Application de la logique de Hoare aux réseaux de régulation génétique avec multiplexes

Maxime FOLSCHETTE

École Centrale de Nantes
Cursus ingénieur : Ei3 Informatique
Cursus Master Recherche : M2 ASP-SPIE

Encadrants :

Olivier ROUX
Morgan MAGNIN

Enjeux et problématique

- Gène = séquence de l'ADN codant
  la production d'une protéine
- Les gènes ne s'expriment pas tous
  à tout moment / dans toutes les cellules
- Étudier certains mécanismes de régulation
- Modéliser les systèmes de gènes

Enjeux et problématique

- Étude des interactions entre gènes

  Modéliser les mécanismes de régulation
  Étude du comportement et de la dynamique

- Utilisation du modèle

  Simplification cohérente (discrétisation)
  Mais explosion combinatoire
  → Difficultés d'analyse

- Nouvel outil

  Logique de Hoare
  → **Implémentations**

Plan de la présentation

- **Modèle de Thomas et logique de Hoare**

  Réseaux de régulation avec multiplexes
  Triplets de Hoare et plus faible pré-condition
  Inférence de paramètres

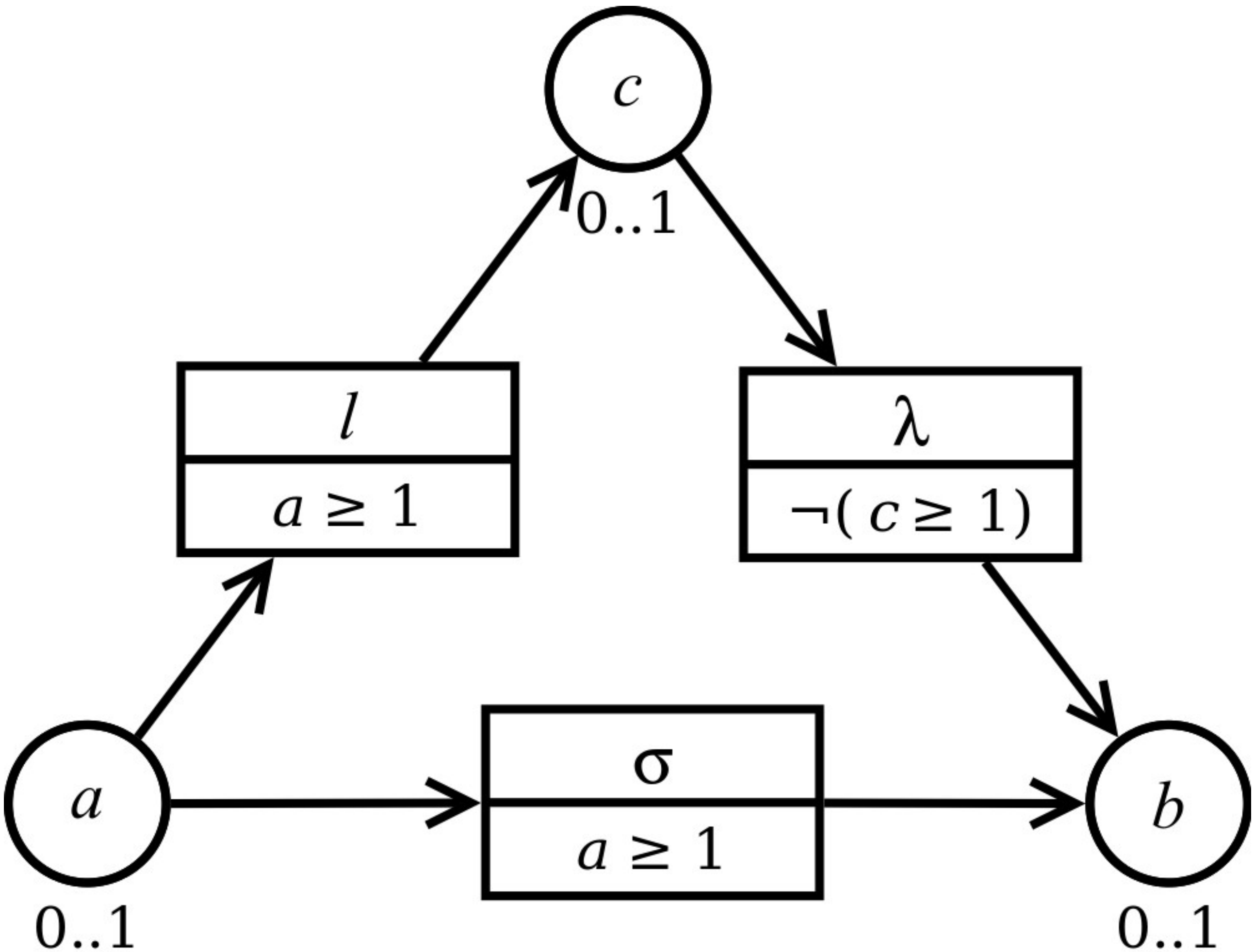
- **Implémentation avec Coq**
- **Implémentation avec OCaml**

  Présentation des langages
  Implémentation des outils et de la logique
  Obtention des résultats / Exemples

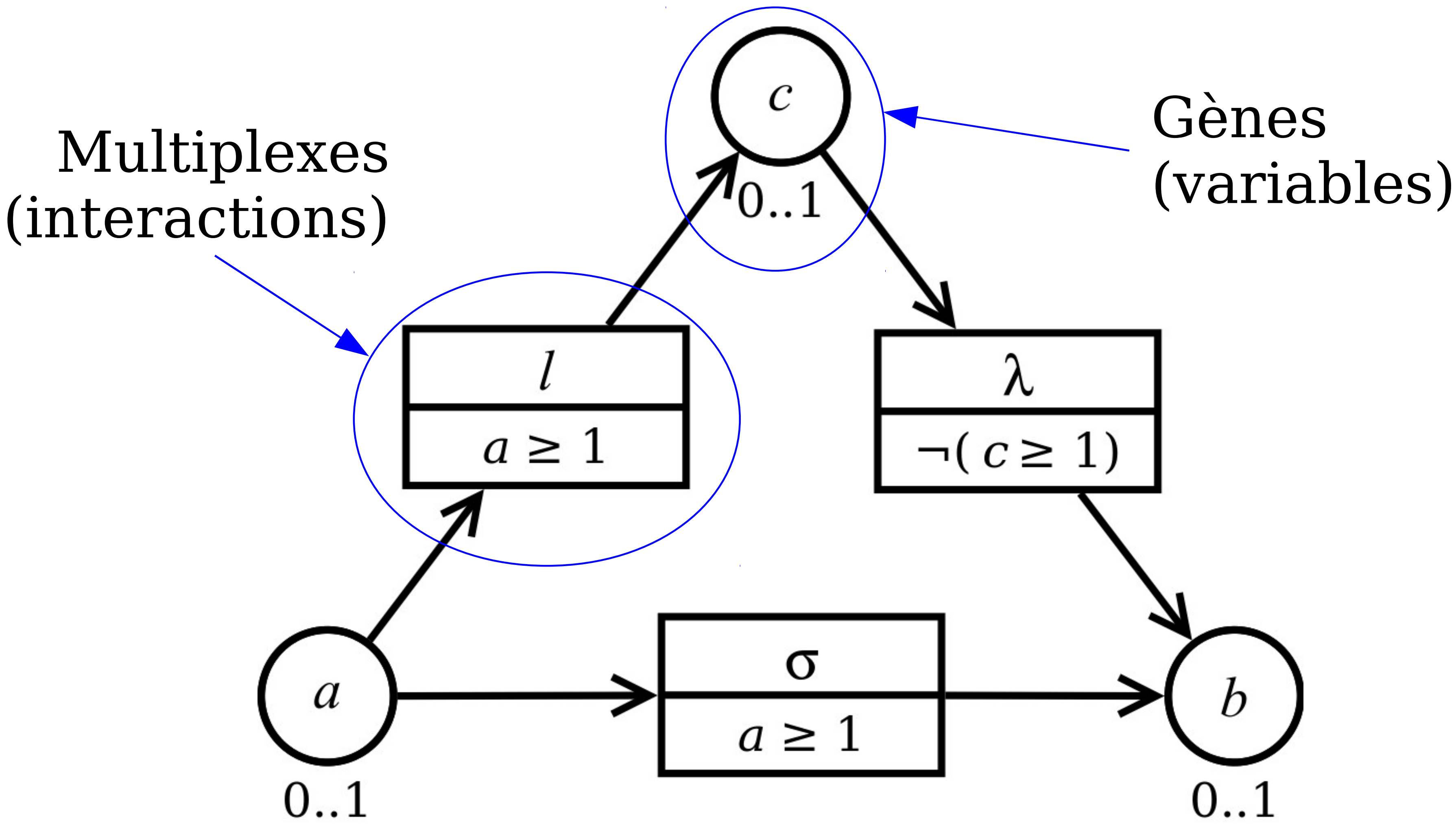
- **Discussion**

  Problèmes rencontrés
  Pistes de développement

Graphe d'interaction [1, 2, 6]



Graphe d'interaction [1, 2, 6]



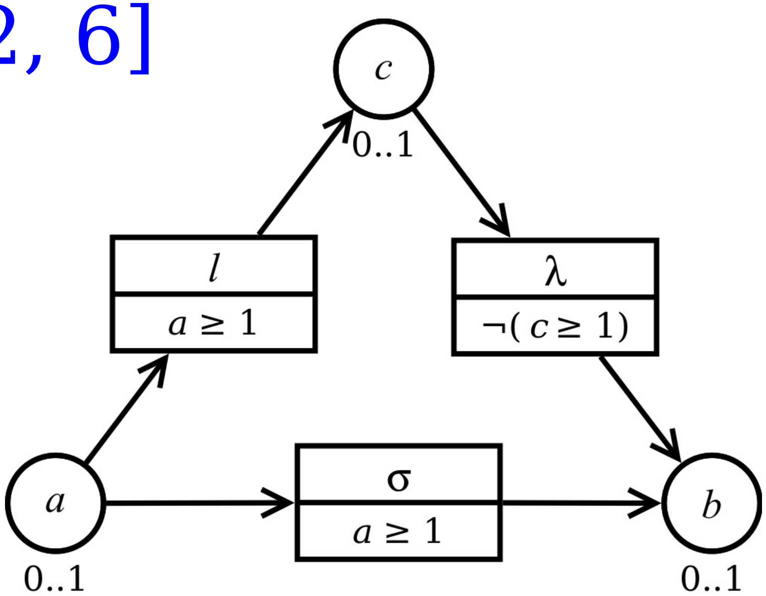
Paramétrisation [1, 2, 6]

Carte des tendances :

$$\begin{aligned} K : V \times \mathcal{P}(M) &\to \mathbb{N} \\ (\ v\ ;\ \omega\ ) &\mapsto \boldsymbol{k_{v,\omega}} \end{aligned}$$

Où : $k_{v,\omega} \leq b_v$ et $\omega \subset G^{-1}(v)$

Paramétrisation [1, 2, 6]

