



# Extraction de contrats 42 à partir de programmes SystemC

Pierre-Yves Delahaye


Mardi 25 mai 2010

## **Tuteurs**

Matthieu Moy  
Kévin Marquet

# Plan de la soutenance

- Introduction
  - Problématique
  - Une réponse : SystemC
  - Le formalisme 42
  - Contribution du TER
- Algorithmes d'extraction de contrats 42
  - Traitement des `wait` et `notify`
  - Traitement des branchements
  - Exemple de contrat généré par l'outil développé
- Implémentation
  - La chaîne d'extraction
  - La forme intermédiaire
  - Comment fonctionne réellement le backend ?
- Conclusion



**Introduction**

Algorithmes  
d'extraction  
de contrats 42

Implémentation

Conclusion

# INTRODUCTION

# Problématique

## Les besoins de l'industrie :

- Réduire le **coût** et le **temps** de développement des systèmes embarqués
- Faire face à l'augmentation de la **complexité** de ces systèmes

**Introduction**  
Problématique  
Une réponse :  
SystemC  
Le formalisme 42  
Contribution du  
TER

**Algorithmes**  
d'extraction  
de contrats 42

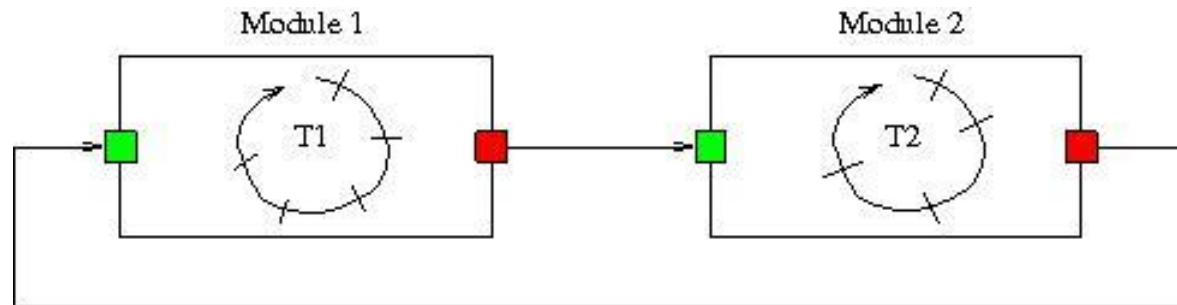
**Implémentation**

**Conclusion**

# Une réponse : SystemC

## Qu'est-ce que SystemC :

- Une **bibliothèque C++**
- Permet une modélisation **modulaire** de **haut niveau** des systèmes matériels



Deux modules SystemC qui communiquent

# Pourquoi utiliser SystemC :

		Propriétés de SystemC	
		Haut niveau d'abstraction	Développement modulaire
Les avantages		Tests du matériel possibles très tôt	Travail en équipe facilité
		Développement du logiciel en parallèle	Réutilisation possible de modules déjà conçus

SystemC s'est donc imposé comme un standard

## Introduction

Problématique  
Une réponse :  
SystemC  
Le formalisme 42  
Contribution du  
TER

Algorithmes  
d'extraction  
de contrats 42

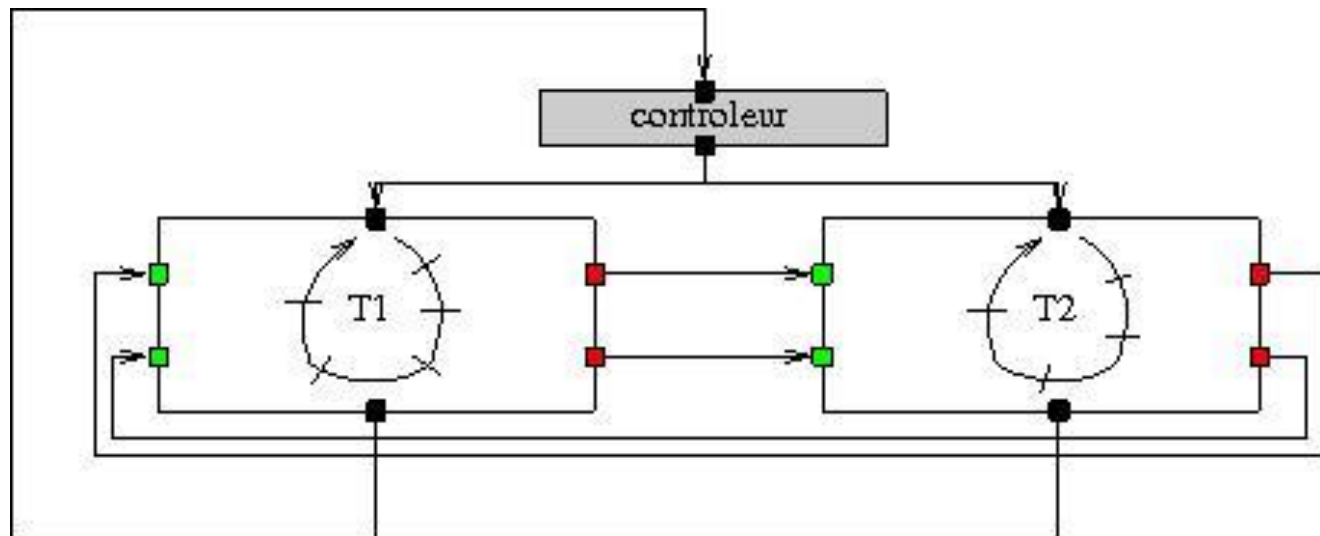
Implémentation

Conclusion

# Le formalisme 42 (T. Bouhadiba, F. Maraninchi)

## Modélisation 42 d'un système

- Des composants (correspondent aux modules SystemC)
- Des contrats (un pour chaque composant)
- Un contrôleur



