Modelling & analysing of gene networks: A genetically modified Hoare logic

If only Tony Hoare and René Thomas had met in the 70s. . .

Gilles Bernot

University Nice Sophia Antipolis, I3S laboratory, France

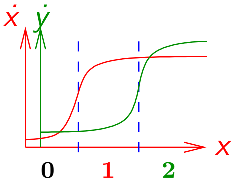
Joint work with Jean-Paul Comet and Olivier Roux

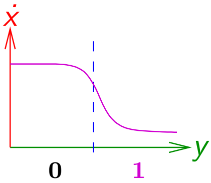
Acknowledgments: Epigenomics Project Genopole®
as well as A. Richard, Z. Khalis and J. Behaegel

Menu

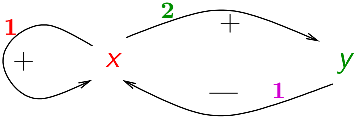
1. Extended Thomas' discrete gene networks
2. Reminders on standard Hoare logic
3. Assertions and Trace specifications
4. Genetically modified Hoare logic
5. Examples

R. Thomas Discrete Gene Networks





In each state,
a variable v tries to
go toward the interval
numbered $K_{v,\omega}$



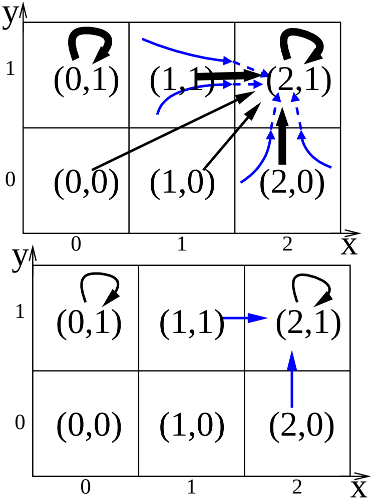
No help :	K_x	K_y
x helps :	$K_{x,x}$	$K_{y,x}$
Absent y helps :	$K_{x,\overline{y}}$	
Both :	$K_{x,x\overline{y}}$	

(x,y)	*Focal Point*
(0,0)	$(K_{x,\overline{y}}, K_y)$
(0,1)	(K_x, K_y)
(1,0)	$(K_{x,x\overline{y}}, K_y)$
(1,1)	$(K_{x,x}, K_y)$
(2,0)	$(K_{x,x\overline{y}}, K_{y,x})$
(2,1)	$(K_{x,x}, K_{y,x})$

Presence of an activator = Absence of an inhibitor = **A resource**

State Graphs

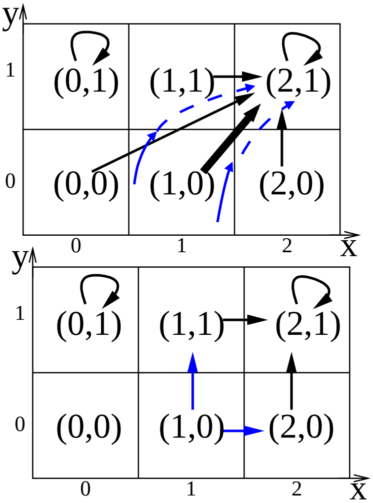
(x,y)	*Focal Point*
$(0,0)$	$(K_{x,\overline{y}}, K_y)=(2,1)$
$(0,1)$	$(K_x, K_y)=(0,1)$
$(1,0)$	$(K_{x,x\overline{y}}, K_y)=(2,1)$
$(1,1)$	$(K_{x,x}, K_y)=(2,1)$
$(2,0)$	$(K_{x,x\overline{y}}, K_{y,x})=(2,1)$
$(2,1)$	$(K_{x,x}, K_{y,x})=(2,1)$



