Vérification de modèles spécifiés avec CTL

Cécile Braunstein

LIP6 - ASIM

Plan du cours

- Le problème de l'explosion combinatoire
- La composition : cas des "features intégrations"
- Simulation et bi-simulation
- Une autre composition : la conception incrémentale

La vérification formelle

Étant donnée un système et une spécification, **Est-ce que ce système satisfait la spécification ?**

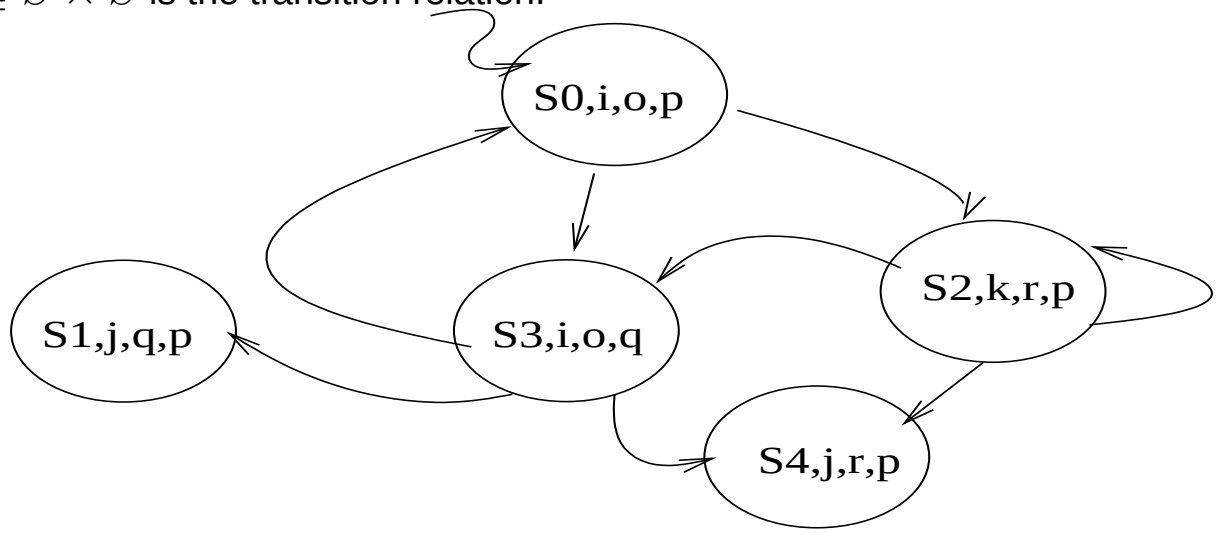
Restriction du problème :

- Le système : Une machine d'états finis M = $\langle S, I, AP, \mathcal{L}, R \rangle$
- Le spécification : Un formule CTL φ.
- La question : M $\models \varphi$?

Machine d'états finis / Kripke structure

La machine à état est étiqueté sur état uniquement, M = $\langle S, I, AP, \mathcal{L}, R \rangle$ avec :

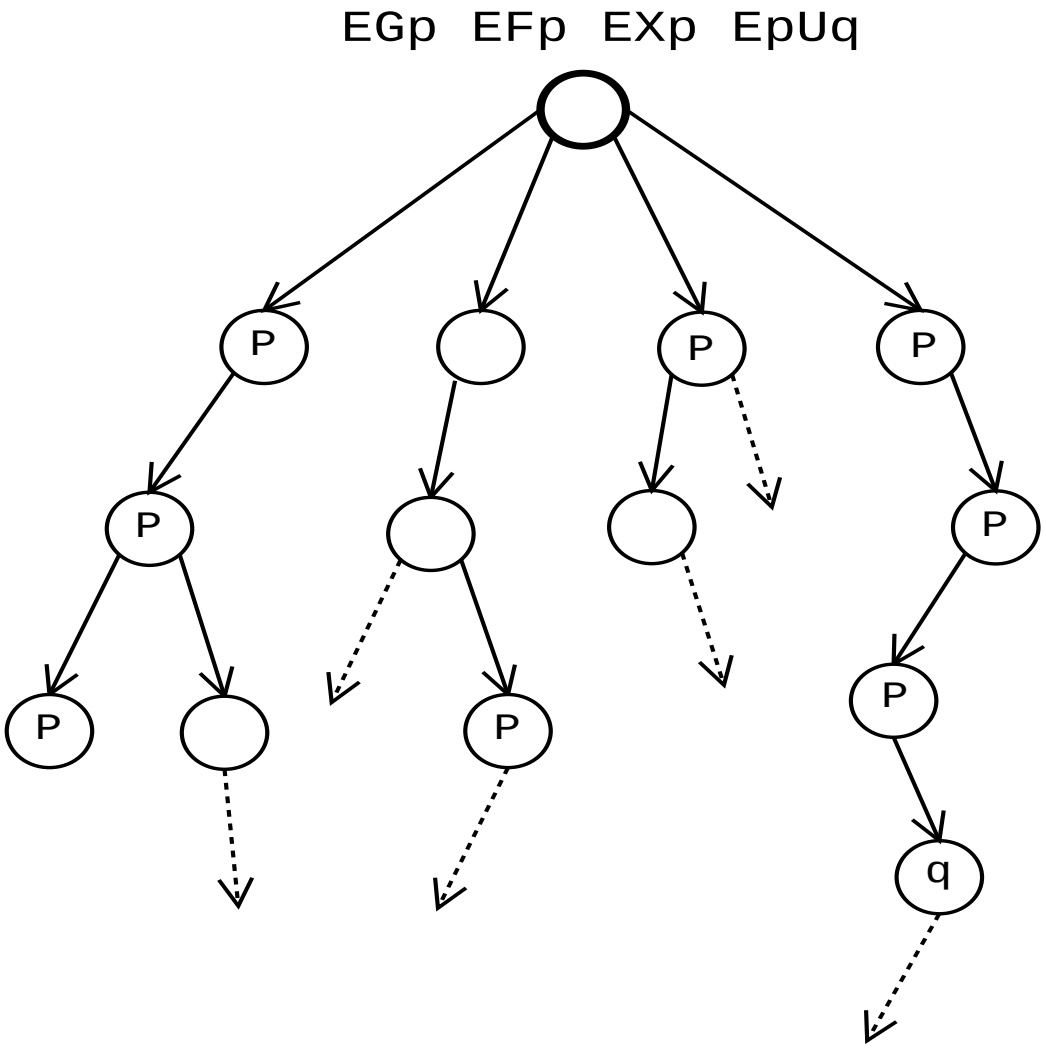
- S is a finite set of states.
- $I \subseteq S$ is the set of initial states.
- AP is a finite set of atomic propositions.
- $\mathcal{L} = \{l_0, \ldots, l_{|AP|-1}\}$ is a vector of $|\text{AP}|$ functions.
- $R \subseteq S \times S$ is the transition relation.