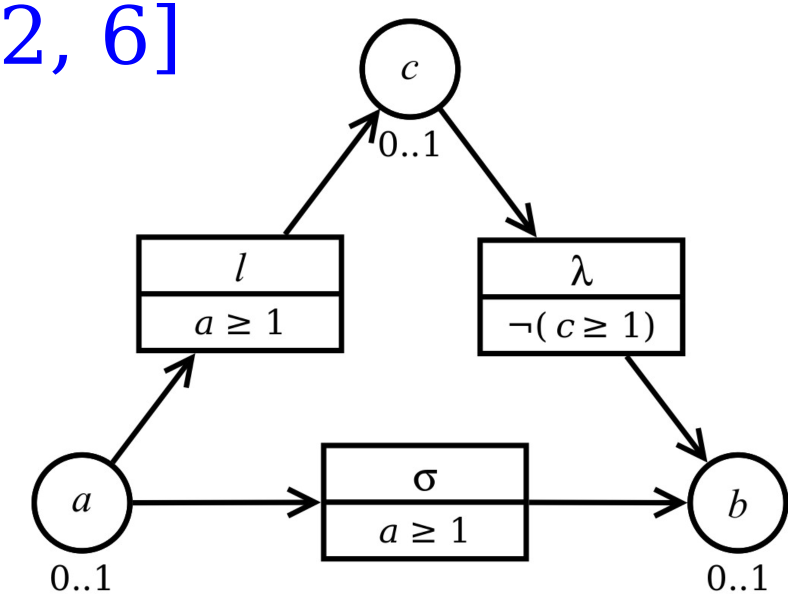


Carte des tendances :

$$\begin{aligned} K : V \times \mathcal{P}(M) &\to \mathbb{N} \\ (\ v\ ;\ \omega\ ) &\mapsto \boldsymbol{k_{v,\omega}} \end{aligned}$$

Où : $k_{v,\omega} \leq b_v$ et $\omega \subset G^{-1}(v)$

Paramétrisation [1, 2, 6]



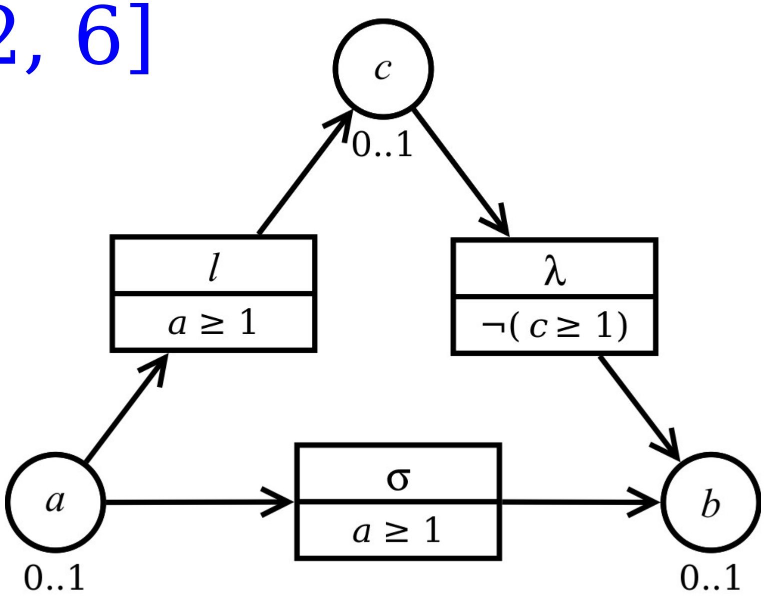
Carte des tendances :

$$\mathrm{K} : \mathrm{V} \times \mathcal{P}(\mathrm{M}) \to \mathbb{N}$$
$$(\ v\ ;\ \ \omega\ \ ) \mapsto \boldsymbol{k_{v,\omega}}$$

Où : $k_{v,\omega} \leq b_v$ et $\omega \subset \mathrm{G}^{-1}(v)$

ω	$k_{a,\omega}$
$\emptyset$	1

Paramétrisation [1, 2, 6]



Carte des tendances :

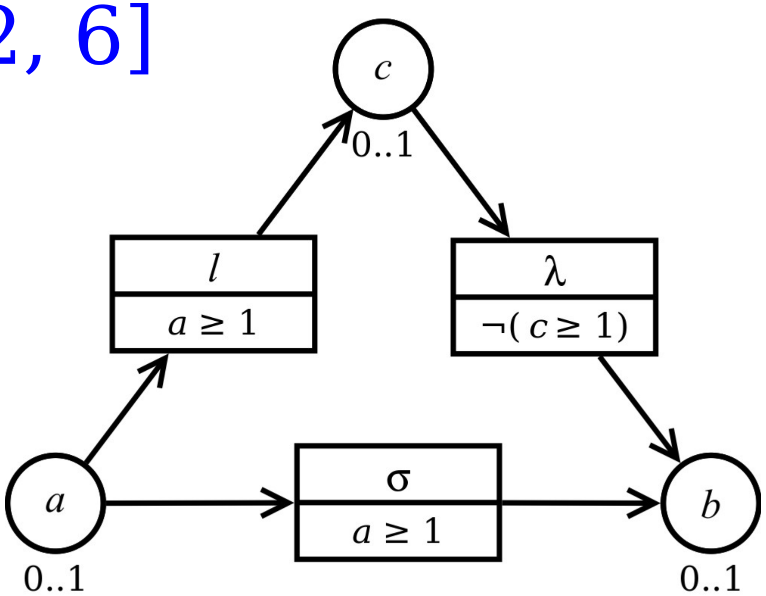
$$\begin{aligned} K : V \times \mathcal{P}(M) &\rightarrow \mathbb{N} \\ (\ v\ ;\ \ \omega\ ) &\mapsto \boldsymbol{k_{v,\omega}} \end{aligned}$$

Où : $k_{v,\omega} \leq b_v$ et $\omega \subset G^{-1}(v)$

ω	$k_{a,\omega}$
$\emptyset$	1

ω	$k_{c,\omega}$
$\emptyset$	0
$\{l\}$	1

Paramétrisation [1, 2, 6]



Carte des tendances :

$$\begin{aligned} K : V \times \mathcal{P}(M) &\to \mathbb{N} \\ (\ v\ ;\ \ \omega\ \ ) &\mapsto \boldsymbol{k_{v,\omega}} \end{aligned}$$

Où : $k_{v,\omega} \leq b_v$ et $\omega \subset G^{-1}(v)$

ω	$k_{a,\omega}$
$\emptyset$	1

ω	$k_{b,\omega}$
$\emptyset$	0
$\{\lambda\}$	1
$\{\sigma\}$	0
$\{\lambda, \sigma\}$	1

ω	$k_{c,\omega}$
$\emptyset$	0
$\{l\}$	1

Réseau de régulation [1, 2, 6]

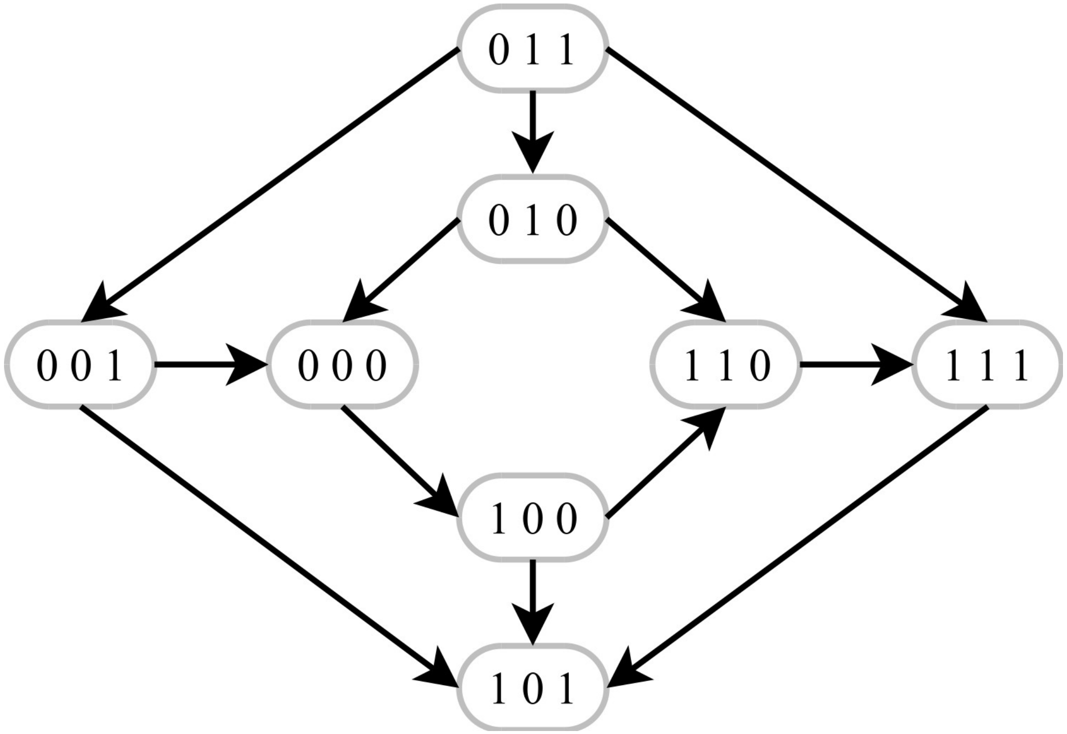
Graphe d'interactions + paramétrisation

→ Décrit entièrement la dynamique du système
→ Permet le calcul du graphe d'états

Réseau de régulation [1, 2, 6]

Graphe d'interactions + paramétrisation

→ Décrit entièrement la dynamique du système
→ Permet le calcul du graphe d'états



Réseau de régulation [1, 2, 6]

Graphe d'interactions + paramétrisation

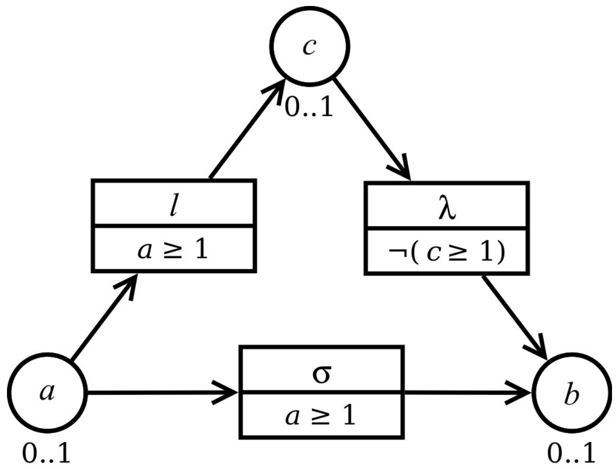
→ Décrit entièrement la dynamique du système
→ Permet le calcul du graphe d'états