State Graphs

(x,y)	*Focal Point*
$(0,0)$	$(K_{x,\overline{y}}, K_y)=(2,1)$
$(0,1)$	$(K_x, K_y)=(0,1)$
$(1,0)$	$(K_{x,x\overline{y}}, K_y)=(2,1)$
$(1,1)$	$(K_{x,x}, K_y)=(2,1)$
$(2,0)$	$(K_{x,x\overline{y}}, K_{y,x})=(2,1)$
$(2,1)$	$(K_{x,x}, K_{y,x})=(2,1)$

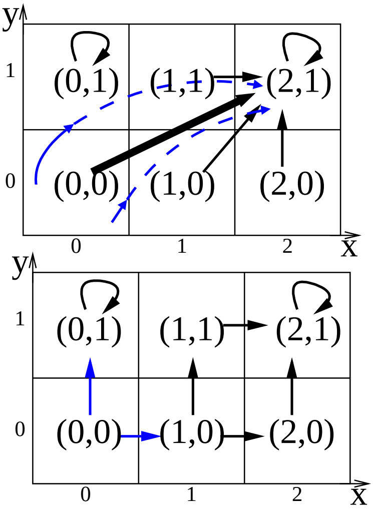
“desynchronization” $\longrightarrow$



State Graphs

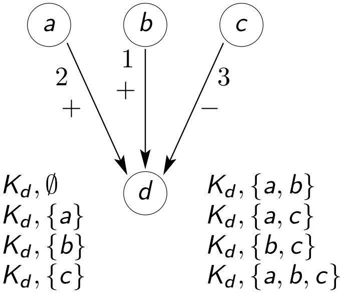
(x,y)	*Focal Point*
(0,0)	$(K_{x,\overline{y}}, K_y)=(2,1)$
(0,1)	$(K_x, K_y)=(0,1)$
(1,0)	$(K_{x,x\overline{y}}, K_y)=(2,1)$
(1,1)	$(K_{x,x}, K_y)=(2,1)$
(2,0)	$(K_{x,x\overline{y}}, K_{y,x})=(2,1)$
(2,1)	$(K_{x,x}, K_{y,x})=(2,1)$

“desynchronization” ⟶
by **units** of Manhattan distance



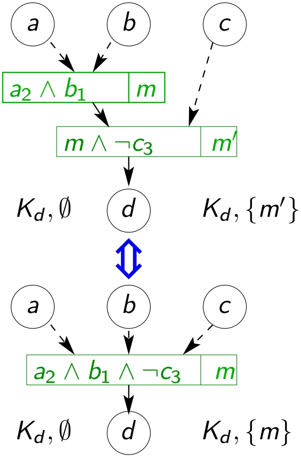
Multiplexes: encode cooperation knowledge

"Proteins of a and b form a complex before acting on d
and c inhibits d whatever a or b"



$K_d, \emptyset$
$K_d, \{a\}$
$K_d, \{b\}$
$K_d, \{c\}$

$K_d, \{a, b\}$
$K_d, \{a, c\}$
$K_d, \{b, c\}$
$K_d, \{a, b, c\}$



$K_d, \emptyset$ $K_d, \{m'\}$

$\Updownarrow$

$K_d, \emptyset$ $K_d, \{m\}$

8 ⟶ 2 parameters

The main problem

Exhaustively

identify the sets of parameters

that cope with known behaviours

from biological experiments

Solution = formal logic

- 2003: enumeration + CTL + model checking (Bernot,Comet,Pérès,Richard)
- 2005: path derivatives + model checking (Batt,De Jong)
- 2005: PROLOG with constraints (Trilling,Corblin,Fanchon)
- 2007: symbolic execution + LTL (Mateus,Le Gall,Comet)
- 2011: traces + enumeration + CTL + model checking (Siebert,Bockmayr)

(several other formal approaches define extensions of the theory)