CTL

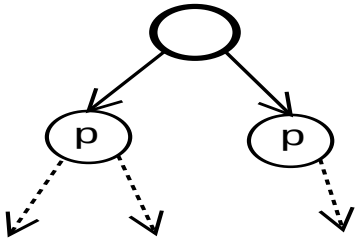
- Logique temporelle arborescente
  - Notion d'ordre d'évênements dans le temps
  - Notion de possible et d'inévitable
- Définie sur des structures de Kripke
- Opérateurs
  - logique propositionnelle : $\neg, \vee, \wedge, \Rightarrow$...
  - opérateurs temporels : X, F, U, G
  - quantificateurs de chemin : E, A
- Tout opérateur temporel est quantifié

ECTL

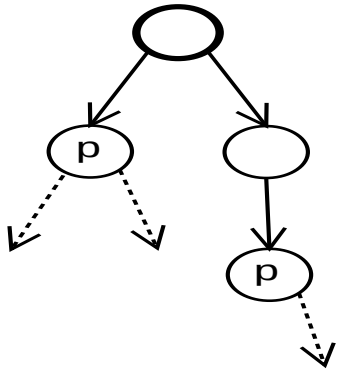


ACTL

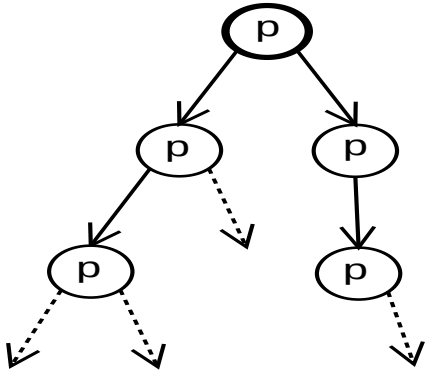
AXp



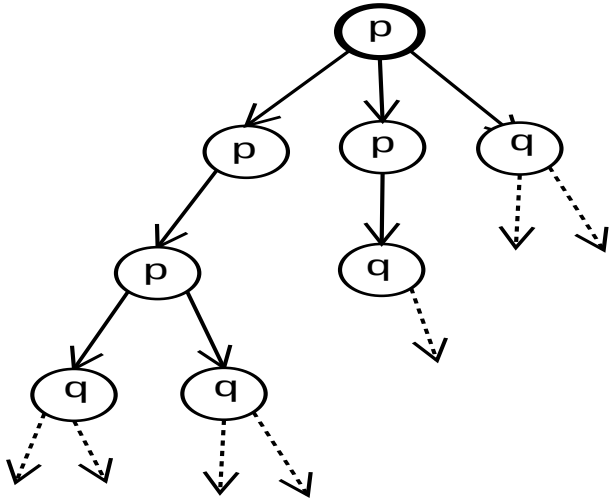
AFp



AGp



ApUq



Le problème de l'explosion combinatoire

model checking : 10^6 états

symbolic model checking (BDD) : 10^{20} états

Une architecture : > 10^6 registres

- Explosion combinatoire : Systèmes trop grands et trop complexes
- Temps de calcul : compléxité linéaire sur la taille du modèle et de la formule

Des solutions :

- Sur les algorithmes et les srtuctures de données : symétries, DDD ...
- Sur les modèles : Abstraction , Raisonnement compositionel (Grumberg et Long 91/94)

L'approche compositionelle est plus intuitive pour le concepteur.

La composition : cas des "Features intégrations" - Présentation générale

Le système est décomposé en un système de base et un ensemble de services (features).

Les services sont des modifications ou des extensions du système de base.

Feature integration est le processus qui intègre les nouveaux services.

La question ici est :

Est-ce que le système reste cohèrent lorsque l'on intègre de nouveaux services ?

Exemple de l'intégration de services pour les téléphones le projet **FireWorks**

- Univeristaires Européens (Universités Birmingham, Namur, Brest, Lisbone ...)
- Industriels (British telecom, France telecom, Belgacom, HP ...)
- But : Création d'une plateforme permettant l'intégration de nouveaux services et la detection des intéractions entre les services.
- Les avantages :
  - Réutilisation de systèmes existant
  - Adaptation et integration des nouveaux services rapides
  - Combinaison variés des services

Interaction

Le problème est l'interaction des services ajoutés entre eux et avec le système de base

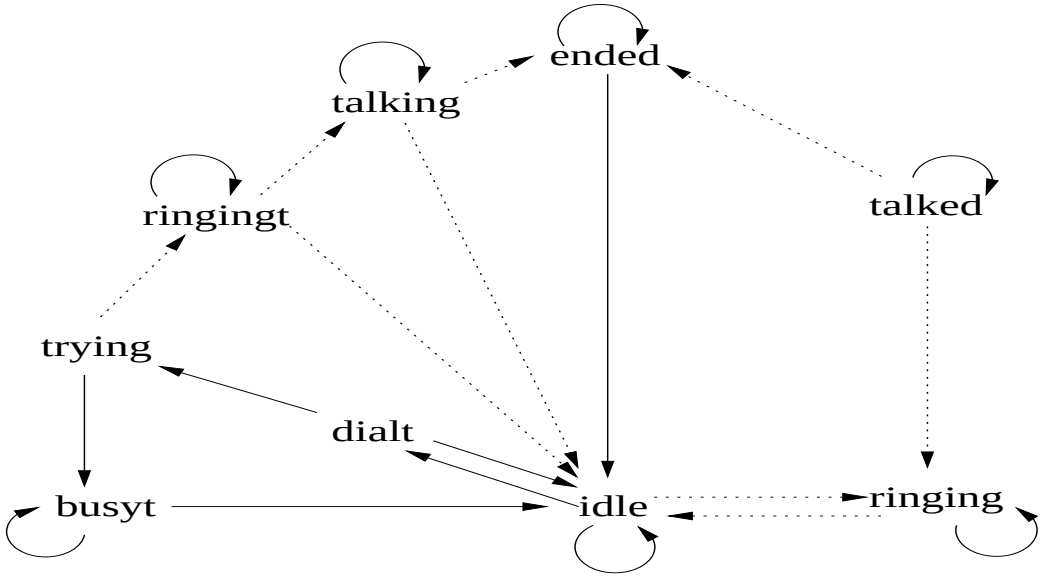
Exemple 1 :	Exemple 2 :
Transfert d'appel si occupé	Rappel automatique
+ Boîte vocale si occupé	+ Transfert d'appel
Mauvaise intéraction	Bonne intéraction