Un exemple de modélisation 42

### Introduction

Problématique  
Une réponse :  
SystemC  
Le formalisme 42  
Contribution du  
TER

Algorithmes  
d'extraction  
de contrats 42

Implémentation

Conclusion

# Les contrats 42

## Un besoin :

Pouvoir réutiliser des modules SystemC, et s'assurer qu'ils se synchroniseront correctement avec le reste du système

## Ce qu'apportent les contrats 42 :

Focalisation sur les entrées/sorties des composants

⇒ un contrat 42 est un modèle comportemental d'un composant, vis-à-vis de l'extérieur

⇒ on a le « mode d'emploi » d'un composant

Introduction

Problématique

Une réponse :

SystemC

Le formalisme 42

Contribution du

TER

Algorithmes

d'extraction

de contrats 42

Implémentation

Conclusion



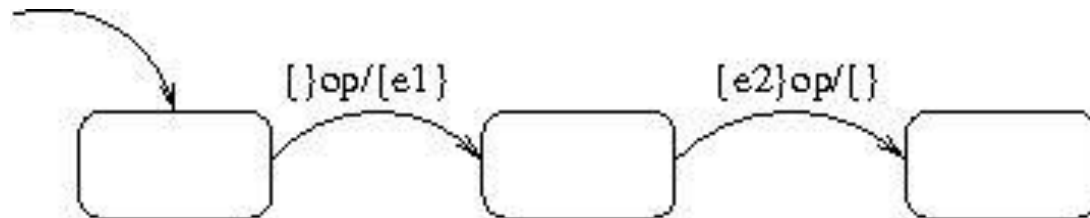
# Un exemple de contrat 42

## Un graphe avec :

- des états
- des transitions

## Une transition :

```
{data_req} control_inputs / control_outputs {data_prod}
```



# Contribution du TER

- **Ce qu'on avait avant le TER :**
  - La possibilité de concevoir des systèmes constitués de modules SystemC
  - La possibilité de modéliser le comportement au niveau entrées/sorties de ces modules SystemC grâce aux contrats 42
- **Ce qui manquait et qui a été fait :**

Algorithmes d'extraction de contrats 42 à partir de modules SystemC



Introduction

**Algorithmes  
d'extraction  
de contrats 42**

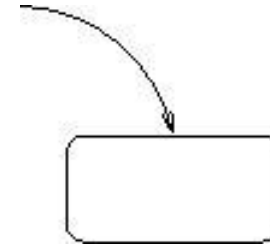
Implémentation

Conclusion

# **ALGORITHMES D'EXTRACTION DE CONTRATS 42**

# Traitement des `wait` et `notify`

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    e2.notify();  
    wait(e1);  
    wait(e2);  
    a++;  
}
```



Événements attendus	Événements notifiés

Introduction

Algorithmes  
d'extraction  
de contrats 42

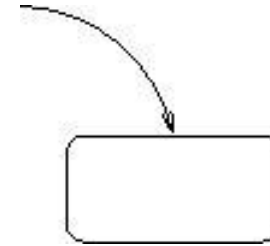
Traitement des  
`wait` et `notify`  
Traitement des  
branchements  
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

# Traitement des `wait` et `notify`

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    e2.notify();  
    wait(e1);  
    wait(e2);  
    a++;  
}
```



Événements attendus	Événements notifiés
	<code>e1</code>

Introduction

Algorithmes  
d'extraction  
de contrats 42

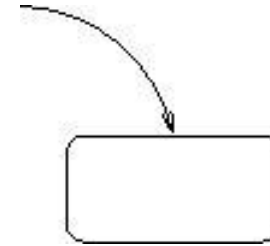
Traitement des  
`wait` et `notify`  
Traitement des  
branchements  
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

# Traitement des `wait` et `notify`

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    e2.notify();  
    wait(e1);  
    wait(e2);  
    a++;  
}
```



Événements attendus	Événements notifiés
	e1
	e2

Introduction

Algorithmes  
d'extraction  
de contrats 42

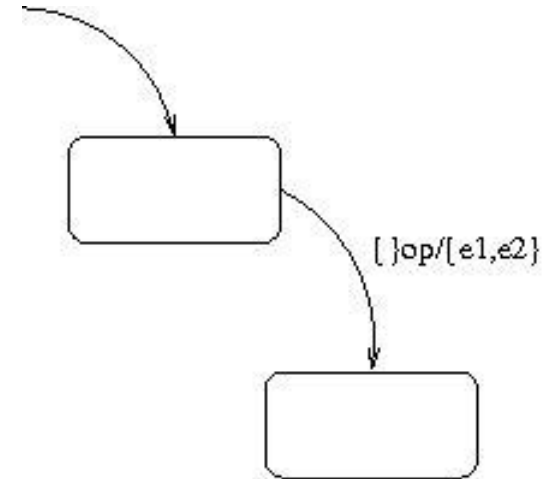
Traitement des  
`wait` et `notify`  
Traitement des  
branchements  
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

# Traitement des `wait` et `notify`

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    e2.notify();  
    wait(e1);  
    wait(e2);  
    a++;  
}
```