Menu

1. Extended Thomas' discrete gene networks
2. Reminders on standard Hoare logic
3. Assertions and Trace specifications
4. Genetically modified Hoare logic
5. Examples

swap(x,y)

```
aux := x  ;
x   := y  ;
y   := aux
```

→ triple "$\{P\}program\{Q\}$"
precondition P, postcondition Q

swap(x,y)

$$\{(x = x_0) \wedge (y = y_0)\}$$

```
aux := x  ;
x   := y  ;
y   := aux
```

$$\{(y = x_0) \wedge (x = y_0)\}$$

$\rightarrow$ "P $\Longrightarrow$ (weakest precondition)" ?

swap(x,y)

$\{(x = x_0) \wedge (y = y_0)\}$

aux := x ;

x := y ;

$\longleftarrow \{(aux = x_0) \wedge (x = y_0)\}$

y := aux

$\{(y = x_0) \wedge (x = y_0)\}$

$\rightarrow$ backward proof strategy

$swap(x,y)$

$\{(x = x_0) \wedge (y = y_0)\}$

aux := x ;

$\longleftarrow \{(aux = x_0) \wedge (y = y_0)\}$

x := y ;

$\longleftarrow \{(aux = x_0) \wedge (x = y_0)\}$

y := aux

$\{(y = x_0) \wedge (x = y_0)\}$

swap(x,y)

$\{(x = x_0) \land (y = y_0)\}$

$\longleftarrow \{(x = x_0) \land (y = y_0)\}$

```
aux := x ;
```

$\longleftarrow \{(aux = x_0) \land (y = y_0)\}$

```
x   := y ;
```

$\longleftarrow \{(aux = x_0) \land (x = y_0)\}$

```
y   := aux
```

$\{(y = x_0) \land (x = y_0)\}$

swap(x,y)

$\{(x = x_0) \wedge (y = y_0)\}$

$\longleftarrow \{(x = x_0) \wedge (y = y_0)\}$

aux := x ;

$\longleftarrow \{(aux = x_0) \wedge (y = y_0)\}$

x := y ;

$\longleftarrow \{(aux = x_0) \wedge (x = y_0)\}$

y := aux

$\{(y = x_0) \wedge (x = y_0)\}$

$$\frac{}{\{Q[v \leftarrow expr]\} \quad v := expr \quad \{Q\}} \; :=$$

$$\frac{\{P\}p_1\{Q'\} \qquad \{Q'\}p_2\{Q\}}{\{P\}p_1\,;\,p_2\{Q\}} \; ;$$

swap(x,y)

$\{(x = x_0) \wedge (y = y_0)\}$

$\leftarrow \{(x = x_0) \wedge (y = y_0)\}$

aux := x ;

$\leftarrow \{(aux = x_0) \wedge (y = y_0)\}$

x := y ;

$\leftarrow \{(aux = x_0) \wedge (x = y_0)\}$

y := aux

$\{(y = x_0) \wedge (x = y_0)\}$

$$\frac{}{\{Q[v \leftarrow expr]\}\ \ v := expr\ \ \{Q\}}\ :=$$

$$\frac{\{P\}p_1\{Q'\} \qquad \{Q'\}p_2\{Q\}}{\{P\}p_1; p_2\{Q\}}\ ;$$

$$\dfrac{\dfrac{\dfrac{}{\{Q_3\}a_1\{Q_2\}}:= \qquad \dfrac{}{\{Q_2\}a_3\{Q_1\}}:=}{\{P\}a_1; a_2\{Q_1\}}\ ; \qquad \dfrac{}{\{Q_1\}a_3\{Q\}}:=}{\{P\}a_1; a_2; a_3\{Q\}}\ ;$$

abs(x)

$\{(x = x_0)\}$ $\longleftarrow$ $$\left\{ \begin{pmatrix} (x < 0) & \wedge \\ (-x \geq 0) & \wedge \\ ((-x)^2 = x^2) & \end{pmatrix} \vee \begin{pmatrix} (x \geq 0) & \wedge \\ (x \geq 0) & \wedge \\ (x^2 = x^2) & \end{pmatrix} \right\}$$

```
if (x<0) :
    r  := −x;
    r  := x
```

