

# Modular Verification of Interactive Systems with an Application to Biology

Peter Drábik

Dipartimento di Informatica, Università di Pisa, Italy

Amsterdam – CS2Bio'10 – June 10, 2010

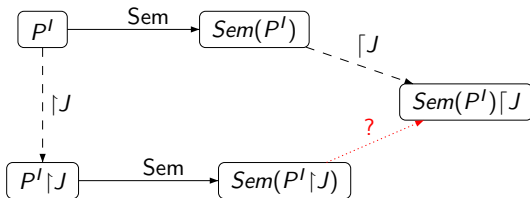
# Introduction

- Analysis techniques from CS in Biology
- **Model checking** – important verification tool
- Exploring **all** possible **behaviours** of the system
- **State explosion** problem
- **Modularity** – a natural method for trying to avoid this issue

# Motivation

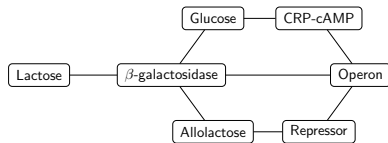
- Goal – verify **properties of subsystems**, and infer that these hold in the complete system
- Class of properties identified by Grumberg *et al.* as **ACTL** – the universal fragment of CTL
- Addressed by Attie for verification and synthesis of concurrent programs
  - Synchronisation skeletons – move of a component may depend on the states of other components
  - Not suitable for describing biological systems
- **Synchronised moves** of more components are crucial to model biological phenomena
- Extension – **sync-programs** – enable synchronised move of an arbitrary number of automata

## Modular verification – principle



- Define: prog. language, semantics, projections
- Show:  $Sem(P^I \upharpoonright J) \sqsupseteq Sem(P^I) \lceil J$
- Computation is **preserved** (infinite path)
- Properties talking about **all computations** (ACTL)
- **Positive answer** carried over to the whole system

## Interaction graph



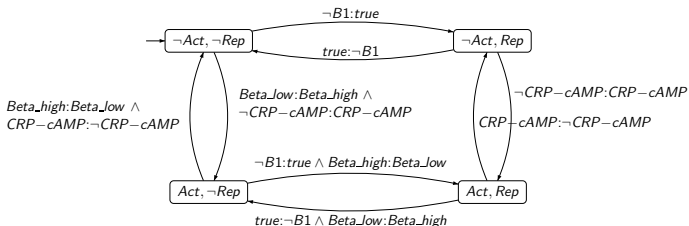
System is made of components.

### Definition

**Interaction graph**  $I$  – components that interact directly

**Atomic propositions**  $AP_i$  – pairwise disjoint

# Sync-automaton



## Definition

### Sync-automaton $P_i^!$

- states – mappings of  $AP_i$  to  $\{true, false\}$
- moves –  $s_i \xrightarrow{c_i} t_i$

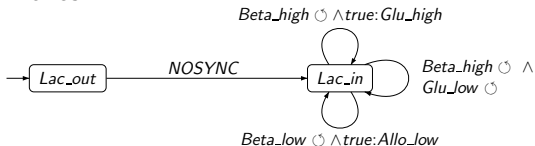
**Synchronisation condition**  $c_i$  is of the form  $\bigwedge_{j \in L} A_j: B_j$ .

# Synchronisation conditions

Expression  $\bigwedge_{j \in L} A_j : B_j$

Definition allows

- loops
- multiple moves
- *NOSYNC* moves



# Sync-program

Parallel composition of sync-automata related by an interaction graph

## Definition

The **sync-program** is a tuple

$$P^I = (S_0^I, P_1^I || \dots || P_n^I),$$

where each  $P_i^I$  is a **sync-automaton**. Set  $S_0^I = S_1^0 \times \dots \times S_n^0$  is the set of **initial states** of the sync-program.

# Semantics – Definition

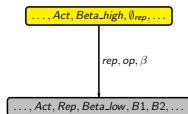
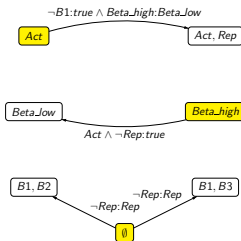
## Definition

Semantics of  $P^I = (S_I^0, P_1^I || \dots || P_n^I)$  is a labelled transition system on  $I$ -states.

