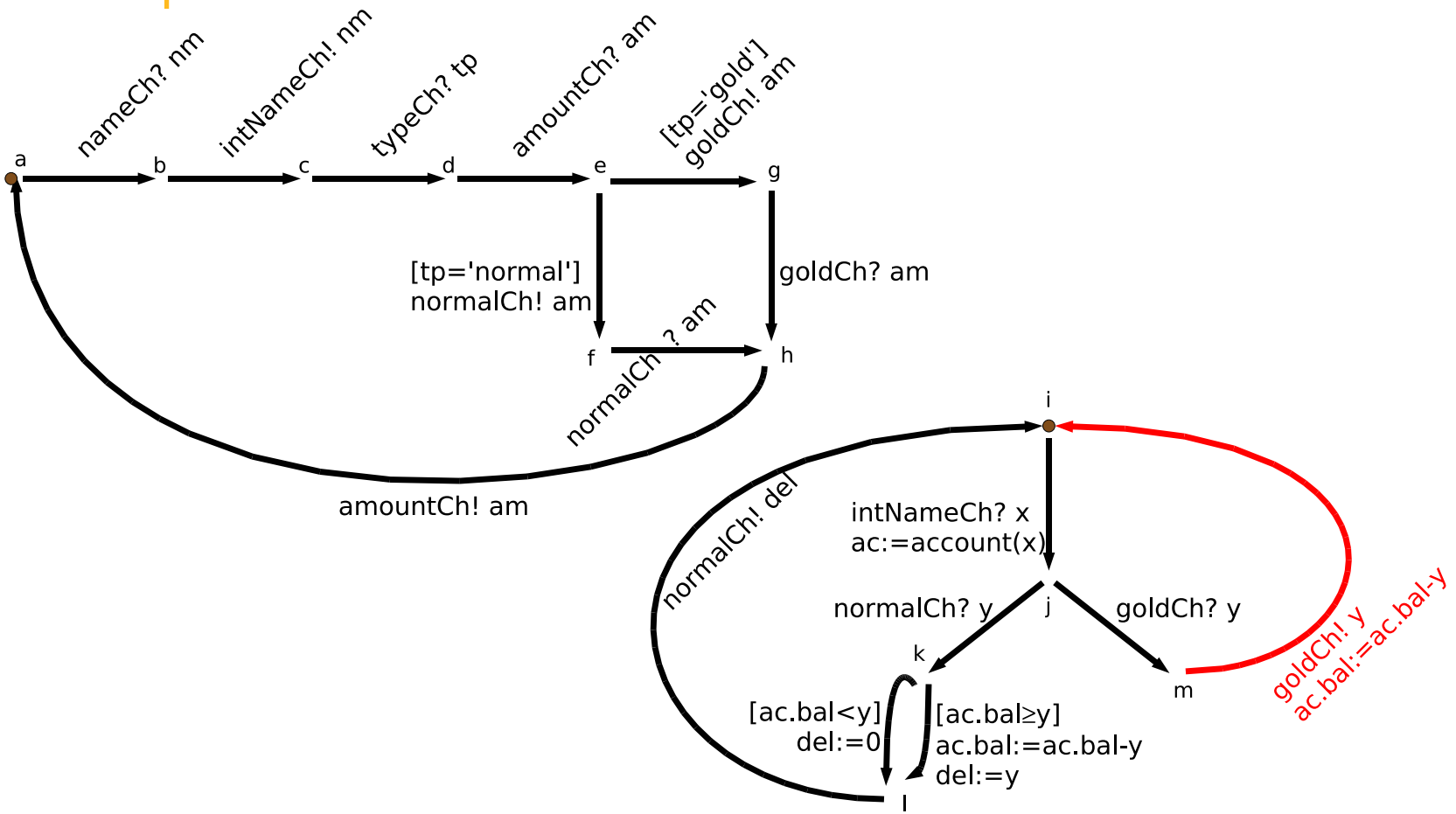
## Tranche d'une spécification

- Une spécification
- Un critère de réduction
  - Transitions choisies dans la spécification
- Une tranche par rapport à un critère
  - Spécification réduite
  - Formée des transitions atteignables depuis le critère via une séquence de dépendances
    - Pas d'occurrence de plusieurs dépendances de communications consécutives