Problème : paramétrisation

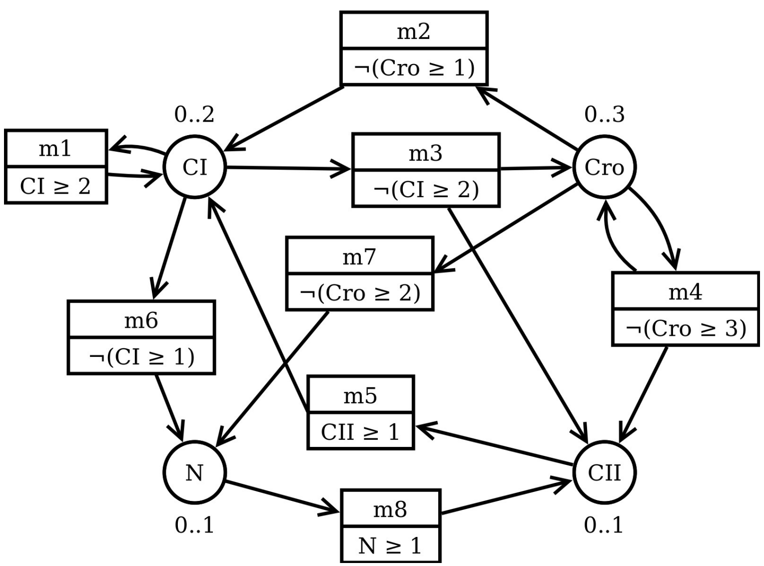
→ Nécessaire pour obtenir le bon comportement
→ Multiples paramétrisations possibles
→ Recherche ⇒ **Explosion combinatoire**

$$N = \prod_{v} (b_v + 1)^{2^{\left|G^{-1}(v)\right|}}$$

Explosion combinatoire



$$N = 128$$



$$N = 6\,879\,707\,136$$

Logique de Hoare [4]

Triplets de Hoare :

$$\{\ \boldsymbol{P}\ \}\ \boldsymbol{Q}\ \{\ \boldsymbol{R}\ \}$$

P : Pré-condition
Q : Programme informatique
R : Post-condition

« Si ***P*** est vraie avant exécution de ***Q***,
alors ***R*** sera vraie après exécution de ***Q***. »

Exemple : { y = 4 } y:=y+1 { y = 5 }

Axiomes et règles [4]

- **Axiomes** pour fonder la logique
  - Programme vide : { P } `Skip` { P }
  - Affectation : { $P[expr/var]$ } $var{:=}expr$ { P }

- **Règles** pour construire la logique
  - Conséquence
  - Composition de deux programmes
  - Structures `Si-Alors-Sinon` et `Tant_que`
  - Quantificateurs existentiel et universel

Plus faible pré-condition [5]

$$\{\ P\ \}\ Q\ \{\ R\ \}$$

Connus

Plus faible pré-condition [5]

$$\{ P \} \; Q \; \{ R \}$$

Opérateur : **WP(*Q*, *R*)**

Connus

Plus faible pré-condition [5]

$\{ P \}\ Q\ \{ R \}$

Opérateur : **WP(*Q*, *R*)** Connus

Nouvelles règles pour cet opérateur :

- Composition
- Affectation

$$\mathbf{WP}(var{:=}expr,\ R) \equiv R[expr/var]$$

- Structures `Si-Alors-Sinon` et `Tant_que`

→ On connaît exactement la pré-condition d'une affectation

Application au modèle de Thomas [6]

$$\{\ P\ \}\ Q\ \{\ R\ \}$$

Application au modèle de Thomas [6]

Assertion sur le niveau d'expression des gènes

ex : $a = 1 \land c = 1$

$$\{\ P\ \}\ Q\ \{\ R\ \}$$

Application au modèle de Thomas [6]

Assertion sur le niveau d'expression des gènes

ex : $a = 1 \wedge c = 1$

$$\{ P \} \; Q \; \{ R \}$$

Application au modèle de Thomas

[6]

Évolution du système : incrémentations ou décrémentations

ex : $c+$

→ **Affectations**

Assertion sur le niveau d'expression des gènes

ex : $a = 1 \land c = 1$

$$\{\ P\ \}\ Q\ \{\ R\ \}$$

Application au modèle de Thomas

[6]

Évolution du système : incrémentations ou décrémentations

ex : $c+$

→ **Affectations**

Assertion sur le niveau d'expression des gènes

ex : $a = 1 \wedge c = 1$

$$\{ P \} \; Q \; \{ R \}$$

Application au modèle de Thomas

[6]

Évolution du système : incrémentations ou décrémentations

ex : $c+$

→ **Affectations**

Assertion sur le niveau d'expression des gènes

ex : $a = 1 \wedge c = 1$

$$\{ P \} \; Q \; \{ R \}$$

Calcul de la plus faible pré-condition

Application au modèle de Thomas

[6]

Évolution du système : incrémentations ou décrémentations

ex : $c+$

→ **Affectations**

Assertion sur le niveau d'expression des gènes

ex : $a = 1 \wedge c = 1$

$$\{ P \} \; Q \; \{ R \}$$

Calcul de la plus faible pré-condition

⇒ Assertion sur :

- le niveau d'expression des gènes ex : $a = 1 \wedge c = 0$
- la paramétrisation ex : $k_{c,\{l\}} = 1$

Application au modèle de Thomas

[6]

Exemple d'application :

$\{ \quad ? \quad \}\ c+\ \{\ a = 1 \wedge c = 1\ \}$

Application au modèle de Thomas [6]

Exemple d'application :

$$\{ \Phi \wedge a = 1 \wedge c = 0 \} \; c+ \; \{ a = 1 \wedge c = 1 \}$$

Application au modèle de Thomas

[6]

Exemple d'application :

$$\{ \Phi \wedge a = 1 \wedge c = 0 \} \; c+ \; \{ a = 1 \wedge c = 1 \}$$

où Φ signifie « On peut incrémenter c », *i.e.* :

- c est en dessous de son plafond
- la paramétrisation de c permet son incrémentation

Application au modèle de Thomas

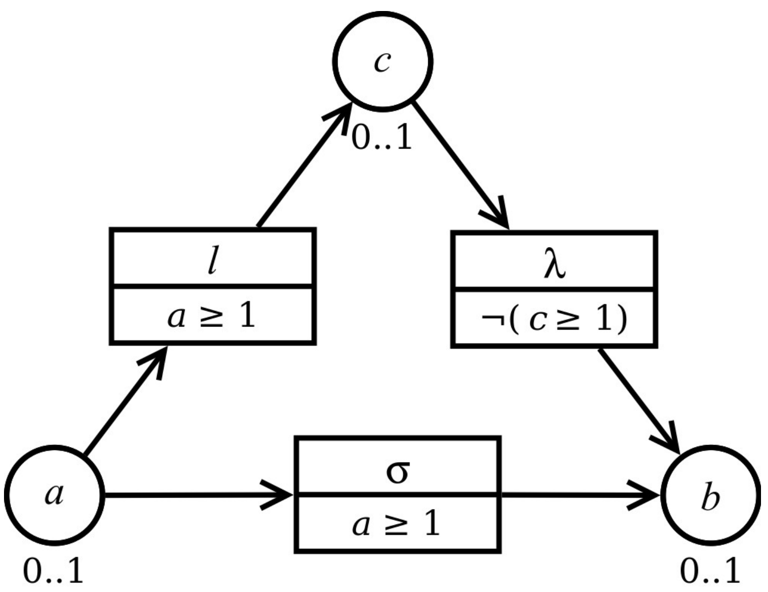
[6]

Exemple d'application :

$\{ \Phi \wedge a = 1 \wedge c = 0 \}\ c+\ \{ a = 1 \wedge c = 1 \}$

$$\Phi \equiv c \geq 0 \wedge c < 1$$
$$\neg\varphi_l \Rightarrow k_{c,\varnothing} > c$$
$$\varphi_l \Rightarrow k_{c,\{l\}} > c$$

où : $\varphi_l \equiv (a \geq 1)$



Application au modèle de Thomas

[6]

Exemple d'application :

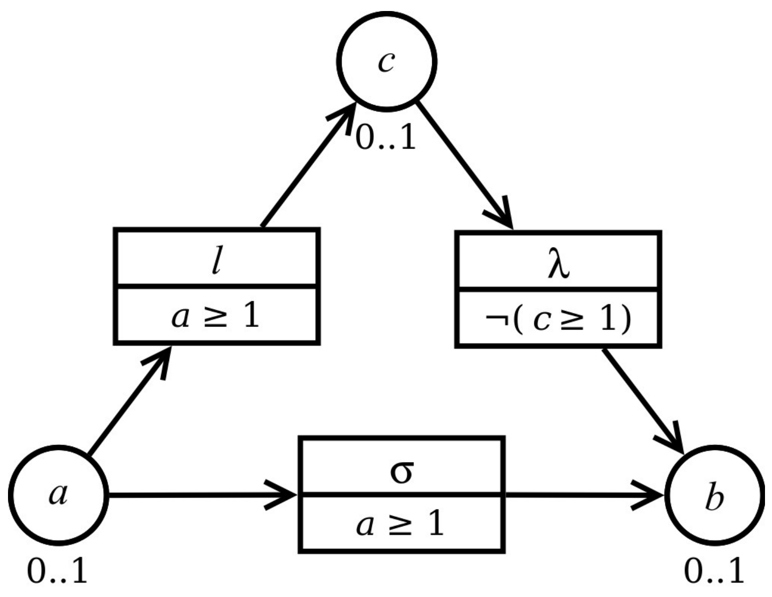
$$\{\ \Phi \wedge a = 1 \wedge c = 0\ \}\ c+\ \{\ a = 1 \wedge c = 1\ \}$$

$$\Phi \equiv c \geq 0 \wedge c < 1$$

$$\neg\varphi_l \Rightarrow k_{c,\varnothing} > c$$

$$\varphi_l \Rightarrow k_{c,\{l\}} > c$$

où : $\varphi_l \equiv (a \geq 1)$



Application au modèle de Thomas

[6]

Exemple d'application :