Événements attendus	Événements notifiés
<code>e1</code>	

Introduction

Algorithmes  
d'extraction  
de contrats 42

Traitement des  
`wait` et `notify`

Traitement des  
branchements

Exemple de  
contrat généré  
par l'outil  
développé

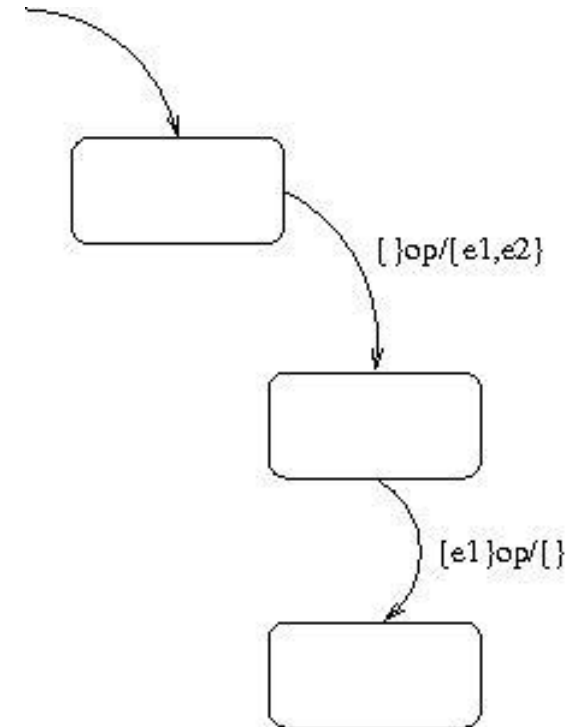
Implémentation

Conclusion

# Traitement des `wait` et `notify`

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    e2.notify();  
    wait(e1);  
    wait(e2);  
    a++;  
}
```

Événements attendus	Événements notifiés
<code>e2</code>	



Introduction

Algorithmes  
d'extraction  
de contrats 42

Traitement des  
`wait` et `notify`  
Traitement des  
branchements  
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

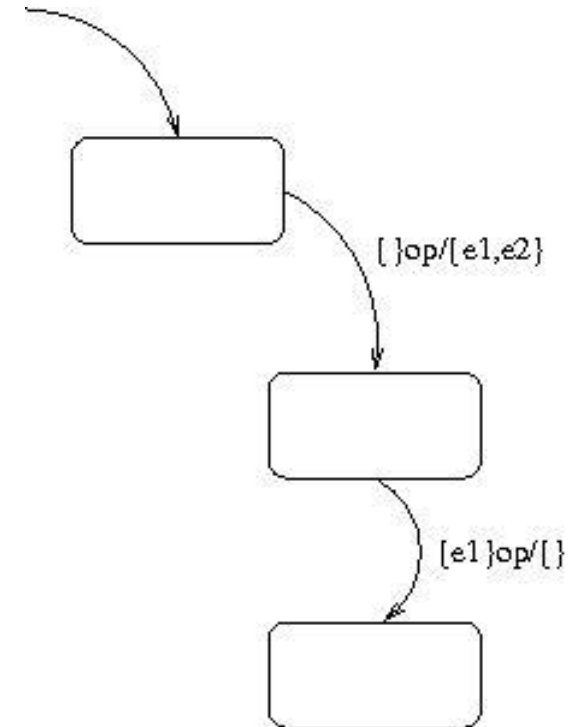
Conclusion



# Traitement des `wait` et `notify`

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    e2.notify();  
    wait(e1);  
    wait(e2);  
    a++;  
}
```