# Exemple

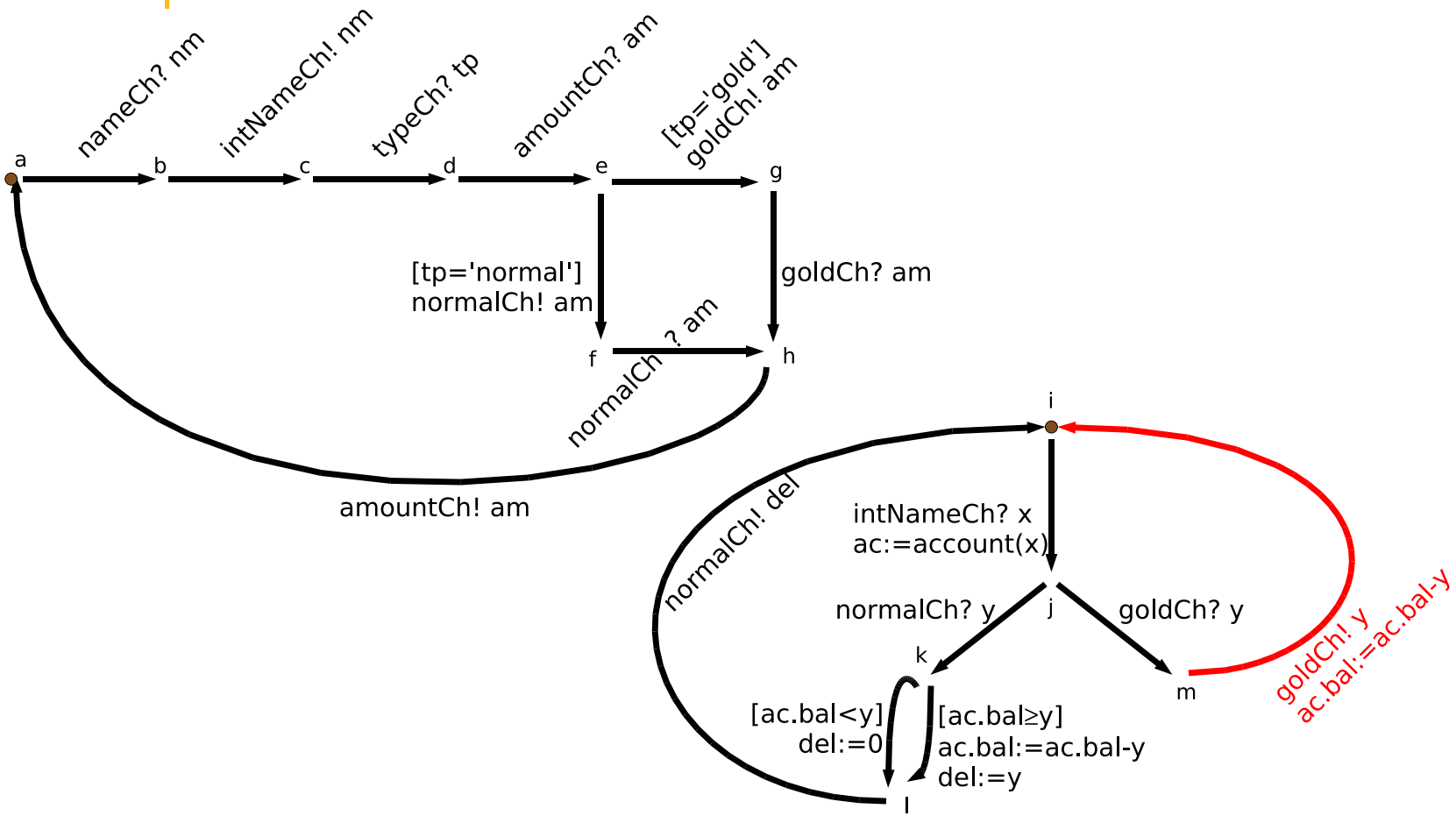

a
b
c
d
e
f
g
h
i
j
k
l
m
nameCh? nm
intNameCh! nm
typeCh? tp
amountCh? am
[tp='gold']
goldCh! am
[tp='normal']
normalCh! am
goldCh? am
normalCh ? am
amountCh! am
normalCh! del
intNameCh? x
ac:=account(x)
normalCh? y
goldCh? y
goldCh! y
ac.bal:=ac.bal-y
[ac.bal<y]
del:=0
[ac.bal≥y]
ac.bal:=ac.bal-y
del:=y