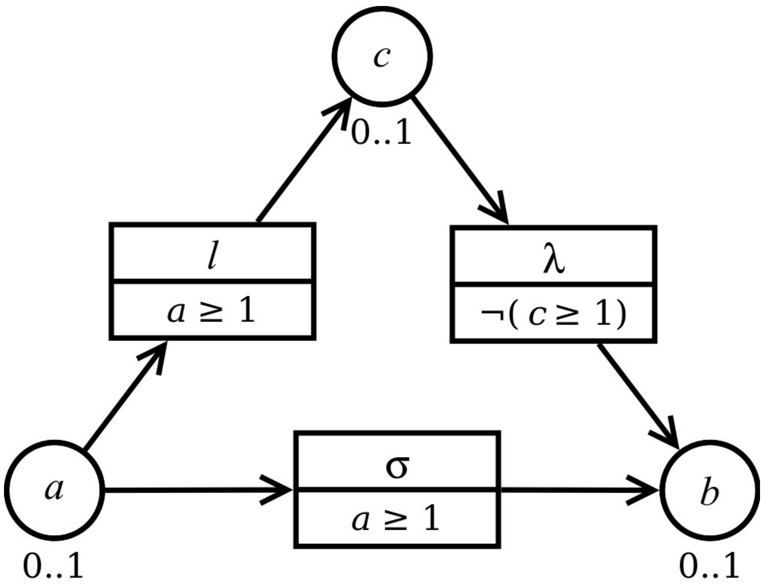
$$\{ \ \Phi \wedge a = 1 \wedge c = 0 \ \} \ c+ \ \{ \ a = 1 \wedge c = 1 \ \}$$

$$\Phi \equiv c \geq 0 \wedge c < 1$$

$$\neg\varphi_l \Rightarrow k_{c,\emptyset} > c$$

$$\varphi_l \Rightarrow k_{c,\{l\}} > c$$

où : $\varphi_l \equiv vrai$



Application au modèle de Thomas

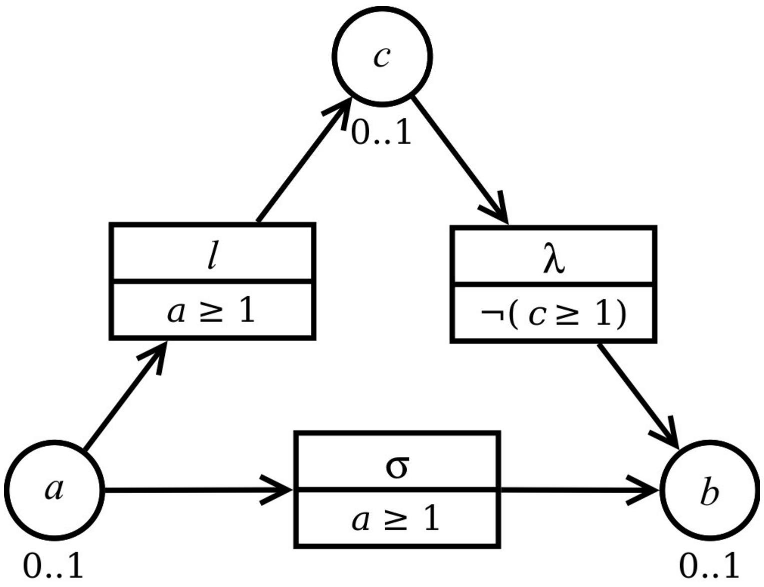
[6]

Exemple d'application :

$$\{\ \Phi \wedge a = 1 \wedge c = 0\ \}\ c+\ \{\ a = 1 \wedge c = 1\ \}$$

$$\Phi \equiv c \geq 0 \wedge c < 1$$
$$\neg\varphi_l \Rightarrow k_{c,\emptyset} > c$$
$$\varphi_l \Rightarrow k_{c,\{l\}} > c$$

où : $\varphi_l \equiv vrai$



Application au modèle de Thomas

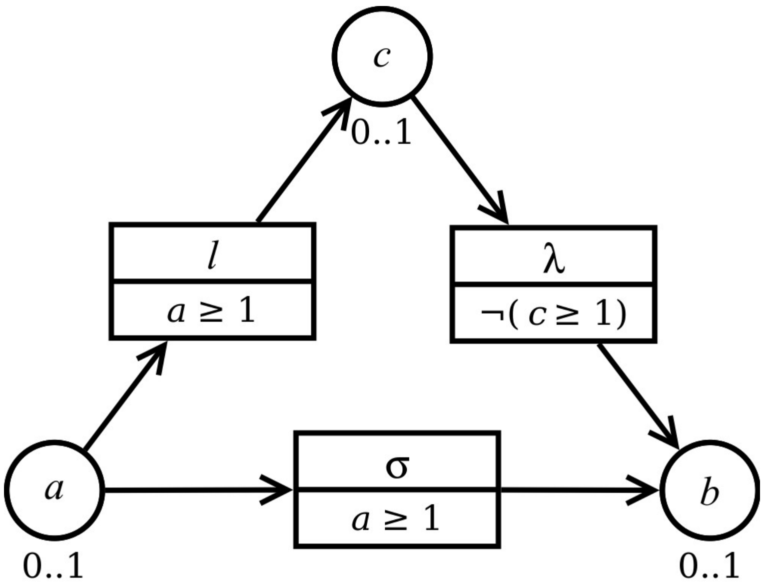
[6]

Exemple d'application :

$$\{\ \Phi \wedge a = 1 \wedge c = 0\ \}\ c+\ \{\ a = 1 \wedge c = 1\ \}$$

$$\Phi \equiv \begin{array}{l} c \geq 0 \wedge c < 1 \\ \neg\varphi_l \Rightarrow k_{c,\emptyset} > c \\ \varphi_l \Rightarrow k_{c,\{l\}} > c \end{array}$$

où : $\varphi_l \equiv vrai$



Application au modèle de Thomas

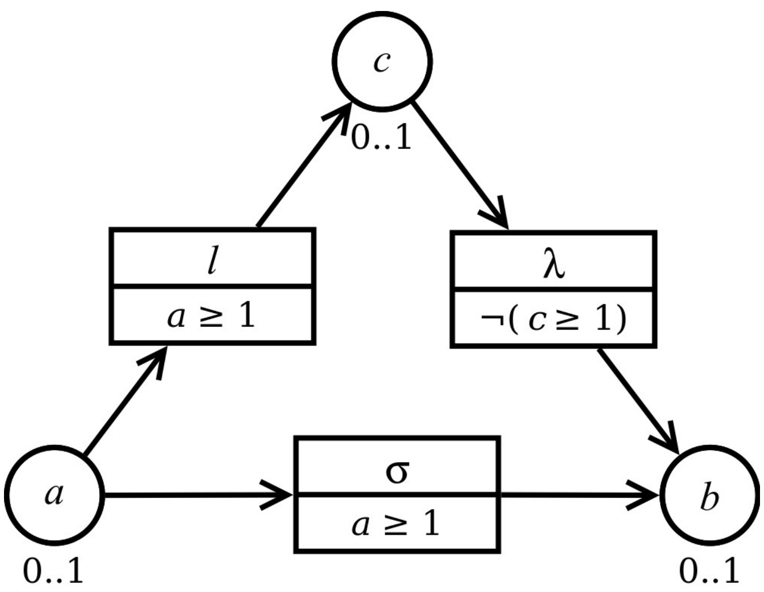
[6]

Exemple d'application :

$\{ \,\Phi \wedge a = 1 \wedge c = 0\, \}\; c+ \;\{\, a = 1 \wedge c = 1 \,\}$

$\Phi \equiv$ $c \geq 0 \wedge c < 1$
$\neg\varphi_l \Rightarrow k_{c,\emptyset} > c$
$\varphi_l \Rightarrow k_{c,\{l\}} > 0$

où : $\varphi_l \equiv vrai$



Application au modèle de Thomas

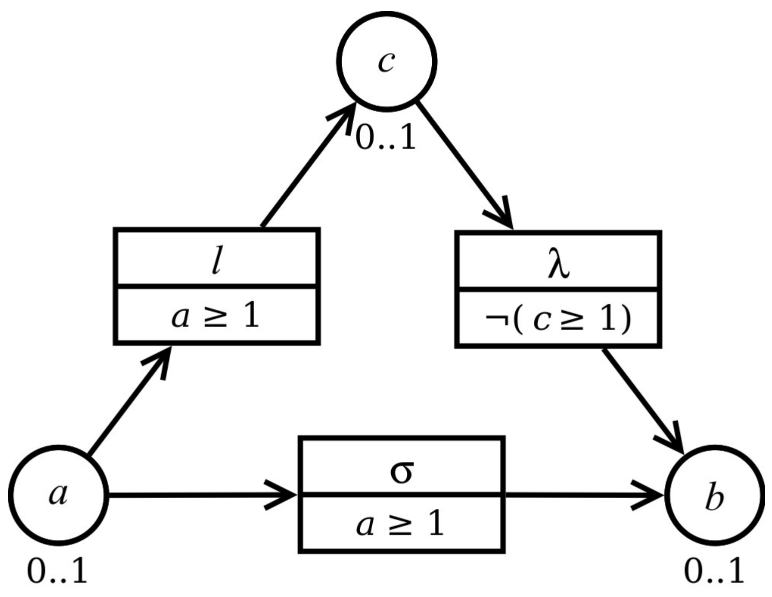
[6]

Exemple d'application :

$$\{\ \Phi \wedge a = 1 \wedge c = 0\ \}\ c+\ \{\ a = 1 \wedge c = 1\ \}$$

$$\begin{aligned} \Phi \equiv\ & c \geq 0 \wedge c < 1 \\ & \neg\varphi_l \Rightarrow k_{c,\emptyset} > c \\ & \varphi_l \Rightarrow k_{c,\{l\}} > 0 \end{aligned}$$

où : $\varphi_l \equiv vrai$



$$\Phi \equiv k_{c,\{l\}} = 1$$

Discussion

Inconvénient du modèle de Thomas :

Beaucoup de paramétrisations possibles

Utilisation de la logique de Hoare :

permet l'inférence de paramètres biologiques *via* l'opérateur de plus faible pré-condition

Avantages :

- Pas de recherche exhaustive
- Dispense de construire le graphe d'états

Présentation de Coq et pistes d'implémentation

Assistant de preuves formelles

→ Permet d'effectuer des démonstrations mathématiques de façon formelle

Langage : Gallina

→ Programmation fonctionnelle

Bases de l'implémentation :

→ Tutorial on Hoare Logic de Sylvain Boulmé
→ Software Foundations de Benjamin C. Pierce

Présentation de Coq

Définitions inductives :

```
Inductive nat : Type :=
  | O : nat
  | S : nat -> nat.
```

- Créent de nouveaux types
- Définitions syntaxiques (sans sémantique)
  → la sémantique vient avec l'utilisation

Présentation de Coq

Définitions fonctionnelles :

```
Definition deux : nat :=
  S (S O).

Definition plus_deux (n : nat) : nat :=
  S (S n).
```

- Créent de nouveaux objets à partir d'objets existants
- Programmation centrée sur le résultat du calcul (et non sur l'impact sur la machine)

Présentation de Coq

Définitions de points fixes :

```
Fixpoint plus (n : nat) (m : nat) : nat :=
  match n with
    | O => m
    | S n' => S (plus n' m)
  end.
```

$$n + m \Rightarrow (n-1) + m \Rightarrow (n-2) + m \Rightarrow \ldots \Rightarrow 0 + m$$

- Permettent des définitions récursives
- Assurance de terminaison requise
  (décroissance structurelle d'un argument)

Présentation de Coq

Propriétés

```
Definition p1 : Prop :=
  2 + 2 = 4.
Definition p2 : Prop :=
  2 + 2 = 22.
Definition p3 : Prop :=
  forall (n:nat), 0 + n = n.
```

- Propriétés mathématiques
- Sans valeur de vérité (cf `p2`)
- Une fonction peut retourner une propriété

Présentation de Coq

Preuves :

- Débutées lorsqu'un théorème est formulé :

```
Theorem obvious : forall n:nat,
  plus_deux n = deux -> n = O.
Proof.
```

- Environnement de preuve :

```
n : nat
H : plus_deux n = deux
============================
 n = O
```

Contexte (hypothèses)

Objectif(s)