Les intéractions ne sont pas toujours mauvaises mais il faut à chaque fois vérifier que toutes les propriétés du systèmes étendus sont conformes aux propriétés des sevices et du systèmes de bases.

Exemple : Le système de base

Le système de base POTS (Plain old telephone system) : 4 téléphones

Un télephone est caractérisé par son état (state) et le numéro du télephone avec lequel il comunique (dialled).



Propriétés :

1. Tous les télephones peuvent appeller n'importe quel autre téléphone
2. Si un télephone essaye d'appeller alors, au coup d'après il y aura toujours le signal sonnerie ou le signal occupé
3. On ne peut jamais avoir deux télephones communicant avec un même troisième télephone.

Exemple : Les Features

- Call Waiting (CW) : Double appel. Ajout du numéro en attente et de si on entend le message d'attente.

    4. Dans le cas d'un double appel, un seul correspondant entend le message d'attente.

- Call Forward on Busy (CFB) : Transfert d'appel si occupé. Ajout du numéro du transfert.

    5. Si le télephone est occupé alors les appels entrant sont bien transférés. Ajout du numéro à rappelé, et si l'option a été choisit.

- Ring Back When Free (RBWF) : Rappel automatique. Ajout du numéro à rappelé et si les sservice est actif.

    6. Si le service est activé alors dès que les deux téléphones sont dans l'état **idle** l'appel est effectué.

Résultats de l'intégrations des features

Features	1	2	3	4	5	6
POTS	✓	✓	✓	–	–	–
CW	✓	×	×	✓	–	–
CFB	✓	×	✓	–	✓	–
RBWF	✓	✓	✓	–	–	✓
CW + CFB	✓	×	×	✓	✓	–
CFB + CW	✓	×	×	×	✓	–
CW + RBWF	✓	×	×	×	–	✓
RBWF + CW	✓	×	×	✓	–	×
CFB * RBWF	✓	×	✓	–	✓	✓

Inconvénients

- Verification des propriétés à chaque étapes : les anciennes + les nouvelles
- Pas de rapport entre les différents systèmes construits
- Les nouveaux comportements peuvent altérés le comportement de base
- Les incréments sont trop généraux

Présevation de CTL

On voudrait pouvoir écrire une relation entre des modèles.

Plus particulièrement entre un modèle et une composition.

$\longrightarrow$ Les relations de simulation et de bisimulation préserve les propriétés CTL

La simulation (Milner)

Soit deux modèles $M_1 = \langle S_1, I_1, AP_1, \mathcal{L}_1, R_1 \rangle$ et $M_2 = \langle S_2, I_2, AP_2, \mathcal{L}_2, R_2 \rangle$.

H$\subseteq S_1 \times S_2$ est une relation de simulation ssi

Pour tout $(s_1, s_2) \in$ **H** :

- $\mathcal{L}_1(s_1) \cap AP_2 = \mathcal{L}_2(s_2)$
- Pour tout successeur t_1 de s_1 il existe un successeur t_2 de s_2 tel que $(t_1,t_2) \in$**H**

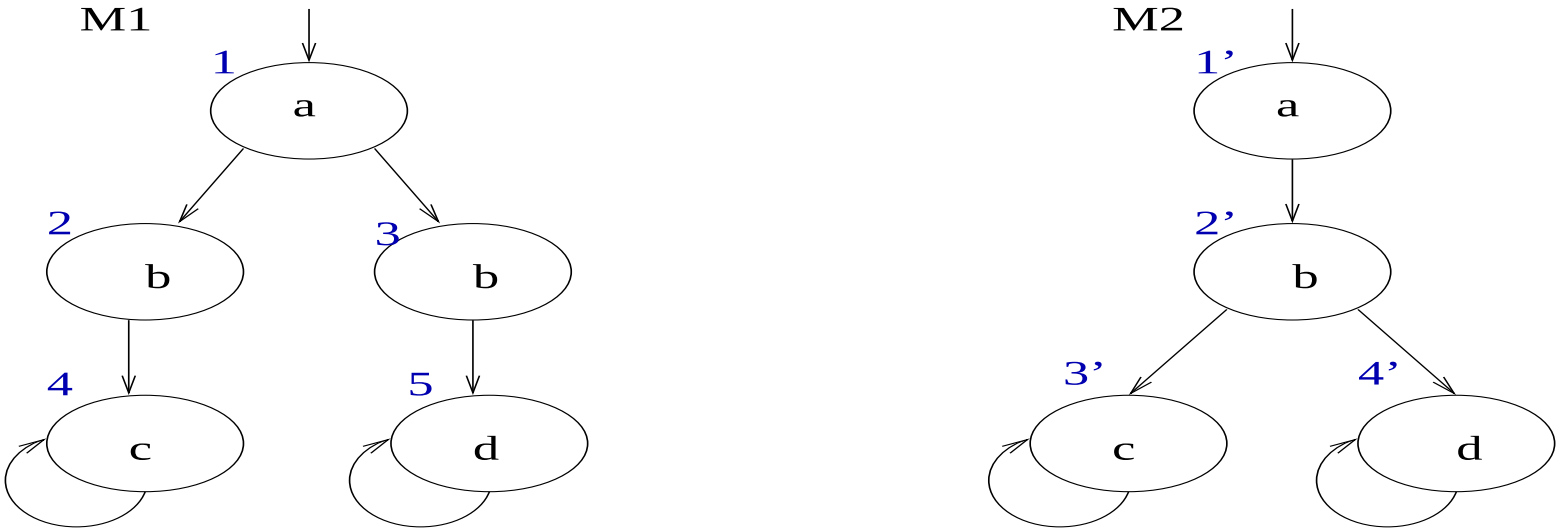
  Notation : $s_1 \preceq s_2$

  On dit que **H** est simulation de M_1 dans $M2$ ssi :