# Exemple



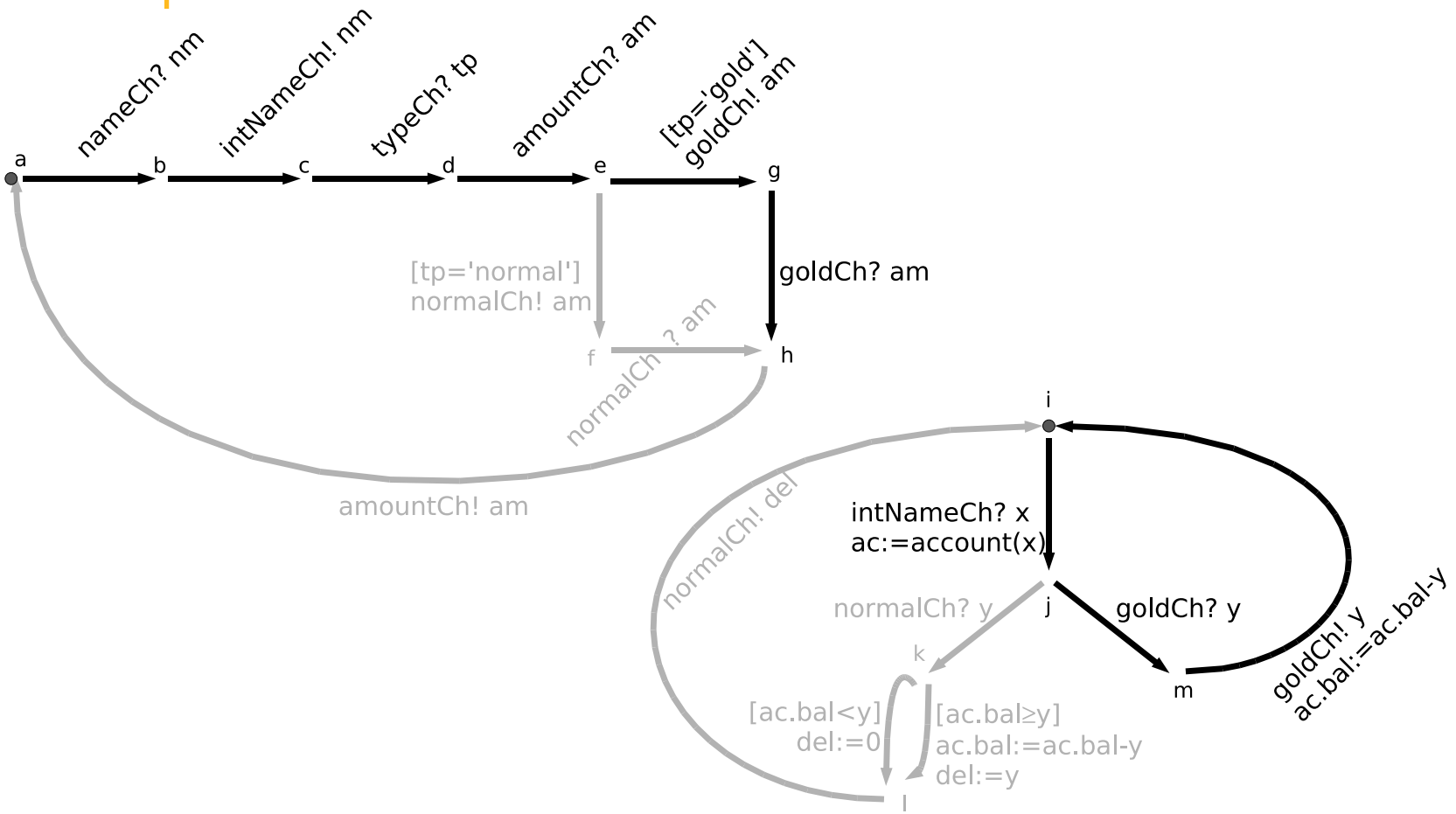

$(m \rightharpoonup i)$

# Exemple




$$(m \rightarrowtail i)\left\{\begin{array}{l} \xleftarrow{dd}(i \rightarrowtail j) \\ \xleftarrow{com}(g \rightarrowtail h) \\ \xleftarrow{dd}(j \rightarrowtail m) \end{array}\right. \quad \xleftarrow{com}(b \rightarrowtail c) \quad \xleftarrow{dd}(a \rightarrowtail b) \quad \xleftarrow{cd}(e \rightarrowtail g)\left\{\begin{array}{l} \xleftarrow{dd}(c \rightarrowtail d) \\ \xleftarrow{dd}(d \rightarrowtail e) \end{array}\right.$$