$\{(r \geq 0) \wedge (r^2 = x_0^2)\}$

$$\frac{\{Q_1\}p_1\{Q\} \qquad\qquad \{Q_2\}p_2\{Q\}}{\{(e \wedge Q_1) \vee (\neg e \wedge Q_2)\}\ \texttt{if}\ e\ \texttt{then}\ p_1\ \texttt{else}\ p_2\ \{Q\}}\ \textbf{if}$$

Also:

While loop: $$\frac{\{e \wedge I\}p\{I\} \qquad (\neg e \wedge I) \Longrightarrow Q}{\{I\}\mathit{while}\ e\ \mathit{with}\ I\ \mathit{do}\ p\{Q\}}$$

Empty program: $$\frac{P \Longrightarrow Q}{\{P\}\varepsilon\{Q\}}$$ use sparingly: loses *weakest* precondition!

Menu

1. Extended Thomas' discrete gene networks
2. Reminders on standard Hoare logic
3. Assertions and Trace specifications
4. Genetically modified Hoare logic
5. Examples

Assertions that formalize Thomas'framework

ω is the set of ressources of v:

$$\Phi_v^\omega \quad \equiv \quad \Big(\bigwedge_{m \,\in\, \omega} \varphi_m \Big) \wedge \Big(\bigwedge_{m \,\in\, G^{-1}(v) \setminus \omega} \neg \varphi_m \Big)$$

v can increase:

$$\Phi_v^+ \quad \equiv \quad \bigwedge_{\omega \subset G^{-1}(v)} (\Phi_v^\omega \Longrightarrow K_{v,\omega} > v)$$

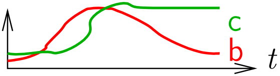
v can decrease:

$$\Phi_v^- \quad \equiv \quad \bigwedge_{\omega \subset G^{-1}(v)} (\Phi_v^\omega \Longrightarrow K_{v,\omega} < v)$$

Trace specifications

- $x+ \mid x- \mid x := n \mid assert(\varphi)$
- $p_1 ; p_2 ; \cdots ; p_n$
- *if* φ *then* p_1 *else* p_2
- *while* φ *with* ψ *do* p
- $\forall(p_1, p_2, \cdots, p_n)$
- $\exists(p_1, p_2, \cdots, p_n)$

Examples:



- $b+; c+; b-$
- $\exists(b+, b-, c+, c-, \varepsilon)$
- *while* $(b < 2)$ *with* $(c > 0)$
  do $\exists(b+, b-, \forall((c-; a-), c+))$ *od*;
  $b-$

Menu

1. Extended Thomas' discrete gene networks
2. Reminders on standard Hoare logic
3. Assertions and Trace specifications
4. Genetically modified Hoare logic
5. Examples

Additional inference rules

Incrementation rule: $$\frac{}{\{\ \Phi_v^+\ \wedge\ Q[v\leftarrow v+1]\ \}\ v+\ \{Q\}}$$

Decrementation rule: $$\frac{}{\{\ \Phi_v^-\ \wedge\ Q[v\leftarrow v-1]\ \}\ v-\ \{Q\}}$$

Assertion rule: $$\frac{}{\{\ \varphi\wedge Q\ \}\ \mathit{assert}(\varphi)\ \{Q\}}$$

Universal quantifier rule: $$\frac{\{P_1\}p_1\{Q\} \qquad \{P_2\}p_2\{Q\}}{\{P_1\wedge P_2\}\ \forall(p_1,p_2)\ \{Q\}}$$

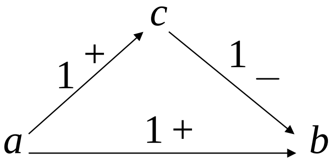
Existential quantifier rule: $$\frac{\{P_1\}p_1\{Q\} \qquad \{P_2\}p_2\{Q\}}{\{P_1\vee P_2\}\ \exists(p_1,p_2)\ \{Q\}}$$