- **H** est une relation de simulation
- Pour tout état $s_1 \in I_1$ il existe un état $s_2 \in I_2$ tel que $(s_1,s_2) \in$**H**

  Notation : $M_1 \preceq M_2$

Exemple de simulation



$$M_1 \preceq M_2$$

$$M_1 \not\preceq M_2$$

H = {(1,1’),(2,2’),(3,2’),(4,3’),(5,4’)}

La Bisimulation (Park)

Soit deux modèles $M_1 = \langle S_1, I_1, AP_1, \mathcal{L}_1, R_1 \rangle$ et $M_2 = \langle S_2, I_2, AP_2, \mathcal{L}_2, R_2 \rangle$.

H$\subseteq S_1 \times S_2$ est une relation de bisimulation ssi

Pour tout $(s_1, s_2) \in$ **H** :

- $\mathcal{L}_1(s_1) = \mathcal{L}_2(s_2)$ tés
- Pour tout successeur t_1 de s_1 il existe un successeur t_2 de s_2 tel que $(t_1,t_2) \in$**H**
- Pour tout successeur t_2 de s_2 il existe un successeur t_1 de s_1 tel que $(t_1,t_2) \in$**H**

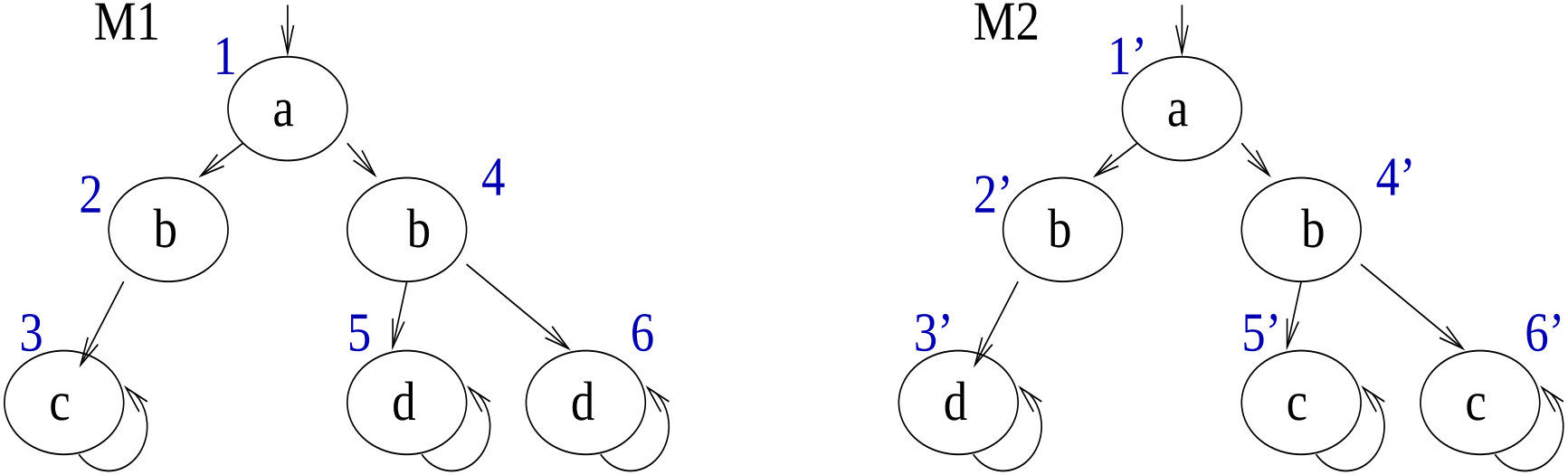
Notation : $s_1 \equiv s_2$

On dit que **H** est bisimulation entr M_1 et $M2$ ssi :

- **H** est une relation de bisimulation
- Pour tout état $s_1 \in I_1$ il existe un état $s_2 \in I_2$ tel que $(s_1,s_2) \in$**H**
- Pour tout état $s_2 \in I_2$ il existe un état $s_1 \in I_1$ tel que $(s_1,s_2) \in$**H**