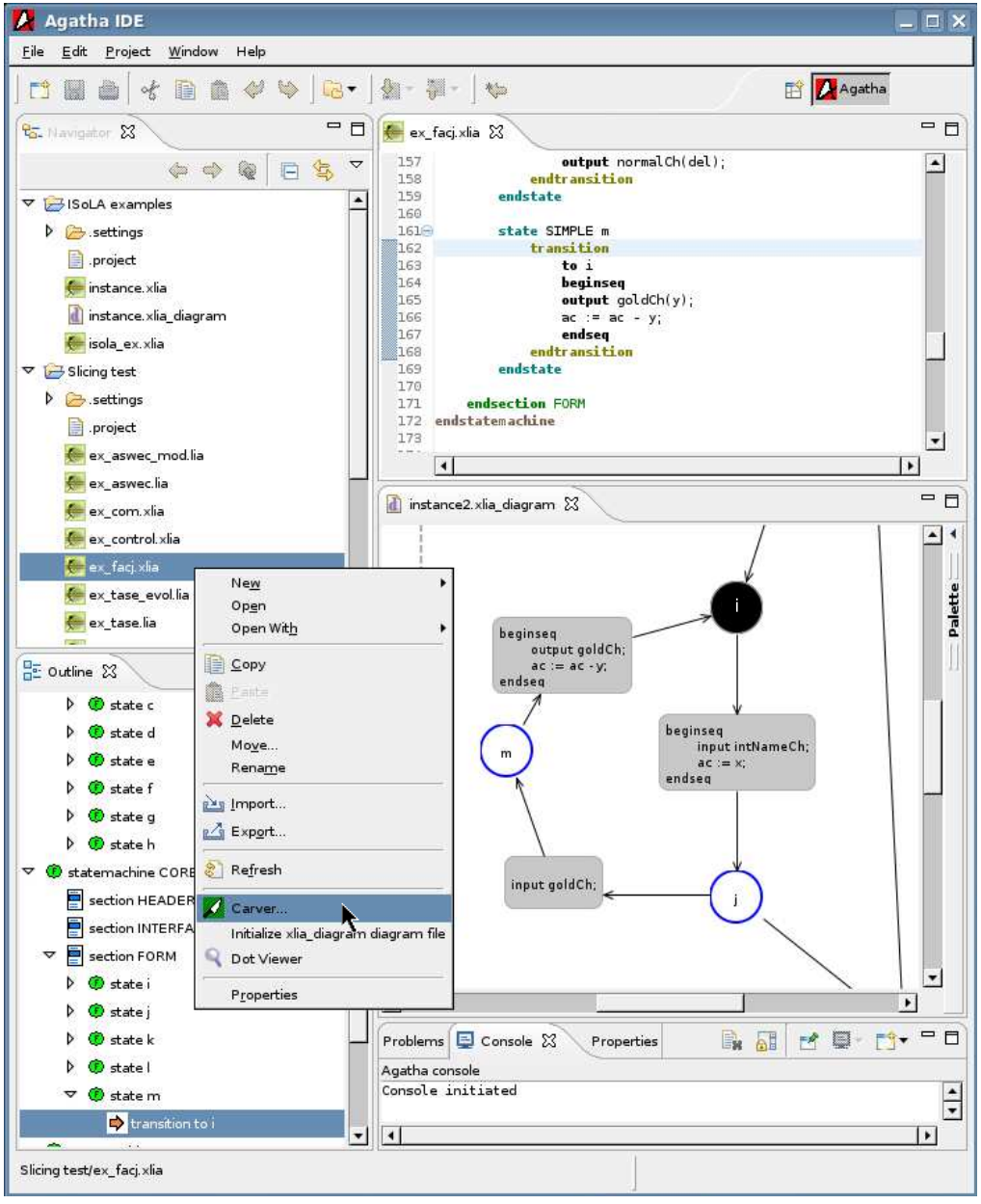
# Exemple



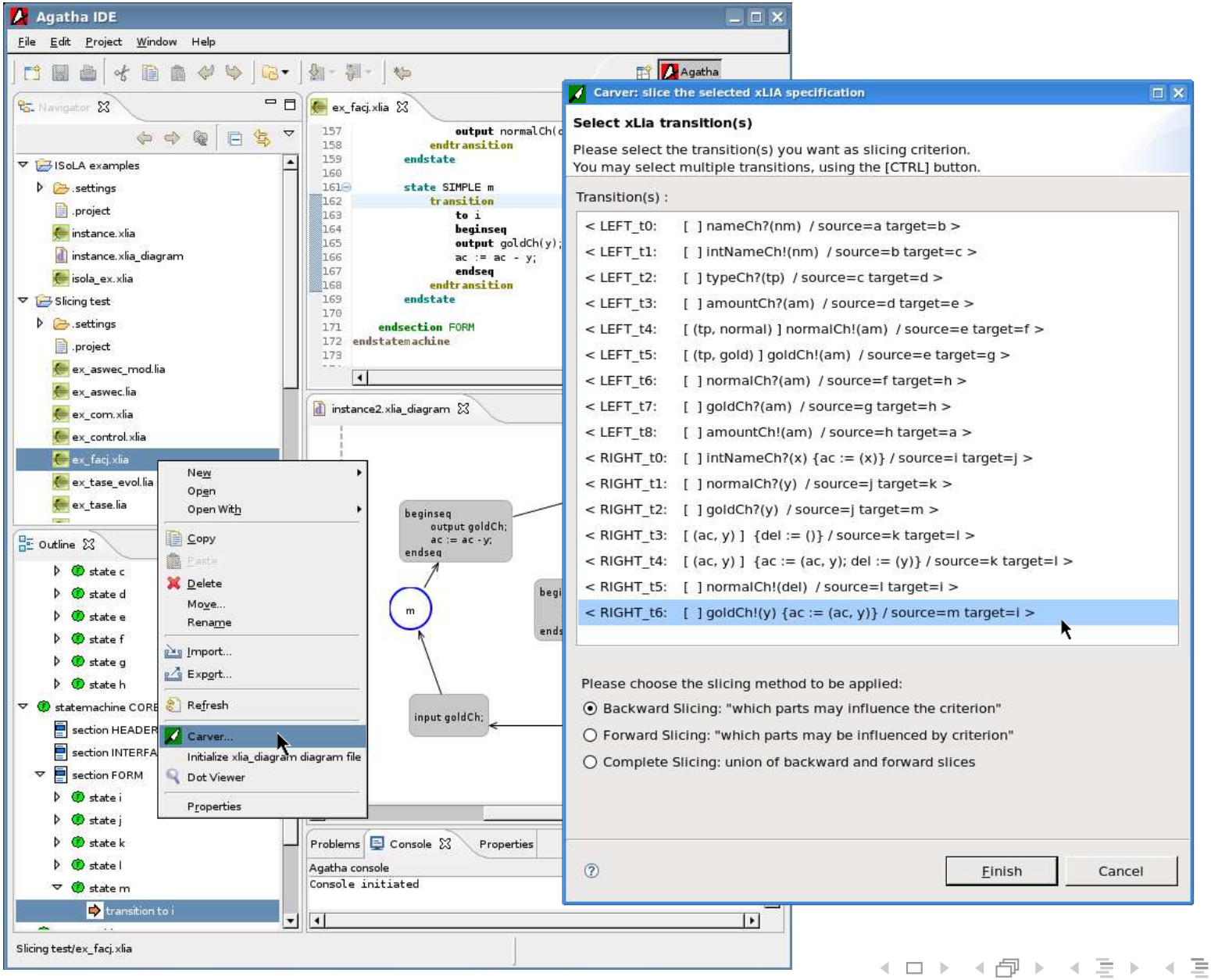

$$(m \rightharpoonup i) \begin{cases} \xleftarrow{dd} (i \rightharpoonup j) \\ \xleftarrow{com} (g \rightharpoonup h) \\ \xleftarrow{dd} (j \rightharpoonup m) \end{cases} \quad \xleftarrow{com} (b \rightharpoonup c) \quad \xleftarrow{dd} (a \rightharpoonup b)$$

$$\xleftarrow{cd} (e \rightharpoonup g) \begin{cases} \xleftarrow{dd} (c \rightharpoonup d) \\ \xleftarrow{dd} (d \rightharpoonup e) \end{cases}$$

# Mise en œuvre : Carver



# Mise en œuvre : Carver



# Plan

# Principales contributions

1. Un cadre théorique pour les analyses statiques de spécifications formées d'automates communicants

# Principales contributions

1. Un cadre théorique pour les analyses statiques de spécifications formées d'automates communicants
2. La définition formelle des relations de dépendances qui existent dans les spécifications formées d'automates communicants

# Principales contributions

1. Un cadre théorique pour les analyses statiques de spécifications formées d'automates communicants