Événements attendus	Événements notifiés
<code>e2</code>	



Introduction

Algorithmes  
d'extraction  
de contrats 42

Traitement des  
`wait` et `notify`

Traitement des  
branchements

Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

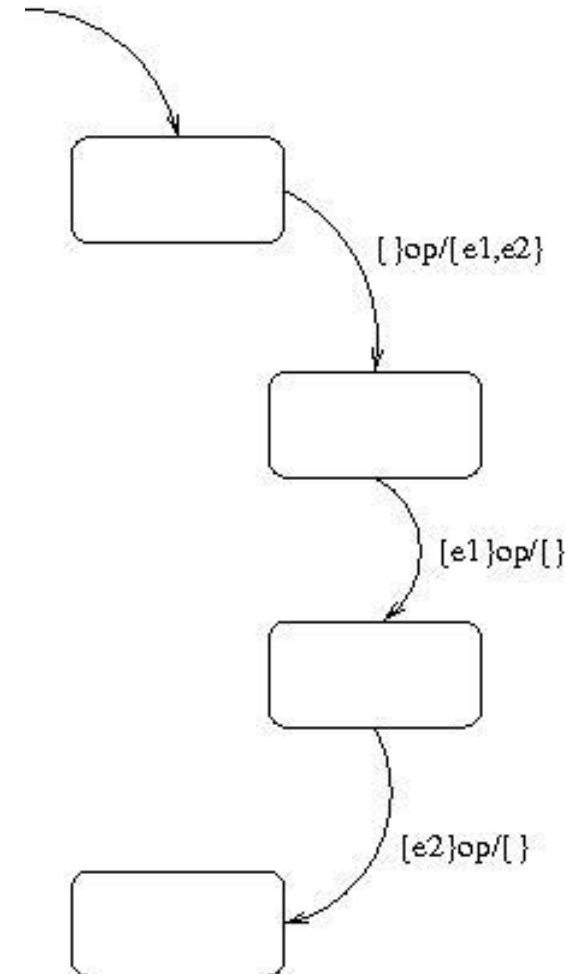
Conclusion

# Traitement des `wait` et `notify`

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    e2.notify();  
    wait(e1);  
    wait(e2);  
    a++;  
}
```

Return Inst

Evénements attendus	Evénements notifiés



Introduction

Algorithmes  
d'extraction  
de contrats 42

Traitement des  
`wait` et `notify`  
Traitement des  
branchements  
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

# Traitement des branchements

Une difficulté est apparue lorsque le programme SystemC comporte plusieurs blocs `if..then..else` consécutifs.

Il y a **deux stratégies** possibles dans ce cas là :

- l'une conduit à une **explosion** du nombre d'états du contrat 42
- l'autre augmente la **granularité** par rapport au programme SystemC

Introduction

Algorithmes  
d'extraction  
de contrats 42

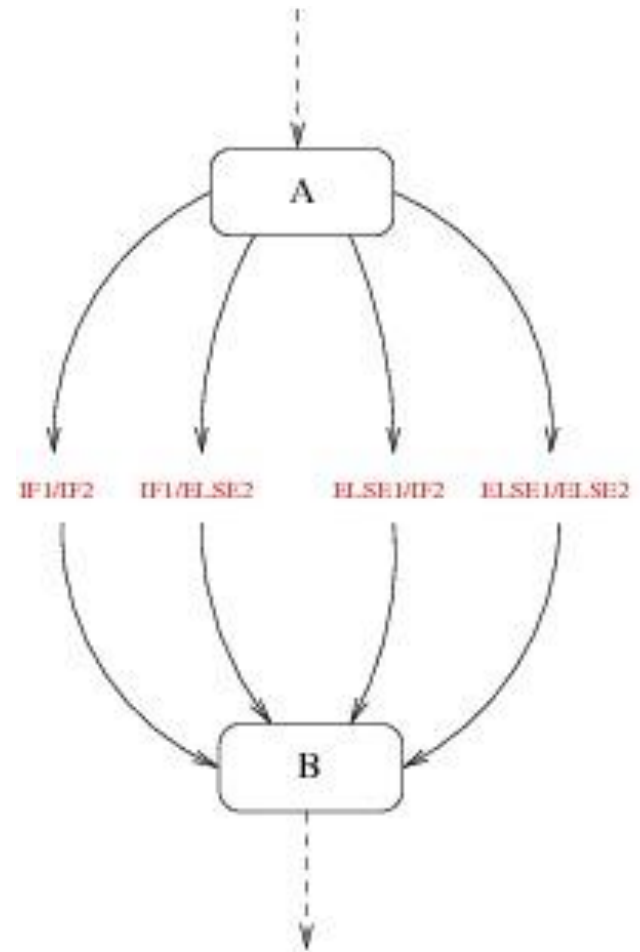
Traitement des  
`wait` et `notify`  
Traitement des  
branchements  
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

# Explosion du nombre d'états :

```
void module1::T1 () {  
    int a=0;  
    int b=0;  
  
    if(a==0) {  
        ...           // IF1  
    }  
    else{  
        ...           // ELSE1  
    }  
  
    if (b==0) {  
        ...           // IF2  
    }  
    else{  
        ...           // ELSE2  
    }  
  
    ...  
  
}
```



$$n_{\text{explosion}} = n_{\text{wait}} \times 2^{n-1}$$

Introduction

Algorithmes  
d'extraction  
de contrats 42

Traitement des  
wait et notify

Traitement des  
branchements

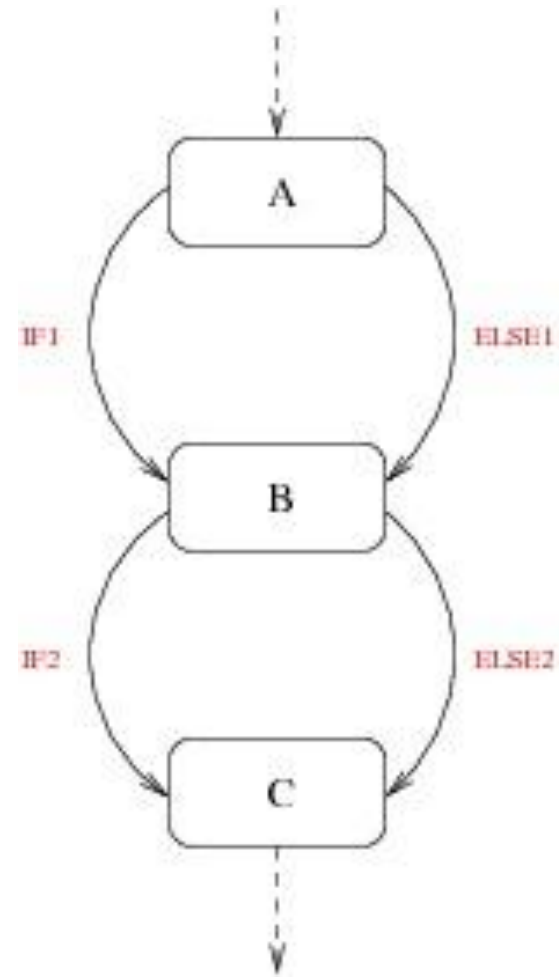
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

# Augmentation de la granularité :

```
void module1::T1 () {  
    int a=0;  
    int b=0;  
  
    if(a==0) {  
        ...           // IF1  
    }  
    else{  
        ...           // ELSE1  
    }  
  
    if (b==0) {  
        ...           // IF2  
    }  
    else{  
        ...           // ELSE2  
    }  
  