- Utilisation de tactiques

  Développer, Simplifier, Reformuler, Conclure…

Implémentation avec Coq

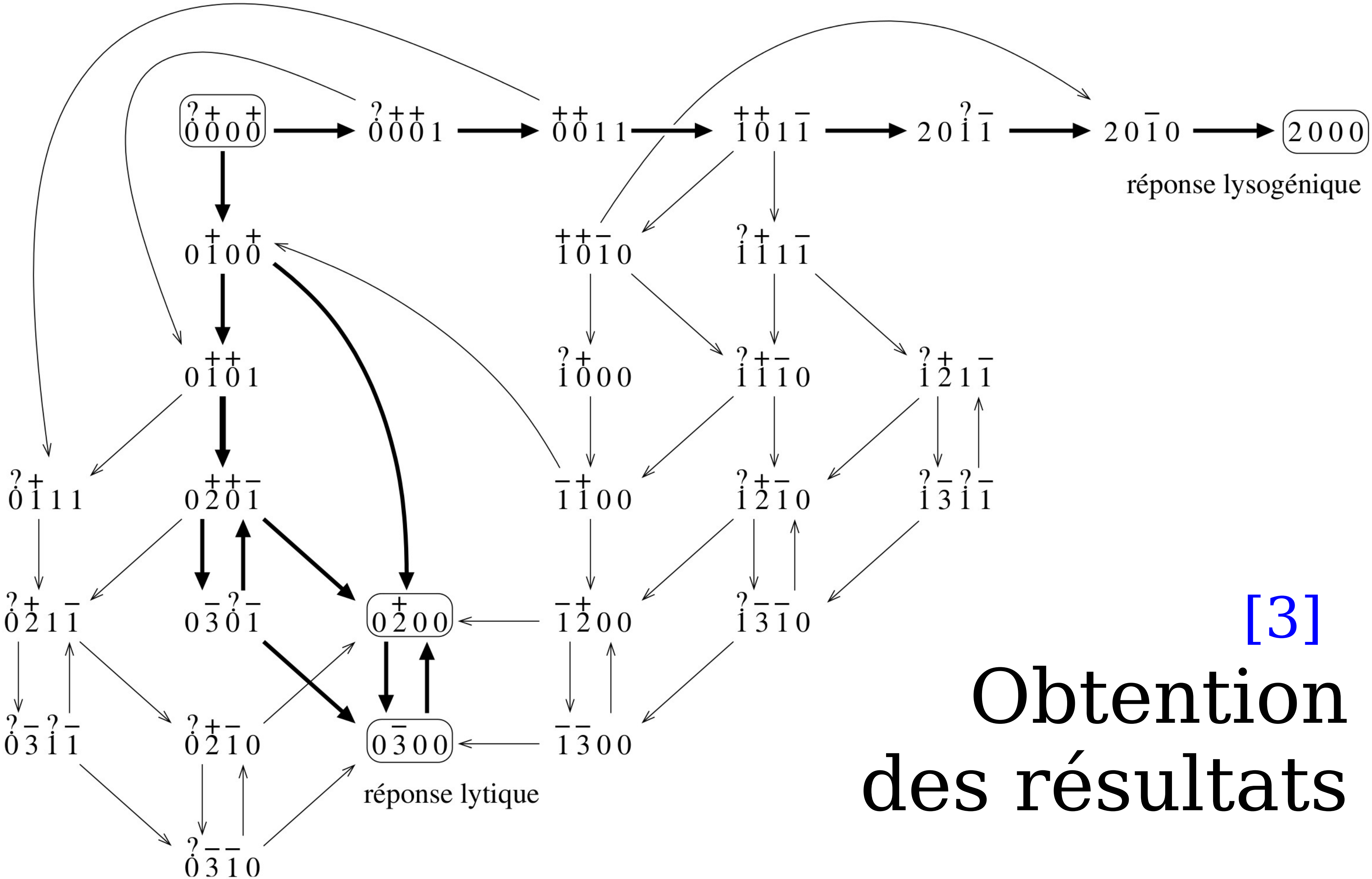
Modèle de Thomas

- Variables / multiplexes :
  constructeurs (abstraction)
  relation d'ordre (utilisation)
- Environnements : **listes**

```
[ 1 ; 0 ; 0 ]
  a   b   c
```

Logique de Hoare

- Prédicats : **fonctions** `env -> Prop`
- Programmes : **définition syntaxique**
- Calcul de précondition : **fonction récursive**
- Résultats : **prédicats**



[3]

Obtention des résultats

Obtention des résultats [3]

```
  (*** Réponse lysogénique du phage lambda ***)

  (* Programme *)
Definition prog_lyso :=
  N++ ;; CII++ ;; CI++ ;; CI++ ;; N-- ;; CII--.

  (* Post-condition *)
Definition post_lyso := fun (e:env) =>
  (CI = 2 /\ Cro = 0 /\ CII = 0 /\ N = 0).

  (* Calcul de la plus faible pré-condition *)
Definition pre_lyso := wp prog_lyso post_lyso.

  (* Évaluation *)
Eval compute in pre_lyso.
```

Obtention des résultats

```
((((((CI' + 1 + 1 = 2 /\
    Cro' = 0 /\ CII' + 1 - 1 = 0 /\ N' + 1 - 1 = 0) /\
   1 <= CII' + 1 /\
   CII' + 1 <= 1 /\
   (eval_formula (NEG (ATOMV CI 2))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false /\
    eval_formula (NEG (ATOMV Cro 3))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false /\
    eval_formula (ATOMV N 1)
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false ->
    S (K CII []) <= CII' + 1) /\
   (eval_formula (NEG (ATOMV CI 2))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = true /\
    eval_formula (NEG (ATOMV Cro 3))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false /\
    eval_formula (ATOMV N 1)
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false ->
    S (K CII [m3]) <= CII' + 1) /\
   (eval_formula (NEG (ATOMV CI 2))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false /\
    eval_formula (NEG (ATOMV Cro 3))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = true /\
    eval_formula (ATOMV N 1)
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false ->
    S (K CII [m4]) <= CII' + 1) /\
   (eval_formula (NEG (ATOMV CI 2))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false /\
    eval_formula (NEG (ATOMV Cro 3))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = false /\
    eval_formula (ATOMV N 1)
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = true ->
    S (K CII [m8]) <= CII' + 1) /\
   (eval_formula (NEG (ATOMV CI 2))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = true /\
    eval_formula (NEG (ATOMV Cro 3))
      [CI' + 1 + 1; Cro'; CII' + 1; N' + 1 - 1] = true /\
```

[...]

Obtention des résultats

```
      [...]
H26 : true = false -> true = false -> 2 <= K CI []
H18 : true = false -> 1 <= K CI [m2]
H53 : 2 <= K CI [m2; m5]
H28 : true = false -> S (K N []) <= 1
H42 : 1 <= K CI [m2; m5]
H14 : true = false -> 1 <= K CII [m3; m4]
H13 : true = false -> true = false -> 1 <= K CI []
H43 : true = false -> true = true -> 2 <= K CI [m5]
H21 : 1 <= K CII [m3; m4; m8]
H35 : true = false -> true = true -> 1 <= K CI [m5]
H34 : S (K N [m7]) <= 1
H17 : 1 <= K N [m6; m7]
H49 : false = true -> S (K CII [m4; m8]) <= 1
H36 : S (K CII [m4]) <= 1
      [...]
```

Obtention des résultats

```
      [...]
H26 : true = false -> true = false -> 2 <= K CI []
H18 : true = false -> 1 <= K CI [m2]
H53 : 2 <= K CI [m2; m5]
H28 : true = false -> S (K N []) <= 1
H42 : 1 <= K CI [m2; m5]
H14 : true = false -> 1 <= K CII [m3; m4]
H13 : true = false -> true = false -> 1 <= K CI []
H43 : true = false -> true = true -> 2 <= K CI [m5]
H21 : 1 <= K CII [m3; m4; m8]
H35 : true = false -> true = true -> 1 <= K CI [m5]
H34 : S (K N [m7]) <= 1
H17 : 1 <= K N [m6; m7]
H49 : false = true -> S (K CII [m4; m8]) <= 1
H36 : S (K CII [m4]) <= 1
      [...]
```

Obtention des résultats

```
      [...]
H26 : true = false -> true = false -> 2 <= K CI []
H18 : true = false -> 1 <= K CI [m2]
H53 : 2 <= K CI [m2; m5]
H28 : true = false -> S (K N []) <= 1
H42 : 1 <= K CI [m2; m5]
H14 : true = false -> 1 <= K CII [m3; m4]
H13 : true = false -> true = false -> 1 <= K CI []
H43 : true = false -> true = true -> 2 <= K CI [m5]
H21 : 1 <= K CII [m3; m4; m8]
H35 : true = false -> true = true -> 1 <= K CI [m5]
H34 : S (K N [m7]) <= 1
H17 : 1 <= K N [m6; m7]
H49 : false = true -> S (K CII [m4; m8]) <= 1
H36 : S (K CII [m4]) <= 1
      [...]
```

Résultats sur le phage lambda [3]

→ Informations sur la paramétrisation
à partir des chemins « biologiquement réalistes »

$$
\begin{aligned}
\kappa_{\mathrm{cI}}(\emptyset) &= 0,\ 1 \text{ ou } \mathbf{2}\\
\kappa_{\mathrm{cI}}(\{\mathrm{cI}\}) &= [\mathbf{2}]\\
\kappa_{\mathrm{cI}}(\{\mathrm{cro}\}) &= [\mathbf{0}]\\
\kappa_{\mathrm{cI}}(\{\mathrm{cII}\}) &= [\![\mathbf{2}]\!]\\
\kappa_{\mathrm{cI}}(\{\mathrm{cI}, \mathrm{cro}\}) &= (\mathbf{0}),\ 1 \text{ ou } 2\\
\kappa_{\mathrm{cI}}(\{\mathrm{cI}, \mathrm{cII}\}) &= [\![\mathbf{2}]\!]\\
\kappa_{\mathrm{cI}}(\{\mathrm{cro}, \mathrm{cII}\}) &= 0, 1 \text{ ou } \mathbf{2}\\
\kappa_{\mathrm{cI}}(\{\mathrm{cI}, \mathrm{cro}, \mathrm{cII}\}) &= (0),\ 1 \text{ ou } \mathbf{2}
\end{aligned}
$$