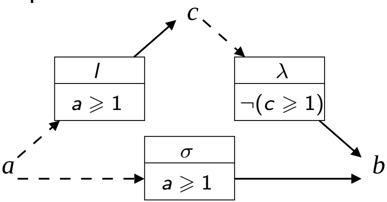
Menu

1. Extended Thomas' discrete gene networks
2. Reminders on standard Hoare logic
3. Assertions and Trace specifications
4. Genetically modified Hoare logic
5. Examples

Example: Feedforward “loop”

Uri Alon most frequent gene network patterns





Behaviour of *b* after switching *a* from off to on ?

Simple off→on→off behaviour of *b* with the help of *c*:

$$\{(a = 1 \ \wedge \ b = 0 \ \wedge \ c = 0)\} \ b+ \ ; \ c+ \ ; \ b- \ \ \{b = 0\}$$

possible if and only if: $K_{b,\{\sigma,\lambda\}} = 1 \ \wedge \ K_{c,\{l\}} = 1 \ \wedge \ K_{b,\{\sigma\}} = 0$

Feedforward example (continued)

Although $b+; c+; b-$ is possible, if c becomes "on" before b, then b will never be able to get "on"

Proof by refutation:

$$\left\{ \begin{array}{l} a = 1 \ \wedge \ b = 0 \ \wedge \ c = 1 \ \wedge \\ K_{b,\sigma\lambda} = 1 \ \wedge \ K_{c,l} = 1 \ \wedge \ K_{b,\sigma} = 0 \end{array} \right\}$$

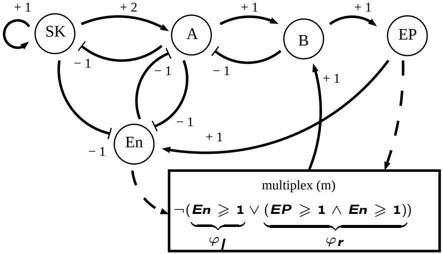
$$\mathit{while}\ b < 1\ \mathit{with}\ I\ \mathit{do}\ \exists(b+, b-, c+, c-, \varepsilon)$$

$$\{\ b = 1\ \}$$

the triple is inconsistent, whatever the loop invariant I !

Cell cycle in mammals

- A 22 gene model reduced to 5 variables using multiplexes



multiplex (m)

$\neg(\underbrace{\boldsymbol{En} \geqslant \mathbf{1}}_{\varphi_l} \vee \underbrace{(\boldsymbol{EP} \geqslant \mathbf{1} \wedge \boldsymbol{En} \geqslant \mathbf{1})}_{\varphi_r})$

(J. Behaegel)

SK = Cyclin E/Cdk2, Cyclin H/Cdk7
A = Cyclin A/Cdk1
B = Cyclin B/Cdk1
En = APC^{G1}, CKI (p21, p27), Wee1

EP = APC^{M}, Phosphatases

- 48 states, 26 parameters, 339 738 624 possible valuations, 12 trace specifications and a few temporal properties

Cell cycle in mammals (continued)

- 13 parameters have been entirely identified (50%) and only 8192 valuations remain possible according to the generated constraints (0.002%)
- Additional reachability constraints (e.g. endoreplication and quiescent phase) have been necessary, on an extended *hybrid* extension of the Thomas' framework, to identify (almost) all parameters
- This initial Hoare logic identification step was crucial: it gave us the sign of the derivatives in all the (reachable) states

Concluding Comments

Pros:

- simple and elegant, very efficient
- easy expression of sequential biological observations, copes well with biological measurements
- dynamic gene Knock Out within the same model
- sound, complete and decidable

Cons:

- no implicit "holes" in the successive biological observations
- thresholds are empirical
- simplification of assertions/constraint solving can be difficult

WP-SMBioNet: proof of feasability using Choco (*Z. Khalis*)

Hybrid extension: . . . coming soon (*J. Behaegel &al.*)