    ...  
}
```



$$n_{\text{granularité}} = n + n_{\text{wait}}$$

Introduction

Algorithmes  
d'extraction  
de contrats 42

Traitement des  
wait et notify

Traitement des  
branchements

Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

# Quelle stratégie choisir ?

Les deux stratégies sont intéressantes.  
On choisit d'implémenter la deuxième.

Introduction

**Algorithmes  
d'extraction  
de contrats 42**

Traitement des  
wait et notify

Traitement des  
branchements

Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

Introduction

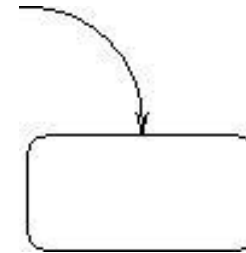
Algorithmes  
d'extraction  
de contrats 42

Traitement des  
wait et notify  
Traitement des  
branchements  
Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    if(a==0){  
        wait(e2);  
    }  
    else{  
        e2.notify();  
    }  
    wait(e2);  
    a++;  
}
```



Evénements attendus	Evénements notifiés

Introduction

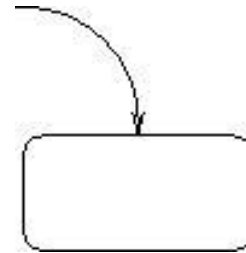
Algorithmes d'extraction de contrats 42

Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

Conclusion

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    if(a==0){  
        wait(e2);  
    }  
    else{  
        e2.notify();  
    }  
    wait(e2);  
    a++;  
}
```



Evénements attendus	Evénements notifiés
	e1



Introduction

Algorithmes d'extraction de contrats 42

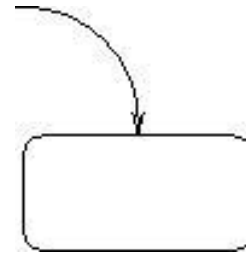
Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

Conclusion

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    if(a==0) {  
        wait(e2);  
    }  
    else {  
        e2.notify();  
    }  
    wait(e2);  
    a++;  
}
```

← Branchement conditionnel



Evénements attendus	Evénements notifiés
	e1

Introduction

Algorithmes d'extraction de contrats 42

Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

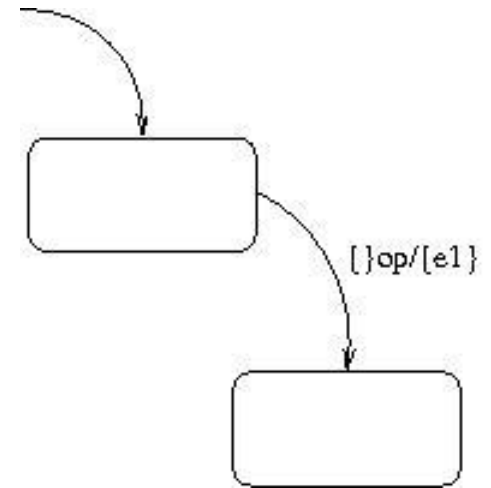
Implémentation

Conclusion

```

void module1::T1() {
  int a=0;
  e1.notify();
  if(a==0){
    wait(e2);
  }
  else{
    e2.notify();
  }
  wait(e2);
  a++;
}

```



Evénements attendus	Evénements notifiés
e2	

Introduction

Algorithmes d'extraction de contrats 42

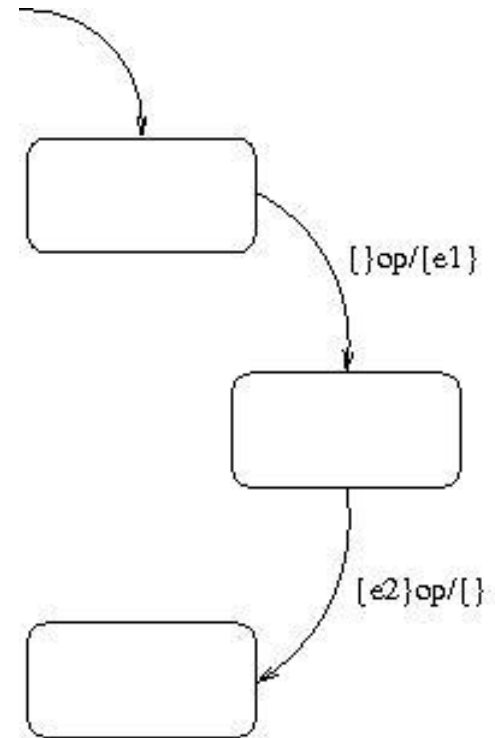
Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

Conclusion

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    if(a==0){  
        wait(e2);  
    }  
    else{  
        e2.notify();  
    }  
    wait(e2);  
    a++;  
}
```

Branchement non conditionnel



Evénements attendus	Evénements notifiés

Introduction

Algorithmes d'extraction de contrats 42

Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

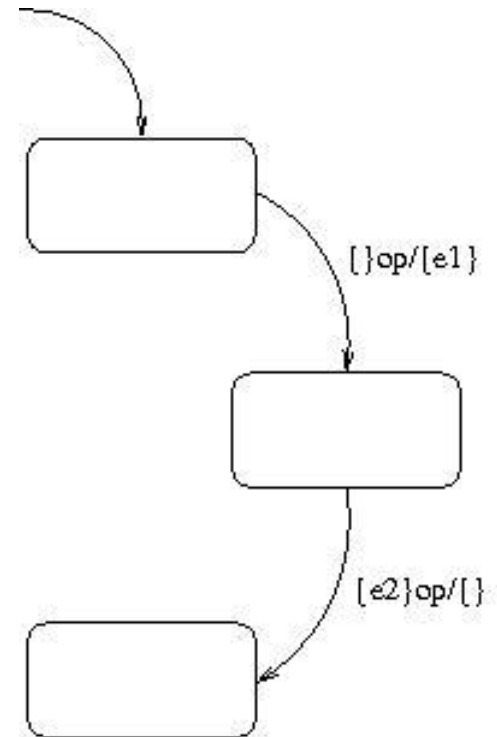
Conclusion