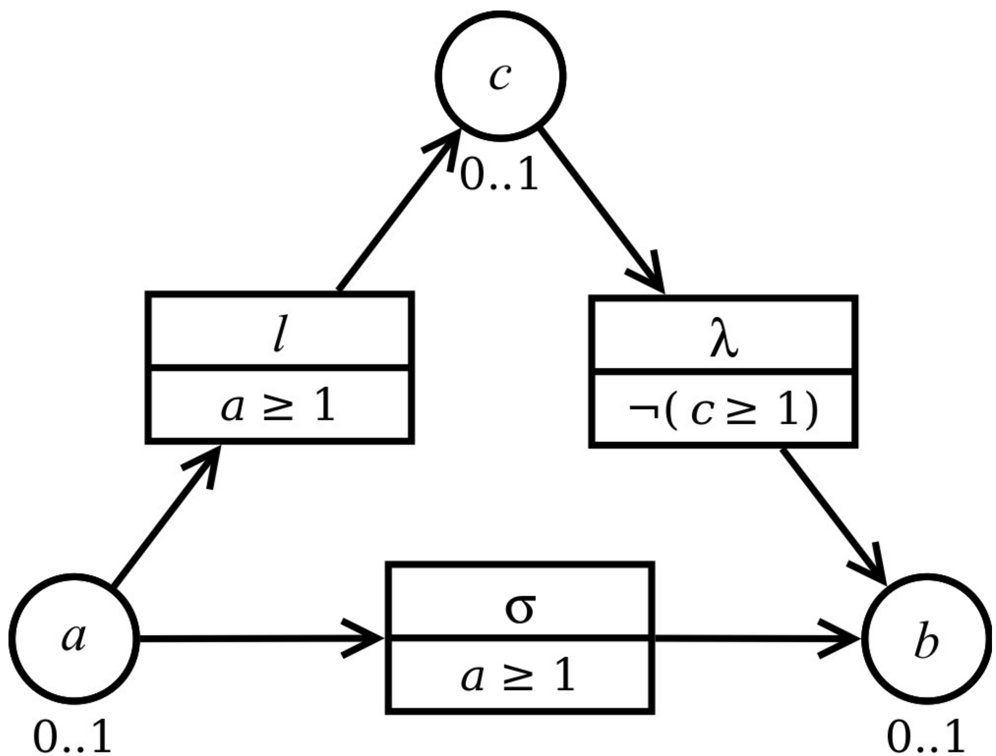
$$
\begin{aligned}
\kappa_{\mathrm{cro}}(\emptyset) &= [\mathbf{3}]\\
\kappa_{\mathrm{cro}}(\{\mathrm{cI}\}) &= [\mathbf{0}]\\
\kappa_{\mathrm{cro}}(\{\mathrm{cro}\}) &= (0),\ (1) \text{ ou } [\mathbf{2}]\\
\kappa_{\mathrm{cro}}(\{\mathrm{cI}, \mathrm{cro}\}) &= (\mathbf{0}),\ (1) \text{ ou } [\![2]\!]
\end{aligned}
$$

$$
\begin{aligned}
\kappa_{\mathrm{cII}}(\emptyset) &= [\mathbf{0}]\\
\kappa_{\mathrm{cII}}(\{\mathrm{cI}\}) &= [\mathbf{0}]\\
\kappa_{\mathrm{cII}}(\{\mathrm{cro}\}) &= [\mathbf{0}]\\
\kappa_{\mathrm{cII}}(\{\mathrm{N}\}) &= [\![\mathbf{1}]\!]\\
\kappa_{\mathrm{cII}}(\{\mathrm{cI}, \mathrm{cro}\}) &= [\![\mathbf{0}]\!]\\
\kappa_{\mathrm{cII}}(\{\mathrm{cI}, \mathrm{N}\}) &= \mathbf{0} \text{ ou } 1\\
\kappa_{\mathrm{cII}}(\{\mathrm{cro}, \mathrm{N}\}) &= \mathbf{0} \text{ ou } 1\\
\kappa_{\mathrm{cII}}(\{\mathrm{cI}, \mathrm{cro}, \mathrm{N}\}) &= [\![\mathbf{0}]\!]
\end{aligned}
$$

$$
\begin{aligned}
\kappa_{\mathrm{N}}(\emptyset) &= [\![\mathbf{1}]\!]\\
\kappa_{\mathrm{N}}(\{\mathrm{cI}\}) &= [\mathbf{0}]\\
\kappa_{\mathrm{N}}(\{\mathrm{cro}\}) &= [\mathbf{0}]\\
\kappa_{\mathrm{N}}(\{\mathrm{cI}, \mathrm{cro}\}) &= [\![\mathbf{0}]\!]
\end{aligned}
$$

Résultats du papier [6]

But : retrouver les résultats du papier
« *A Hoare logic to identify parameter values of discrete models of gene regulatory networks* »



Résultats du papier [6]

$$\mathrm{I} \quad \text{— WP}\,(b+\,;\,c+\,;\,b-,\ a = 1 \wedge b = 0 \wedge c = 1)$$
$$\Rightarrow\ k_{b,\{\sigma,\,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$$

Résultats du papier [6]

I — WP ($b+$; $c+$; $b-$, $a = 1 \wedge b = 0 \wedge c = 1$)

✔ $\Rightarrow k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$

Résultats du papier [6]

I — WP ($b+$; $c+$; $b-$, $a = 1 \wedge b = 0 \wedge c = 1$)

✔ $\Rightarrow\ k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$

II — WP ($b+$; $b-$, Vrai)

$\Rightarrow$ Faux

Résultats du papier [6]

I — WP ($b+ \,;\, c+ \,;\, b-$, $a = 1 \wedge b = 0 \wedge c = 1$)

$\Rightarrow \; k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$

II — WP ($b+ \,;\, b-$, Vrai)

$\Rightarrow$ Faux

Résultats du papier [6]

I — WP ($b+$; $c+$; $b-$, $a = 1 \wedge b = 0 \wedge c = 1$)

✓ $\Rightarrow\ k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$

II — WP ($b+$; $b-$, Vrai)

✓ $\Rightarrow$ Faux

III — { $a = 1 \wedge b = 0 \wedge c = 0 \wedge$
$k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$ }
While ($b < 1$) With (I) Do $\exists(b+, b-, c+, c-)$
{ $b = 1$ } n'est pas un triplet valide

Résultats du papier [6]

I — WP ($b+$; $c+$; $b-$, $a = 1 \land b = 0 \land c = 1$)

$\Rightarrow\ k_{b,\{\sigma,\,\lambda\}} = 1 \land k_{c,\{l\}} = 1 \land k_{b,\{\sigma\}} = 0$

II — WP ($b+$; $b-$, Vrai)

$\Rightarrow$ Faux

III — { $a = 1 \land b = 0 \land c = 0 \land$

$k_{b,\{\sigma,\,\lambda\}} = 1 \land k_{c,\{l\}} = 1 \land k_{b,\{\sigma\}} = 0$ }

While ($b < 1$) With (I) Do $\exists(b+, b-, c+, c-)$

{ $b = 1$ } n'est pas un triplet valide

Conclusion sur Coq

- Les premiers résultats portent
  sur des programmes « simples » (sans boucle)
- Les résultats sont en accord
  avec les résultats obtenus sur papier
- Résultats partiels
- Implémentation incomplète

Présentation d'OCaml et pistes d'implémentation

But : Produire davantage de résultats

Langage de programmation

→ Programmation fonctionnelle
→ Langage proche de Gallina,
mais plus souple et plus classique

Bases de l'implémentation :

→ Travail déjà réalisé avec Coq

Présentation d'OCaml

Définitions :

```
let plus_deux n = n + 2 ;;
```

- Outils puissants
- Plus de souplesse :
  - Exceptions
  - Définitions récursives moins contraignantes
  - Caractéristiques impératives
- Traduction aisée depuis Gallina (syntaxe)

Implémentation avec OCaml

Modèle de Thomas

- Variables / multiplexes : **chaînes de caractères**
- Environnements : **listes d'association**

```
[ ("a", 1) ; ("b", 0) ; ("c", 0) ]
```

Logique de Hoare

- Conditions : **fonctions** `env -> bool`
- Programmes : **définition syntaxique**
- Calcul de précondition : **fonction récursive**
- Résultats : **ensemble de solutions (environnement + paramétrisation)**

Obtention des résultats

```
  (* Programme *)
let prog_ex = (* b+ ; c+ ; b- *)
 Iseq (Iseq (Iincr "b", Iincr "c"), Idecr "b") ;;

  (* Post-condition *)
let post_ex = fun p e -> get "b" e = 0 ;;

  (* Calcul de la plus faible pré-condition *)
let pre_wp_ex = synt_wp prog post ;;
```

Obtention des résultats

```
  (* Programme *)
let prog_ex = (* b+ ; c+ ; b- *)
 Iseq (Iseq (Iincr "b", Iincr "c"), Idecr "b") ;;

  (* Post-condition *)
let post_ex = fun p e -> get "b" e = 0 ;;

  (* Calcul de la plus faible pré-condition *)
let pre_wp_ex = synt_wp prog post ;;

  (* Raffinement *)
let pre_ex = fun p e -> (pre_wp_ex p e) &&
 (get "a" e = 1 && get "b" e = 0 && get "c" e = 0) ;;

  (* Résolution *)
let solution = solvevp pre_ex ;;
```

Obtention des résultats

```
16 solutions
a=1; b=0; c=0;
a/{}=0; b/{}=0; b/{lambda}=0; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=1; b/{}=0; b/{lambda}=0; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=0; b/{}=1; b/{lambda}=0; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=1; b/{}=1; b/{lambda}=0; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=0; b/{}=0; b/{lambda}=1; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=1; b/{}=0; b/{lambda}=1; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=0; b/{}=1; b/{lambda}=1; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=1; b/{}=1; b/{lambda}=1; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=0; c/{l}=1;
a=1; b=0; c=0;
a/{}=0; b/{}=0; b/{lambda}=0; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=1; c/{l}=1;
a=1; b=0; c=0;
a/{}=1; b/{}=0; b/{lambda}=0; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=1; c/{l}=1;
a=1; b=0; c=0;
a/{}=0; b/{}=1; b/{lambda}=0; b/{lambda,sigma}=1; b/{sigma}=0; c/{}=1; c/{l}=1;
a=1; b=0; c=0;

                                                                   [...]
```

Résultats du papier [6]

I — WP ($b+ ; c+ ; b-$, $a = 1 \wedge b = 0 \wedge c = 1$)

$$\Rightarrow \; k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$$

II — WP ($b+ ; b-$, Vrai)

$$\Rightarrow \; \text{Faux}$$

III — { $a = 1 \wedge b = 0 \wedge c = 0 \wedge$
$k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$ }
While ($b < 1$) With (I) Do $\exists(b+, b-, c+, c-)$
{ $b = 1$ } n'est pas un triplet valide

Résultats du papier [6]

I — WP ($b+ ; c+ ; b-$, $a = 1 \wedge b = 0 \wedge c = 1$)

$\Rightarrow k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$

✔ (16 solutions)

II — WP ($b+ ; b-$, Vrai)

$\Rightarrow$ Faux

III — { $a = 1 \wedge b = 0 \wedge c = 0 \wedge$

$k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$ }

While ($b < 1$) With (I) Do $\exists(b+, b-, c+, c-)$

{ $b = 1$ } n'est pas un triplet valide

Résultats du papier [6]

I — WP ($b+$; $c+$; $b-$, $a = 1 \wedge b = 0 \wedge c = 1$)

$\Rightarrow$ $k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$

(16 solutions)

II — WP ($b+$; $b-$, Vrai)

$\Rightarrow$ Faux (aucune solution)

III — { $a = 1 \wedge b = 0 \wedge c = 0 \wedge$

$k_{b,\{\sigma,\lambda\}} = 1 \wedge k_{c,\{l\}} = 1 \wedge k_{b,\{\sigma\}} = 0$ }