2. La définition formelle des relations de dépendances qui existent dans les spécifications formées d'automates communicants
3. La définition formelle d'une tranche de spécification formée d'automates communicants, par rapport à un critère

# Principales contributions

1. Un cadre théorique pour les analyses statiques de spécifications formées d'automates communicants
2. La définition formelle des relations de dépendances qui existent dans les spécifications formées d'automates communicants
3. La définition formelle d'une tranche de spécification formée d'automates communicants, par rapport à un critère
4. La mise au point d'algorithmes efficaces pour calculer les relations de dépendances et les tranches, ainsi que les démonstrations de la complexité polynomiale, de la correction et complétude par rapport aux définitions correspondantes, de ces algorithmes

# Principales contributions

1. Un cadre théorique pour les analyses statiques de spécifications formées d'automates communicants
2. La définition formelle des relations de dépendances qui existent dans les spécifications formées d'automates communicants
3. La définition formelle d'une tranche de spécification formée d'automates communicants, par rapport à un critère
4. La mise au point d'algorithmes efficaces pour calculer les relations de dépendances et les tranches, ainsi que les démonstrations de la complexité polynomiale, de la correction et complétude par rapport aux définitions correspondantes, de ces algorithmes
5. La mise en œuvre des algorithmes dans l'outil Carver, pour la réduction paramétrée de spécifications formées d'automates communicants

# Applications

- Preuve de propriétés
  - Propriété traduite en automate, intégré dans le modèle
  - Sous-spécification “utile” pour prouver la propriété

# Applications

- Preuve de propriétés
  - Propriété traduite en automate, intégré dans le modèle
  - Sous-spécification “utile” pour prouver la propriété
- Modèles paramétrés
  - Déterminer certains paramètres inconnus en fonction de propriétés à vérifier
  - Traduction des propriétés en automates [Mateus et al., 2007]