```

void module1::T1() {
    int a=0;
    e1.notify();
    if(a==0){
        wait(e2);
    }
    else{
        e2.notify();
    }
    wait(e2);
    a++;
}

```

Evénements attendus	Evénements notifiés
e2	



Introduction

Algorithmes d'extraction de contrats 42

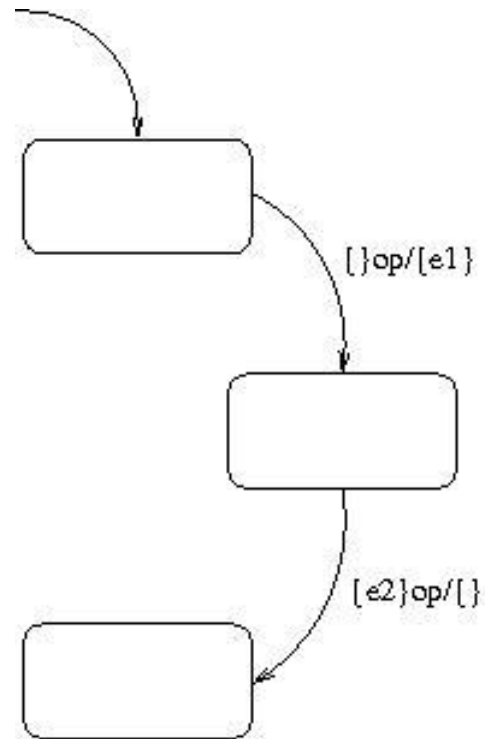
Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

Conclusion

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    if(a==0){  
        wait(e2);  
    }  
    else{  
        e2.notify();  
    }  
    wait(e2);  
    a++;  
}
```

Evénements attendus	Evénements notifiés
e2	



Introduction

Algorithmes d'extraction de contrats 42

Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

Conclusion

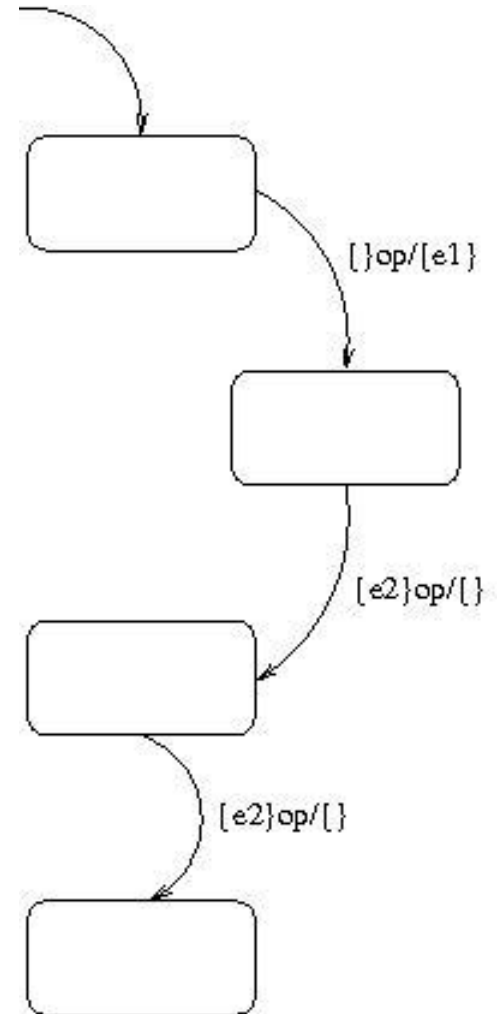
```

void module1::T1() {
  int a=0;
  e1.notify();
  if(a==0){
    wait(e2);
  }
  else{
    e2.notify();
  }
  wait(e2);
  a++;
}

```

← Return Inst

Evénements attendus	Evénements notifiés



Introduction

Algorithmes d'extraction de contrats 42

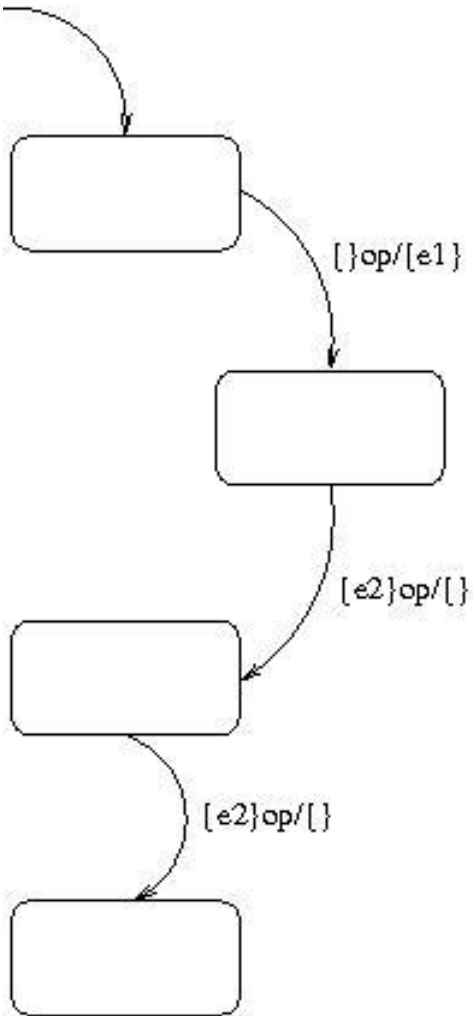
Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