While ($b < 1$) With (I) Do $\exists(b+, b-, c+, c-)$

{ $b = 1$ } n'est pas un triplet valide

Résultats du papier [6]

I — WP ($b+ \; ; \; c+ \; ; \; b-$, $a = 1 \land b = 0 \land c = 1$)

$\Rightarrow \; k_{b,\{\sigma,\lambda\}} = 1 \land k_{c,\{l\}} = 1 \land k_{b,\{\sigma\}} = 0$

(16 solutions)

II — WP ($b+ \; ; \; b-$, Vrai)

$\Rightarrow$ Faux (aucune solution)

III — { $a = 1 \land b = 0 \land c = 0 \land$

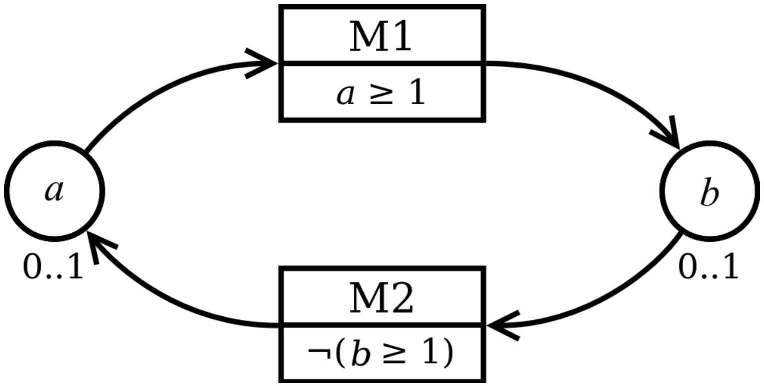
$k_{b,\{\sigma,\lambda\}} = 1 \land k_{c,\{l\}} = 1 \land k_{b,\{\sigma\}} = 0$ }

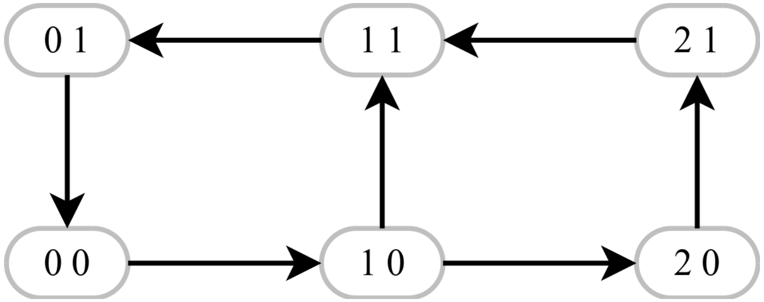
While ($b < 1$) With (I) Do $\exists(b+, b-, c+, c-)$

{ $b = 1$ } n'est pas un triplet valide

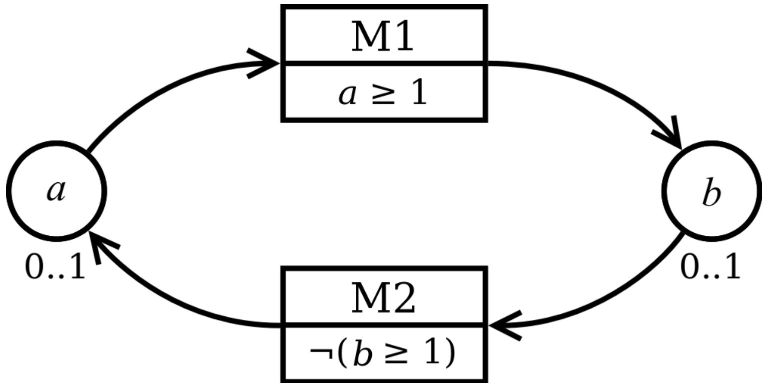
(aucune solution)

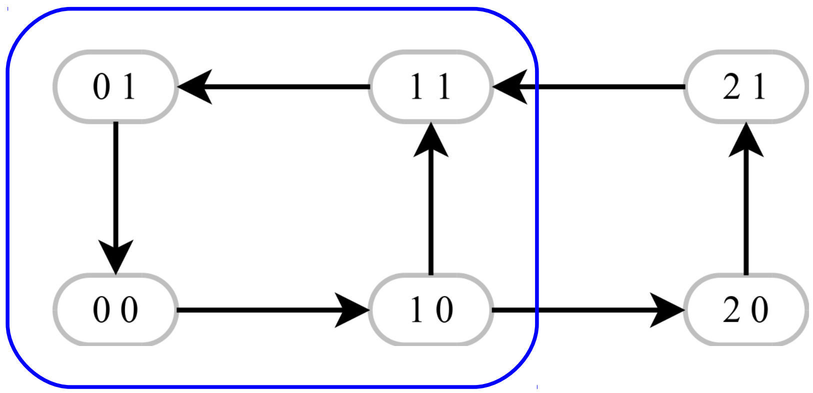
Résultats complémentaires [1]





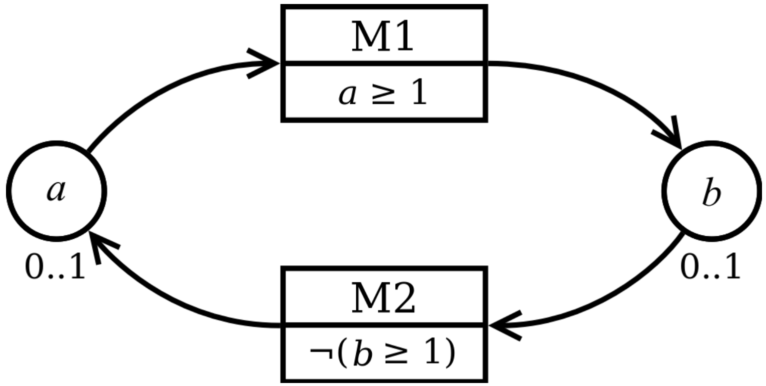
Résultats complémentaires [1]

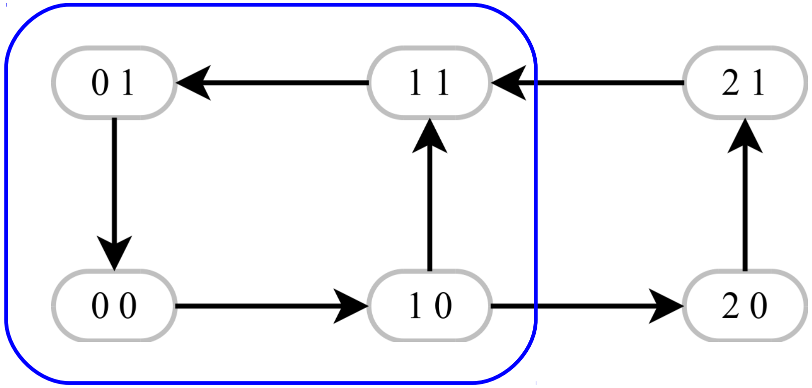




While (*vrai*) With (*Inv*) Do ($a+$; $b+$; $a-$; $b-$)

Résultats complémentaires [1]

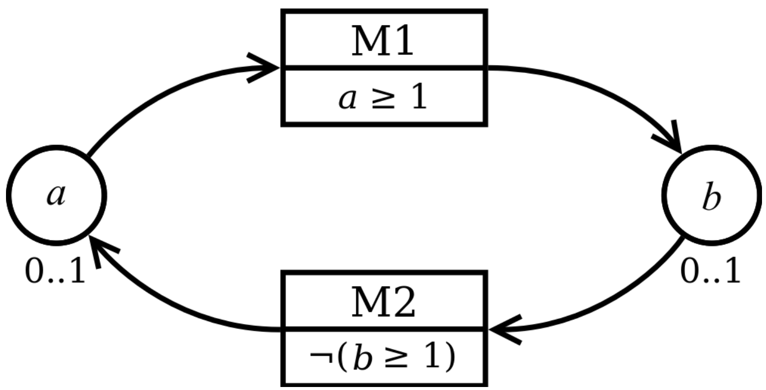


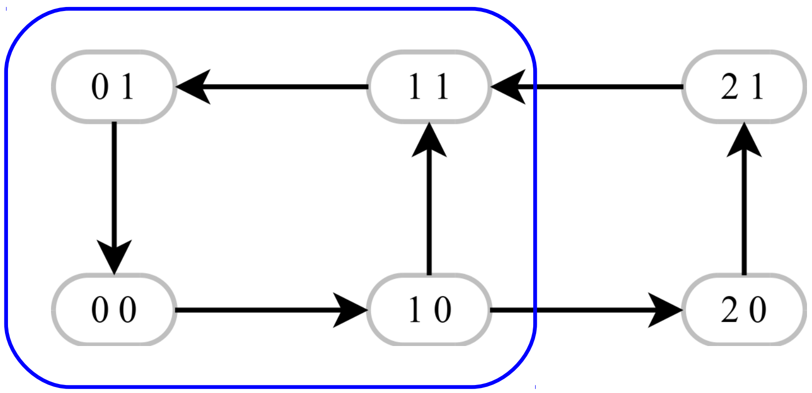


While ($vrai$) With (Inv) Do ($a+$; $b+$; $a-$; $b-$)

- $Inv \equiv vrai$

Résultats complémentaires [1]

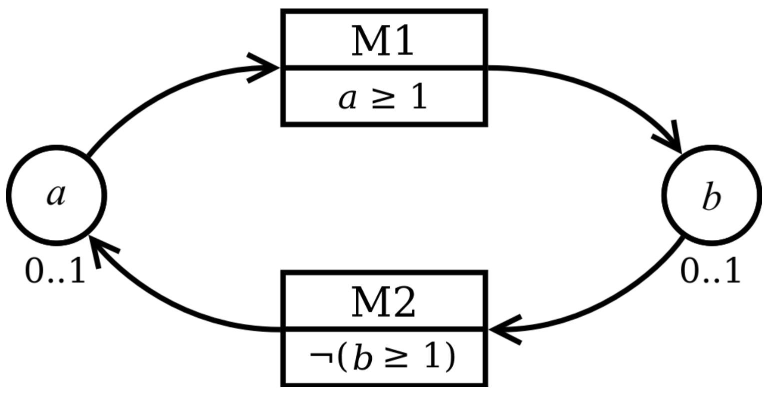


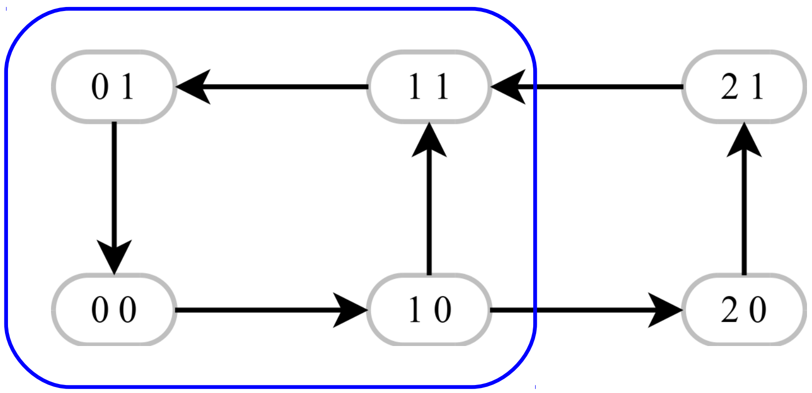


While (*vrai*) With (*Inv*) Do ($a+$; $b+$; $a-$; $b-$)