Notation : $M_1 \equiv M_2$

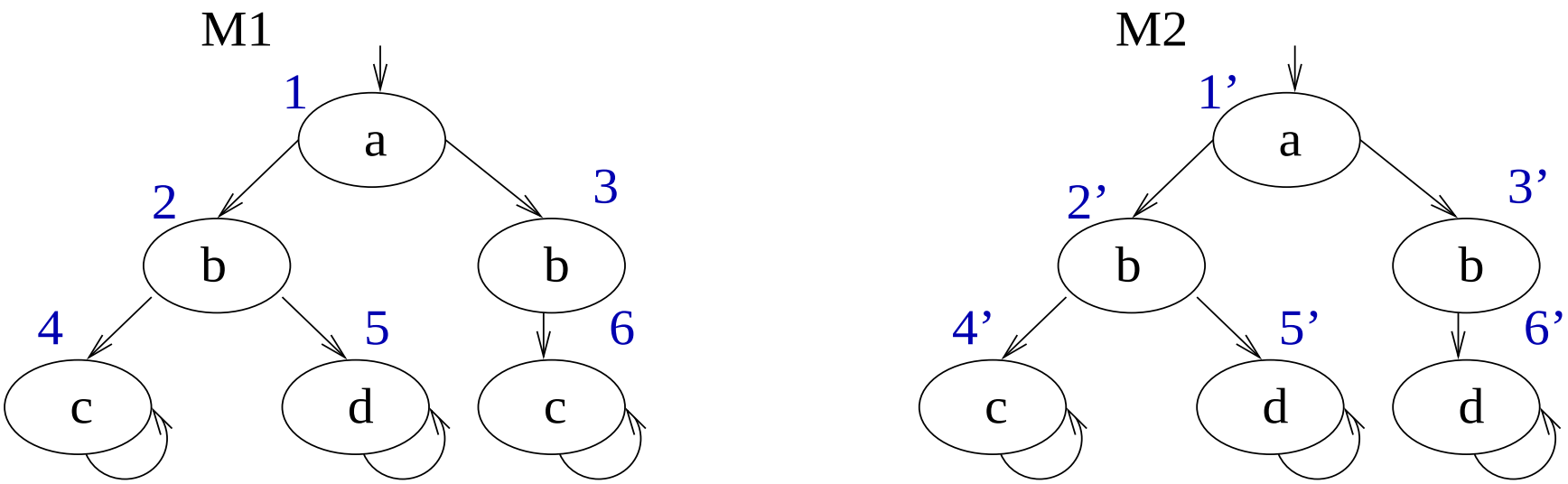
Exemple de bisimulation



$$M_1 \equiv M_2$$

H = {(1,1’),(2,4’),(4,2’),(3,5’),(3,6’),(5,3’)(5,6’)}

Exercice



$M_1 \preceq M_2$, $M_2 \preceq M_1$, $M_1 \equiv M_2$?

Préservation

Théorèmes :

- (Grumberg) Si $M_1 \equiv M_2$ alors pour toute formule CTL φ :

$M_1 \models \varphi \Leftrightarrow M_2 \models \varphi$

- (Grumberg) Si $M_1 \preceq M_2$ alors pour toute formule ACTL φ :

$M_2 \models \varphi \Rightarrow M_1 \models \varphi$

- (Loiseau) Si $M_1 \preceq M_2$ alors pour toute formule ECTL φ :

$M_1 \models \varphi \Rightarrow M_2 \models \varphi$

La conception incrémentale

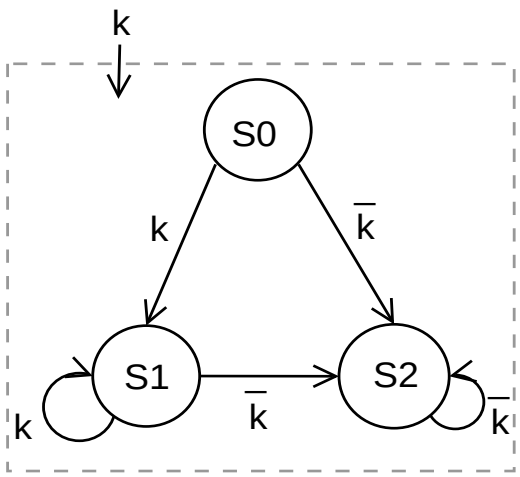
- Stratégie incremental : **additions successives de nouveaux comportement**
  - ➢ Calqué sur la démarche de conception des designers (pipeline flow)

Principes :

- $M_i \longrightarrow M_{i+1}$, au moins 1 nouvel événement en entrée
- Une valeur muette est attribuée à chaque nouveau signal
- M_{i+1} englobe le comportement de M_i
- Conservation des états initiaux de M_i dans M_{i+1}
  (pas de nouveaux états initiaux)

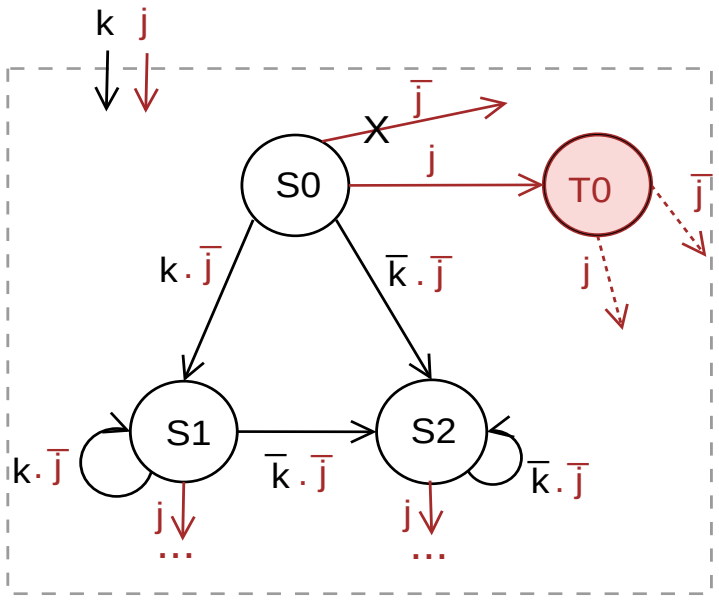
Increment Definition

- Increment INC is a set of new events



INC

New signal j added
quiet value : {0}
active value : {1}



Wi: complete deterministic Moore Machine

Wi+1: complete deterministic Moore Machine

- M_{i+1} simulates M_i
- $K(M_{i+1})$ simulates $K(M_i)$ (Kripke structure)
- $K(M_{i+1})$ includes $K(M_i)$ with the state that were in $K(M_i)$ tagged with the quiet value