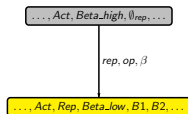
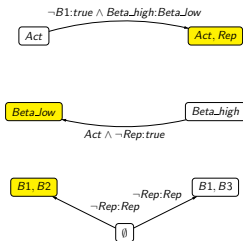
There is a transition  $(s, \ell, t)$  iff

- label  $\ell$  contains **indices** of all automata that **perform a move**, with mutually **satisfied synchronisation conditions**
- $\ell$  is minimal

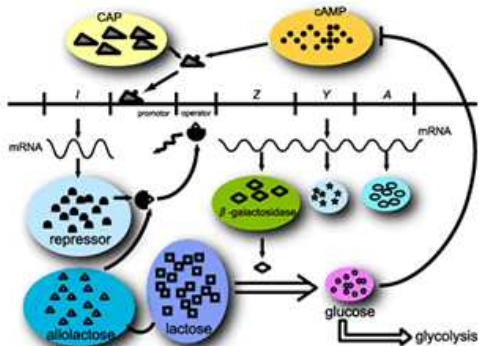
# Semantics – Example



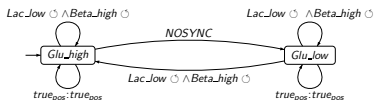
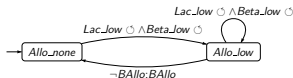
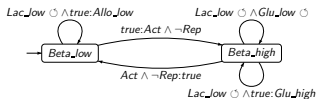
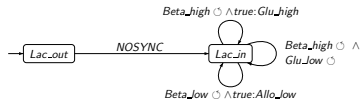
# Semantics – Example



## Lac operon regulation



# The model (1)



## The model (2)

Peter Drábik

Introduction

Sync-programs

Syntax

Semantics

Example

Modular  
verification

Projections

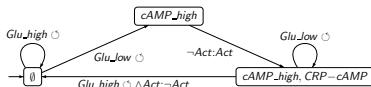
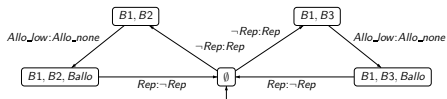
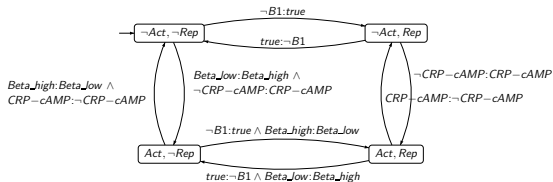
Lemmas

Logic

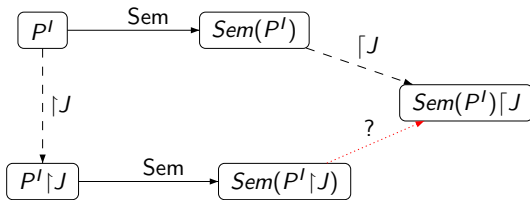
Theorem

Experiments

Conclusions



# Modular verification – principle

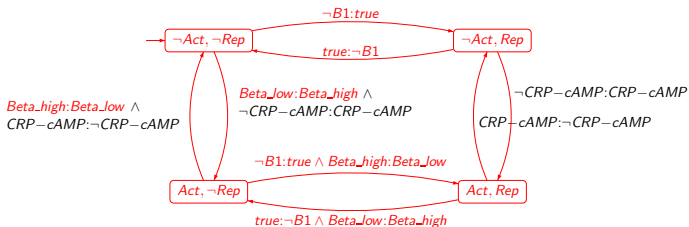


- Define: prog. language, semantics, **projections**
- Show:  $Sem(P^I \upharpoonright J) \sqsubseteq Sem(P^I) \upharpoonright J$

# Syntactical projection

Syntactical projection – subprogram  $P^I \upharpoonright J$

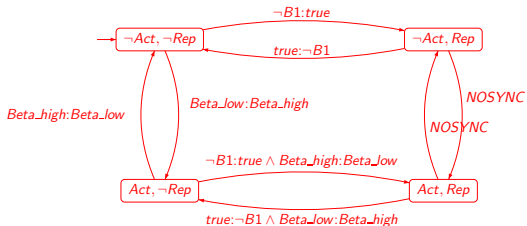
- only sync-automata from  $J$
- sync-automata from  $J$  remain, synchronisation conditions change



# Syntactical projection

Syntactical projection – subprogram  $P^I \upharpoonright J$

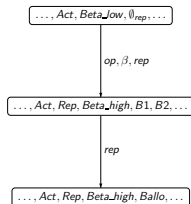
- only sync-automata from  $J$
- sync-automata from  $J$  remain, synchronisation conditions change