- $Inv \equiv vrai$
  → aucune solution ✗

Résultats complémentaires [1]

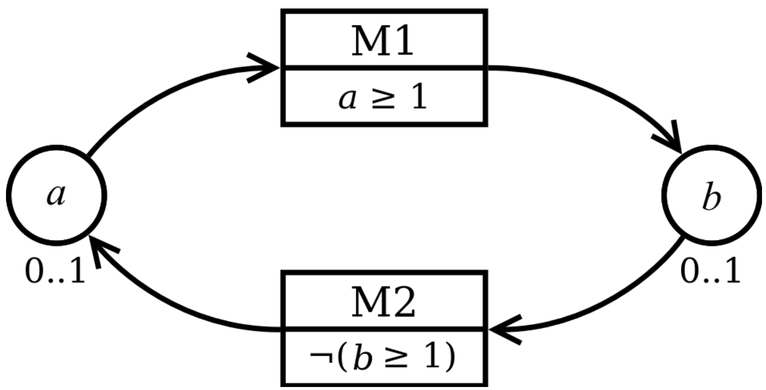


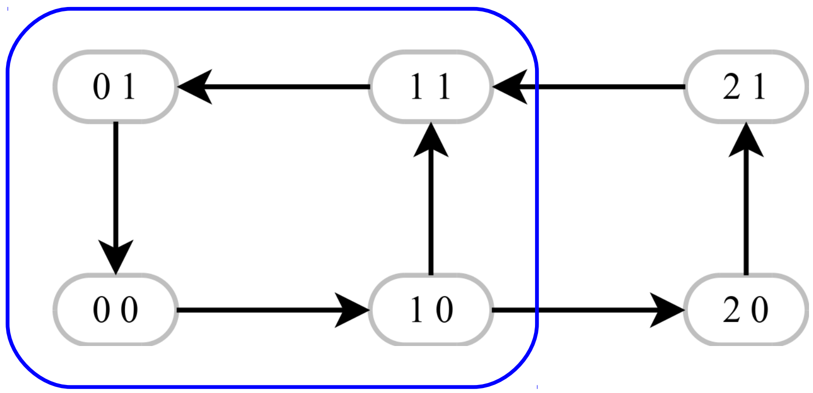


While (*vrai*) With (*Inv*) Do ($a+$; $b+$; $a-$; $b-$)

- $Inv \equiv vrai$
  → aucune solution ✗
- $Inv \equiv a = 0 \wedge b = 0$

Résultats complémentaires [1]

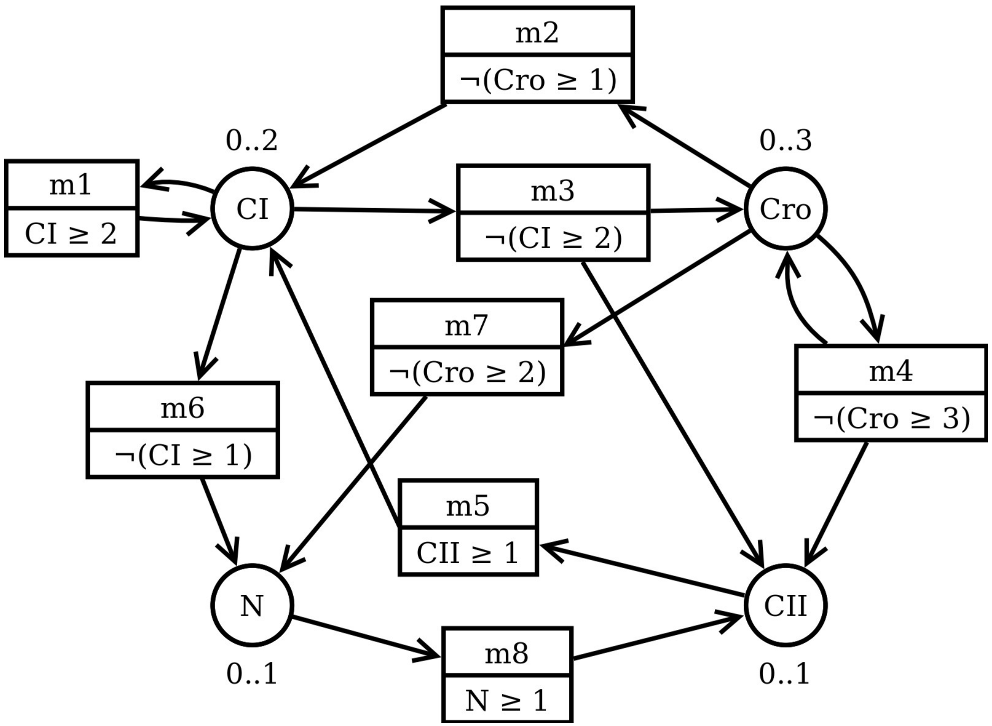




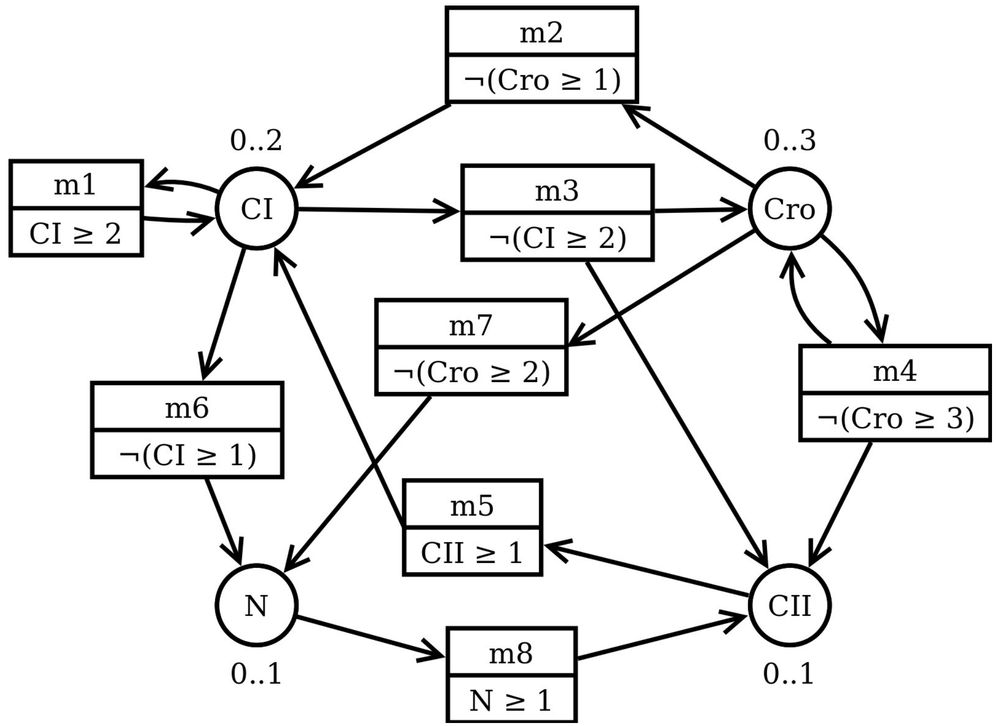
While (*vrai*) With (*Inv*) Do ($a+$; $b+$; $a-$; $b-$)

- $Inv \equiv vrai$
  → aucune solution
- $Inv \equiv a = 0 \wedge b = 0$
  → 2 solutions

Résultats complémentaires [3]



Résultats complémentaires [3]



330 225 942 528 possibilités
→ Dépassement de capacité

Conclusion sur OCaml

- Approche différente : recherche exhaustive
- Amené à évoluer
- Des résultats sur tous les types de programmes
- Les résultats sont en accord
  avec les résultats obtenus sur papier
- Problème de l'explosion combinatoire

Discussion

Définition des environnements

Environnement = état du réseau de régulation

Coq :

- Listes de taille variable
  Problème : définition trop lâche
- Listes de taille fixée
  Problème : complexité supplémentaire

Ocaml :

- Listes d'association
  → Nécessitent d'être vérifiées

Discussion

Sémantique du langage impératif

Définit le comportement du langage de chemins

Coq : La sémantique est utilisée pour les preuves de complétude et correction

Problème : difficultés pour la définir (indéterminisme)

OCaml : Pas de sémantique

Discussion

Complexité et explosion combinatoire

L'utilisation de la logique de Hoare permet d'éviter une partie de la complexité

Coq : Résultats sous forme de propriétés
→ En accord avec la théorie

OCaml : Recherche exhaustive de solutions
Problème : La complexité n'est pas contournée
→ Recréer un environnement de propriétés ?

Discussion

Composante temporelle

Coq : Nécessite une assurance de terminaison du programme étudié (variant décroissant)

Problème : Très difficile à implémenter

OCaml : Possibilité de lancer une récursion infinie

→ Rapide sur des programmes simples

→ Possibilités d'amélioration

Évolutions

Améliorations supplémentaires

- Rechercher de nouveaux exemples d'application

Coq :
- Solveur pour automatiser les simplifications
- Compléter la sémantique

OCaml :
- Problèmes de complexité (solveur)
- Faire le lien avec le format GINML

Conclusion

Modèle de Thomas : Modèle puissant mais difficile à étudier pour des graphes de grande taille

Logique de Hoare : Offre une nouvelle approche pour la déduction de paramètres biologiques

Deux implémentations :

- **Coq :** Calcul des plus faibles pré-conditions
  Nombreux obstacles, incomplet
- **OCaml :** Recherche exhaustive
  Explosion combinatoire, mais résultats

Merci pour votre attention

Bibliographie • Modèle de Thomas

- [1] A. Richard, J.-P. Comet, G. Bernot :
R. Thomas' logical method.
Avril 2008. Tutorials on modelling methods and tools : Modelling a genetic switch and Metabolic Networks, Spring School on Modelling Complex Biological Systems in the Context of Genomics.

- [2] G. Bernot, J.-P. Comet, Z. Khalis :
Gene regulatory networks with multiplexes.
European Simulation and Modelling Conference Proceedings, pages 423–432, France, octobre 2008.

- [3] A. Richard :
Modèle formel pour les réseaux de régulation génétique & Influence des circuits de rétroaction.
Thèse sous la direction de J.-P. Comet et G. Bernot.
Sections 5.4.2–5.4.4, pages 87–95, France, septembre 2006.

Bibliographie • Logique de Hoare

- [4] C. A. R. Hoare :
An axiomatic basis for computer programming.
Communications of the ACM, 12, pages 576–580, octobre 1969.

- [5] E. W. Dijkstra :
Guarded commands, nondeterminacy and formal derivation of programs.
Communications of the ACM, 18 :453–457, Aug. 1975.

- [6] Z. Khalis, G. Bernot, J.-P. Comet, A. Richard, O. Roux, H. Siebert : A hoare logic to identify parameter values of discrete models of gene regulatory networks.
Document de travail.