CTL-property transformations

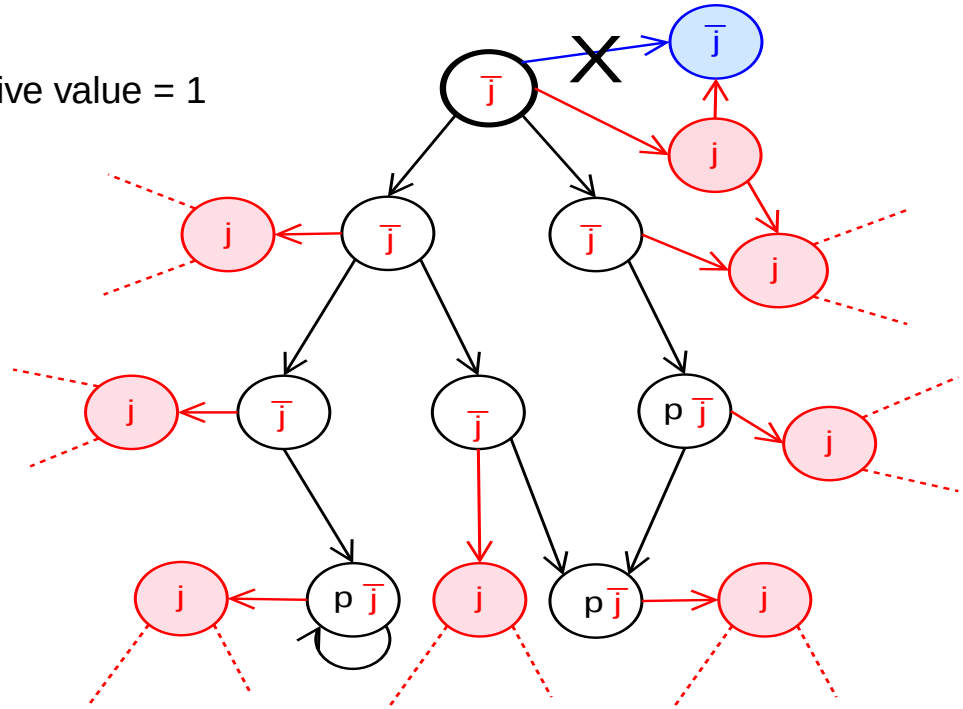
- Goal : Let be φ such that W_i,$s_0 \models \varphi$, what is φ' such that :
  K(W_i),$s_0 \models \varphi \Leftrightarrow K(W_{i+1})$,$s_0$' $\models \varphi$' with $K(W_{i+1})$ obtained by increment from $K(W_i)$
- Principle : Reduction of the computational tree explored

Example : transformation of AFp

$K(W_i) \models$ AFp

INC : j,quiet value = 0,active value = 1

$K(W_{i+1}) \models$ AF(p or j)



Transformations rules

Φ = p $\Leftrightarrow \Phi'$ = p

Φ = not f $\Leftrightarrow \Phi'$ = not f'

Φ = EXf $\Leftrightarrow \Phi'$ = (e = val_qt) $\Rightarrow$ EXf'

Φ = EFf $\Leftrightarrow \Phi'$ = E((e = val_qt) U f')

Φ = EGf $\Leftrightarrow \Phi'$ = EG((e = val_qt) and f')

Φ = Ef U g $\Leftrightarrow \Phi'$ = E(((e = val_qt) and f') U g')

Φ = AXf $\Leftrightarrow \Phi'$ = (e = val_qt) $\Rightarrow$ AXf'

Φ = AFf $\Leftrightarrow \Phi'$ = AF((e $\neq$ val_qt) or f')

Φ = Af U g $\Leftrightarrow \Phi'$ = A(((e = val_qt) and f') U ((e $\neq$val_qt) or g'))

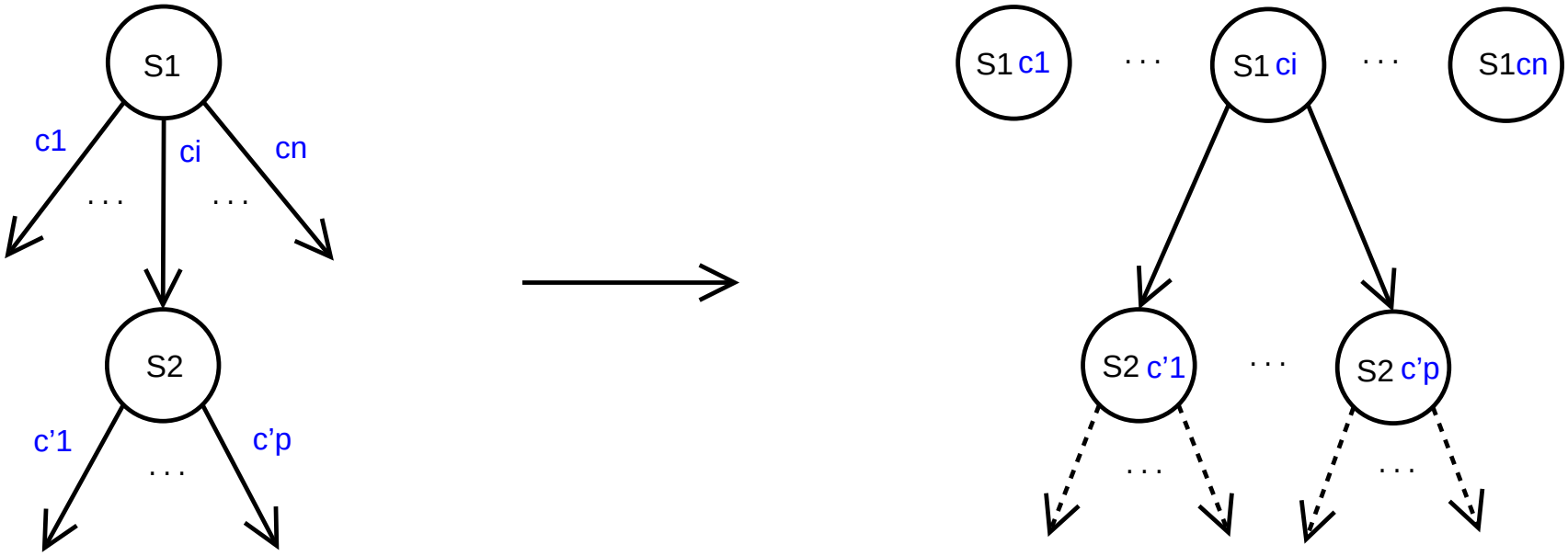
Φ = AGf $\Leftrightarrow \Phi'$ = A(((e = val_qt) and f') W (e $\neq$ val_qt))

Φ = Af W g $\Leftrightarrow \Phi'$ = A(f' W (g' or (e $\neq$ val_qt)))

Conclusion

- Transformation rules
  - All CTL operators are transformable (bi-implication)
  - All CTL formulae are transformable by recursively applying the transformation
  - The transformed CTL formulae have the same complexity as the initial ones
- Application to a concrete component design (VCI-PI protocol converter)
  - System with 330-450 boolean variables
  - Transformations applying on 80 properties automatically
  - The transformed properties do not increase significally the time of verification
- Further studies :
  - Taking advantage of the increment graph structure
  - Others simulation relation (stuttering equivalence)