# Applications

- Preuve de propriétés
  - Propriété traduite en automate, intégré dans le modèle
  - Sous-spécification “utile” pour prouver la propriété
- Modèles paramétrés
  - Déterminer certains paramètres inconnus en fonction de propriétés à vérifier
  - Traduction des propriétés en automates [Mateus et al., 2007]
- Test de conformité [Gaston et al., TestCom'06]
  - Objectif de test vu comme une propriété à tester

# Applications

- Preuve de propriétés
  - Propriété traduite en automate, intégré dans le modèle
  - Sous-spécification "utile" pour prouver la propriété
- Modèles paramétrés
  - Déterminer certains paramètres inconnus en fonction de propriétés à vérifier
  - Traduction des propriétés en automates [Mateus et al., 2007]
- Test de conformité [Gaston et al., TestCom'06]
  - Objectif de test vu comme une propriété à tester
- Gestion des évolutions
  - Exemple : famille de produits
  - Isoler la partie stable de la spécification
  - Élements de la partie stable utiles aux évolutions ?
  - Critère : ensemble des transitions qui peuvent communiquer avec la partie qui évolue
  - Gain potentiel : test de la partie stable

# Perspectives

- Transitions non-atomiques
  - Mettre en relation des parties de transitions (garde, action de communication, affectation)
  - Calculer des tranches plus petites

# Perspectives

- Transitions non-atomiques
  - Mettre en relation des parties de transitions (garde, action de communication, affectation)
  - Calculer des tranches plus petites
- Augmenter la précision des dépendances de communication
  - Dépendance seulement si il existe une exécution où les deux transitions sont exécutées en parallèle
  - Surcoût en complexité

# Perspectives

- Transitions non-atomiques
  - Mettre en relation des parties de transitions (garde, action de communication, affectation)
  - Calculer des tranches plus petites
- Augmenter la précision des dépendances de communication
  - Dépendance seulement si il existe une exécution où les deux transitions sont exécutées en parallèle
  - Surcoût en complexité
- Variables partagées
  - État de l'art (programmes) : algorithmes exponentiels au pire
    - Dépendances d'interférence [Thèse Krinke, 2003]

# Merci pour votre attention

**Questions ?**

# Complexité des algorithmes

- Calcul des dépendances de données

$$O(\sum_{tr \in T} d_{tr}.|T|.|\mathcal{D}_{\mathscr{A}}|.lg(|\mathcal{D}_{\mathscr{A}}|))$$

- Calcul des dépendances de contrôle

$$O(\sum_{tr_c \in conds_{\mathscr{A}}} d_{tr_c}.|conds_{\mathscr{A}}|^2.|T|.lg(|T|))$$

- Calcul des dépendances de communication

$$O(|C|.|T|)$$

- Réduction paramétrée
  - 1er calcul

$$O(\sum_{tr \in T} d_{tr}.|T|^3.lg(|T|))$$

  - Calculs suivants

$$O(|T|.lg(|T|))$$

# Principales publications, soumissions

- Sébastien Labbé, Jean-Pierre Gallois and Marc Pouzet. Slicing communicating automata specifications for efficient model reduction. In *Australian Software Engineering Conference (ASWEC)*, pages 191–200, 2007.
- Sébastien Labbé and Jean-Pierre Gallois. Towards slicing communicating extended automata. In *International Symposium on Formal Methods (FM), Doctoral Symposium*, 2006.
  `http ://fm06.mcmaster.ca/01SebastienLabbe.pdf`
- Sébastien Labbé, Jean-Pierre Gallois and Marc Pouzet. Efficient reduction of communicating automata specifications using slicing techniques. *Journal of Software (JSW)*, 2007. Under review (submitted Sept'07).
- Sébastien Labbé and Jean-Pierre Gallois. Slicing communicating automata specifications : polynomial algorithms for model reduction. *Formal Aspects of Computing (FACJ)*, 2007. Under review (submitted Jan'07).
- Sébastien Labbé and Arnault Lapitre. Carver : a slicing tool for communicating automata specifications. In *Post-proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods (ISoLA)*, 2006. Under review (submitted Jan'07).