Conclusion

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    if(a==0){  
        wait(e2);  
    }  
    else{  
        e2.notify();  
    }  
    wait(e2);  
    a++;  
}
```

Evénements attendus	Evénements notifiés
	e2



Introduction

Algorithmes d'extraction de contrats 42

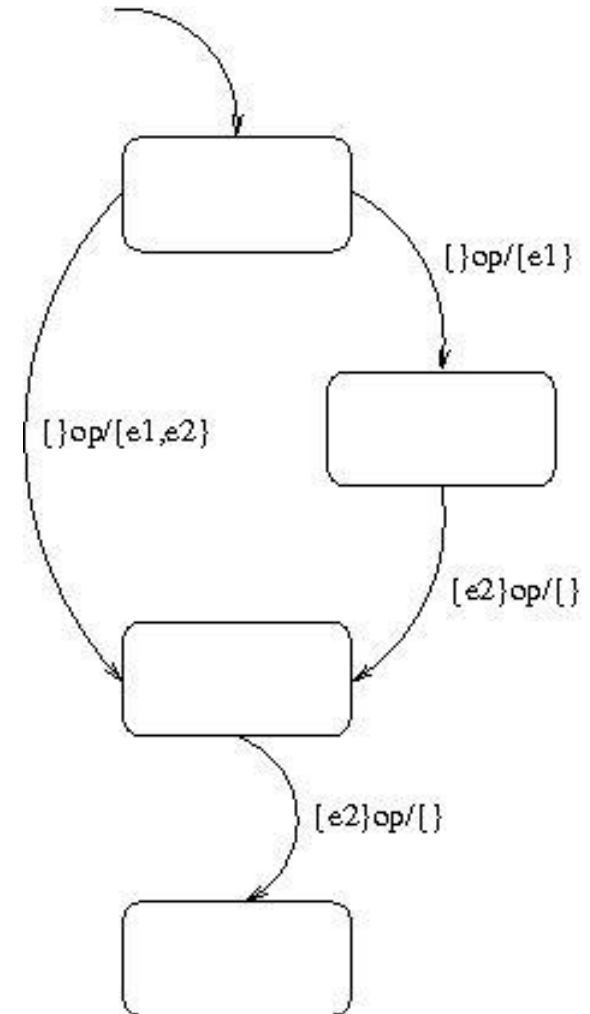
Traitement des wait et notify  
Traitement des branchements  
Exemple de contrat généré par l'outil développé

Implémentation

Conclusion

```
void module1::T1() {  
    int a=0;  
    e1.notify();  
    if(a==0){  
        wait(e2);  
    }  
    else{  
        e2.notify();  
    }  
    wait(e2);  
    a++;  
}
```

Branchement non conditionnel



Evénements attendus	Evénements notifiés



# Exemple de contrat généré par l'outil développé