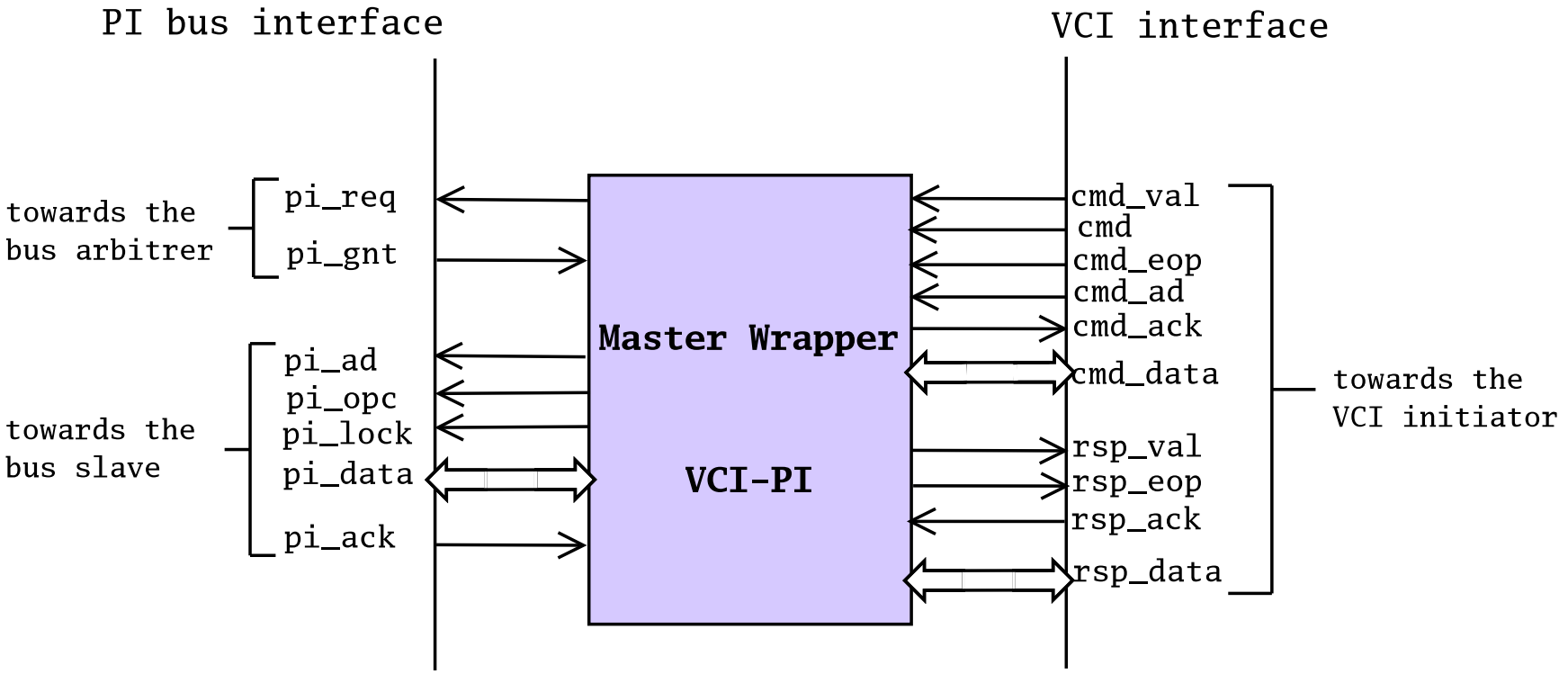
Transformation of a Moore machine into a Kripke structure