# Semantical projection

Syntactical projection –  $\mathcal{M}^I \upharpoonright J$

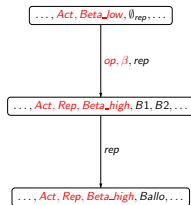
- $I$ -states projected
- transitions projected



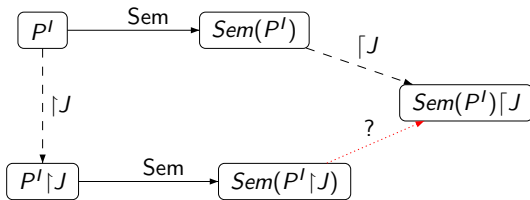
# Semantical projection

Syntactical projection –  $\mathcal{M}^I \upharpoonright J$

- $I$ -states projected
- transitions projected



# Modular verification – principle



- Define: prog. language, semantics, projections
- Show:  $Sem(P^I \upharpoonright J) \sqsubseteq Sem(P^I) \upharpoonright J$

# Principle (1)

- Verification of the properties of the **computation**
- Computation = **maximal path** (fullpath)

## Lemma (Path projection)

*Let  $\mathcal{M}_I$  be semantics of sync-program  $P^I$ . For every  $J \subseteq I$  if  $\pi$  is a path in  $\mathcal{M}_I$  then  $\pi \upharpoonright J$  is a path in  $\mathcal{M}_J$ , where  $\mathcal{M}_J$  is the semantics of sync-program  $P^J = P^I \upharpoonright J$ .*

## Principle (2)

Possible **problem** – by projecting we **lose path maximality**

### Definition

A path  $\pi = (s^1, l^1, s^2, l^2, \dots)$  in  $\mathcal{M}_I$  is **fair** iff for all  $i \in |I|$  we have that  $\{m \mid i \in l^m\}$  is infinite.

### Lemma (Fullpath projection)

Let  $J \subseteq I$  be an interaction graph. If  $\pi$  is a **fair fullpath** in  $\mathcal{M}_I$ , then  $\pi \upharpoonright J$  is a **fair fullpath** in  $\mathcal{M}_J$ .

# ACTL logic

## Definition (ACTL logic)

- *true*, *false*
- $p$ ,  $\neg p$  for  $p \in AP$
- $f \wedge g$ ,  $f \vee g$
- $AX_j f$ ,  $A[fUg]$  and  $A[fU_w g]$

## Features

- Includes:  $AFf$ ,  $AGf$  and  $AG[p \rightarrow AFq]$ .
- $ACTL_J$  – atomic propositions are from  $\{AP_i \mid i \in J\}$
- Can express: exclusion, necessary consequence, necessary persistence, oscillatory behaviour
- Semantics – on LTSs, needs fullpaths

# Property preservation theorem

## Theorem (Property preservation)

*Let  $J \subseteq I$  be an interaction graph,  $s$  an  $I$ -state and  $f$  an  $ACTL_J$  property. If  $\mathcal{M}_J, s \models f$  then  $\mathcal{M}_I, s \models f$ .*

- proved by induction on the formula

## Experiments

*“The increase of allolactose concentration can only be mediated by  $\beta$ -galactosidase in low concentration”.*

- formula  $AG(Allo\_none \wedge Beta\_high \rightarrow A(\neg Allo\_low \cup Beta\_low))$
- true in the semantics of  $P^{allo,\beta}$ .

*“The operon will necessarily oscillate between repressed and unrepressed state”.*

- formula  $AG((rep \rightarrow AF\neg rep) \wedge (\neg rep \rightarrow AFrep))$
- true in  $\mathcal{M}_I$ , but the verification in semantics of  $P^{op}$  fails
- by inspecting the model, we enlarge the fragment needed
- true in the semantics of  $P^{op,rep}$

# Conclusions

## Discussion

- Generality of the approach
  - Systems with finite number of states
  - Fairness
  - Unstructured states

## Further work

- Dynamic systems
- Abstract interpretation – full logic preservation
- Relation to other formalisms ( $\kappa$ -calculus,...)
- Probabilistic extension...

# Conclusions

Thank you.

## References



Attie, P.C.

*Synthesis of large dynamic concurrent programs from dynamic specifications..*

CoRR abs/0801.1687 (2008)



Pinto, M.C., Foss, L., Mombach, J.M., Ribeiro, L.

*Modelling, property verification and behavioural equivalence of lactose operon regulation..*

Computers in Biology and Medicine 37(2) (2007) 134–148