```
void module1::T1() {
    int a=0;
    if(a<4) {
        e1.notify();
        while(a<8) {
            wait(e2);
            a++;
        }
        wait(e1);
        a++;
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE contract PUBLIC « module1-CT" "contract.dtd">
<contract interface="module1-ITF" name="module1-CT">
  <initials>
    <state name='f0' />
  </initials>
  <transitions>
    <transition lab='{ }op/{toto_0xbfbfd5a8;} src='f0' sink='f1' />
    <transition lab='{ }op/{ } src='f0' sink='f3' />
    <transition lab='{toto_0xbfbfd5e8_0}op/{ } src='f1' sink='f1' />
    <transition lab='{ }op/{ } src='f1' sink='f2' />
    <transition lab='{toto_0xbfbfd5a8_0;}op/{ } src='f2' sink='f3' />
  </transitions>
</contract>
```

Introduction

Algorithmes  
d'extraction  
de contrats 42

Traitement des  
wait et notify

Traitement des  
branchements

Exemple de  
contrat généré  
par l'outil  
développé

Implémentation

Conclusion



Introduction

Algorithmes  
d'extraction  
de contrats 42

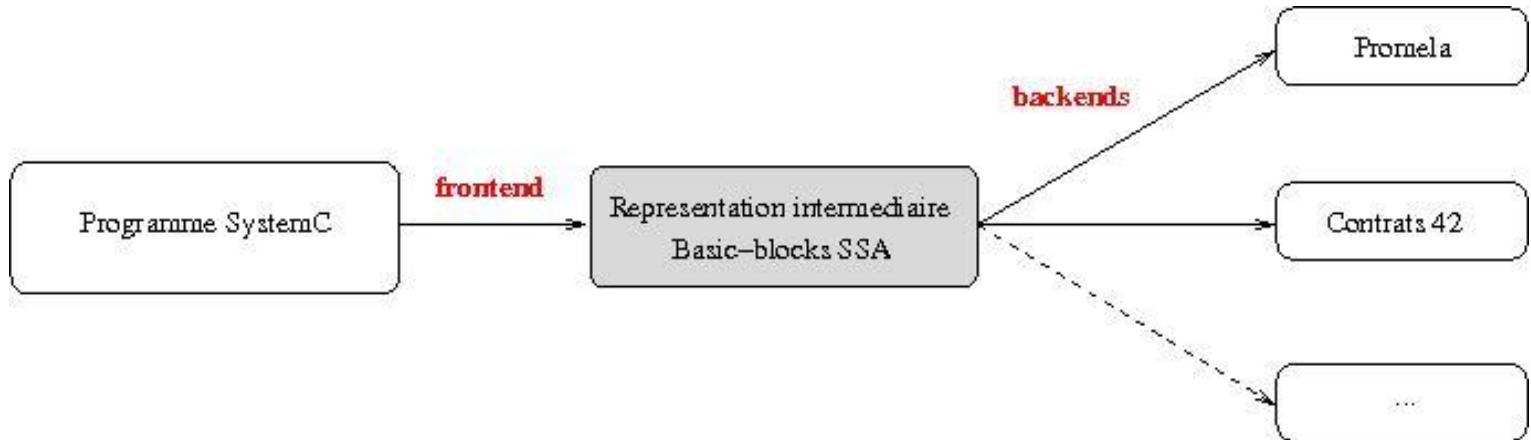
**Implémentation**

Conclusion

# IMPLEMENTATION

# La chaîne d'extraction

Le langage C++ est trop complexe pour qu'on puisse extraire des contrats 42 directement, sans passer par une forme intermédiaire



Introduction

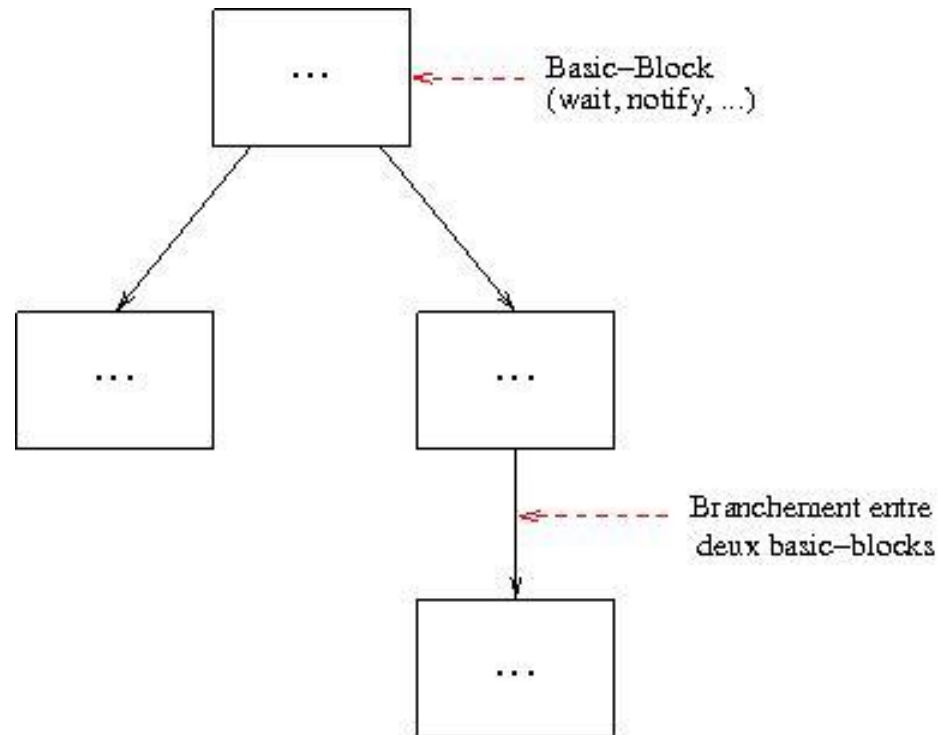
Algorithmes  
d'extraction  
de contrats 42

Implémentation

La chaîne  
d'extraction  
La forme  
intermédiaire  
Comment  
fonctionne  
réellement le  
backend ?

Conclusion

# La forme intermédiaire



**Le frontend : PinaVM** (K. Marquet, M. Moy)

Introduction

Algorithmes  
d'extraction  
de contrats 42

Implémentation

La chaîne  
d'extraction  
La forme  
intermédiaire  
Comment  
fonctionne  
réellement le  
backend ?

Conclusion

# Comment fonctionne réellement le backend ?

- L'algorithme de traitement des branchements est bien celui qui limite le nombre d'états ...  
... mais il distingue la nature des branchements (`if`, `else`, `while`, ...) : inutile !
- Les contrats générés ne sont pas sous forme de fichiers `xml`.


Introduction

Algorithmes  
d'extraction  
de contrats 42

Implémentation

La chaîne  
d'extraction  
La forme  
intermédiaire  
Comment  
fonctionne  
réellement le  
backend ?

Conclusion



Introduction

Algorithmes  
d'extraction  
de contrats 42

Implémentation

**Conclusion**

# CONCLUSION

# Ce qui a été fait :

Génération de contrats 42 :

`wait, notify, if, while, for`

# Les perspectives :

- prendre en compte les appels de fonctions : éléments importants de communication
- vérifier la validité du choix de stratégie de traitement des branchements
- développer un outil pour voir si les contrats de différents composants sont compatibles



# Extraction de contrats 42 à partir de programmes SystemC