Moore Machine

Kripke Structure

VCI-PI Wrapper interface

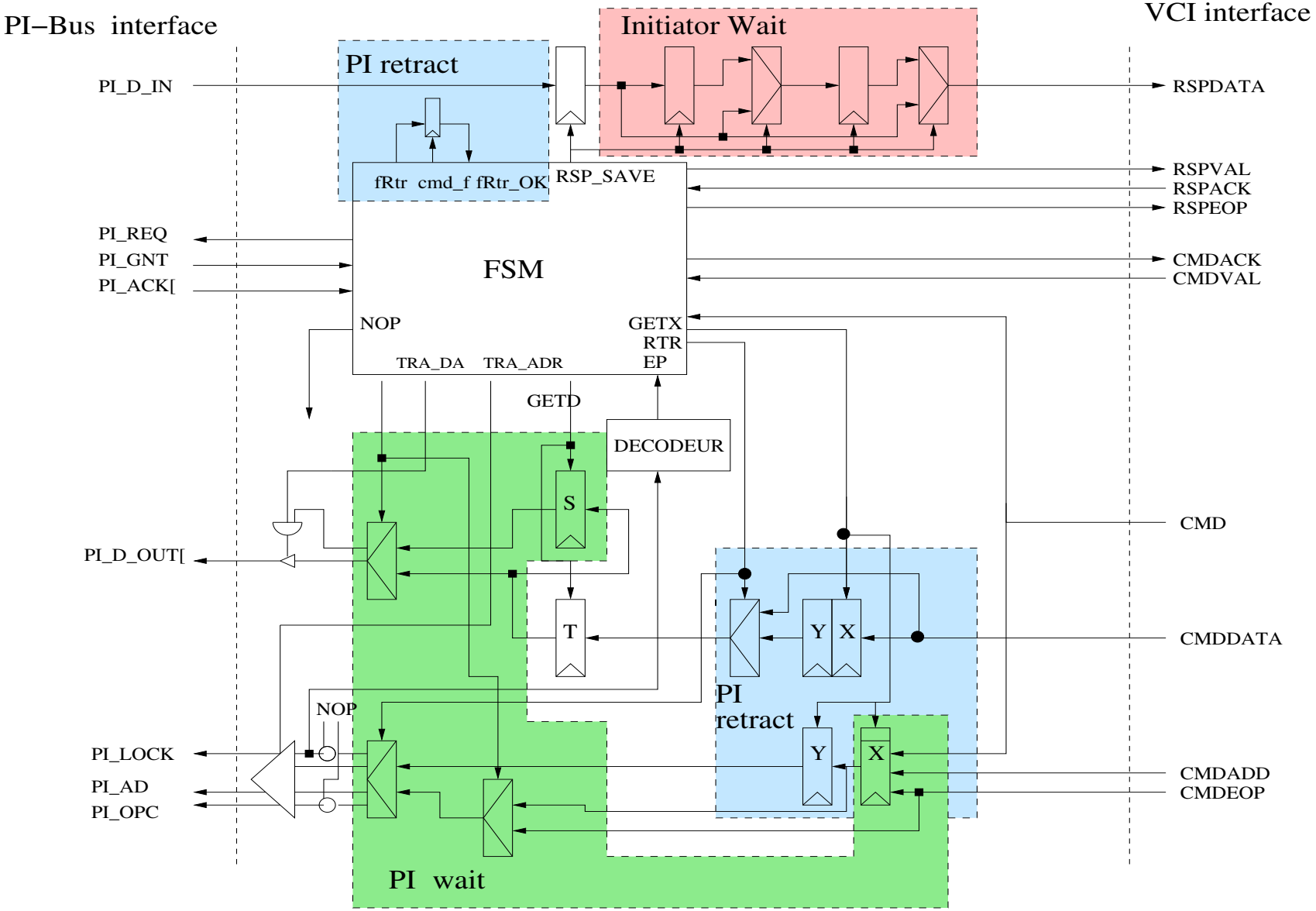


Wrappers Hierarchy

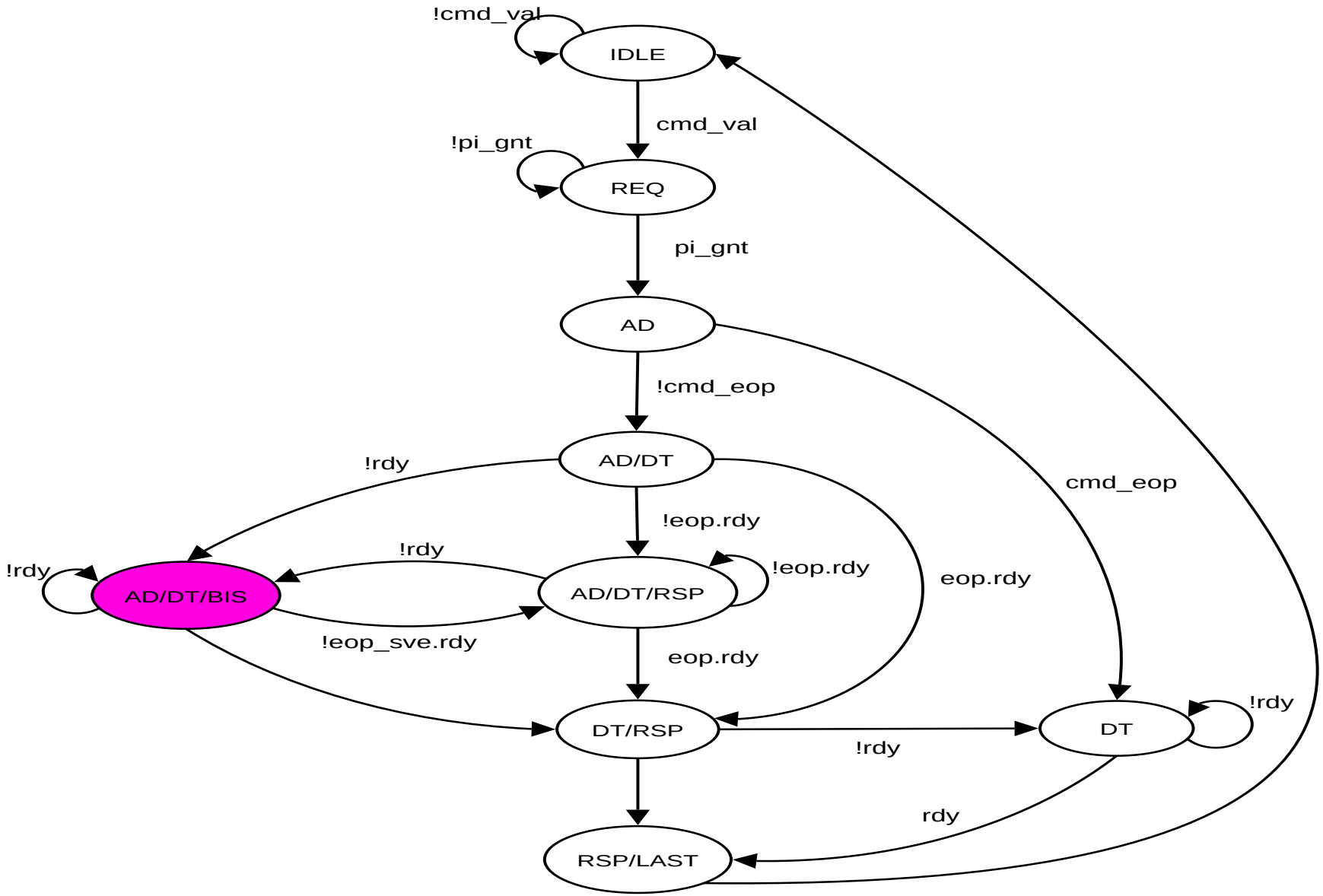
Type of event considered	Initiator is always ready	Initiator may impose wait states
Target is always ready pi_rsp = RDY	A cmd_ack = 1 ; cmd_val = 1 rsp_val = 1 ; rsp_ack = 1	A′ cmd_ack = 1 ; cmd_val = {0,1} rsp_val = 1 ; rsp_ack = {0,1}
Target may impose wait states pi_rsp = {RDY,WAIT}	B cmd_ack = {0,1} ; cmd_val = 1 rsp_val = {0,1} ; rsp_ack = 1	B′ cmd_ack = {0,1} ; cmd_val = {0,1} rsp_val = {0,1} ; rsp_ack = {0,1}
Target may impose retract pi_rsp = {RDY,WAIT,RTR}	C cmd_ack = {0,1} ; cmd_val = 1 rsp_val = {0,1} ; rsp_ack = 1	C′ cmd_ack = {0,1} ; cmd_val = {0,1} rsp_val = {0,1} ; rsp_ack = {0,1}

→ : Increment

VCI-PI Wrappers Datapath (C)



VCI-PI Wrappers FSM (B')



VCI-PI Wrappers FSM(A and B)