

Modular Verification of Biological Systems

Peter Drábik

Supervisors: Andrea Maggiolo-Schettini and Paolo Milazzo

Dipartimento di Informatica, Università di Pisa, Italy

Ph.D. thesis defence – December 22, 2011

Biological systems

- **Systems Biology**: understanding the principles of functioning of biological systems via modelling and analysis
- Biological systems (cells in particular) are **complex systems of interactive components**
 - main actors: DNA, RNA, proteins, membranes
 - main processes: gene regulation networks, metabolic pathways, signalling pathways
- The dynamics of cellular processes is usually **analysed** by means of
 - Ordinary Differential Equations (ODEs)
 - stochastic simulation
- These approaches can give either an **average system behaviour** or **number of possible system behaviours**
- This is often enough to validate hypotheses on the system functioning or suggest refinements

Model checking of biological systems

- Simulation techniques do not cover all possible system behaviours
- On the contrary, techniques such as **model checking** permit **all the possible behaviours** of a system to be explored
- Model checking:
 - usually requires the system to be modelled as a sort of **finite state automaton**
 - requires behavioural properties of interest to be expressed as **temporal logic formulae**
 - performs an **exhaustive exploration** of a model of the system behaviour to assess the truth of the properties
- Attempts to apply model checking to biological systems are in the literature

Modular verification

- Unfortunately, the complexity of realistic systems makes the analysis by model checking often unfeasible
 - **state explosion problem**
- A method for trying to avoid the state space explosion problem is to exploit system **modularity**
 - properties of interest often deal with a small number of system components
 - unnecessary components could be **abstracted** away from the model in order to make model checking tractable
- Goal – verify **properties of subsystems**, and infer that these hold in the complete system
- **Modular verification** – addressed for verification of concurrent programs

Aim of the thesis

- Develop a **modular verification technique for biological systems**
 - Apply techniques from computer science
- **Evaluate** how well it performs in the biological setting
- Individuate **class of systems** for which it can be applied
- **Implement** the technique in form of a **tool**
- Investigate **implications for computer science**

Peter Drábik

Introduction

Sync-programs

Modular
verification

Theory
Practice
Example

Generalisation

Modular
verification of
pathways

Dynamic
systems

Conclusions

- Define a **formalism** suitable for biological systems, neither too simple nor too complex
- Show how to **apply the modular technique** to this formalism
- **Evaluate** on a small example, implementation by exploiting an existing **tool**
- **Generalise** the class of formalisms we could apply the technique to
- Realistic **case study**, a particular class of systems
- Biologically motivated **extension**: dynamic systems
- Conclusions

Sync-programs

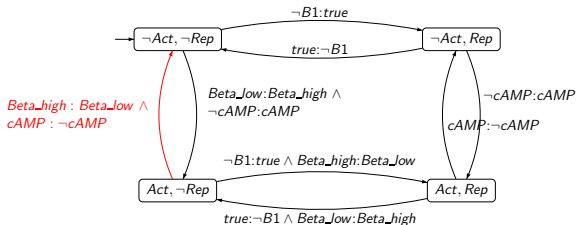
Yet another formalism

- **synchronising** finite-state **automata**
- not too complex
- expressive enough: able to describe a qualitative abstraction of a **reasonable class of biological systems**

Features

- a parallel composition of automata
- similar to sync-skeleton programs, for which there is a modular verification technique
- equipped with synchronisation, necessary to model biological systems

Sync-automaton



Sync-automaton $P_i^!$

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

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

- state-based synchronisation conditions
- multi-way synchronisation

Semantics of synchronisation

Peter Drábik

Introduction

Sync-programs

Modular
verification

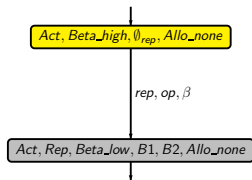
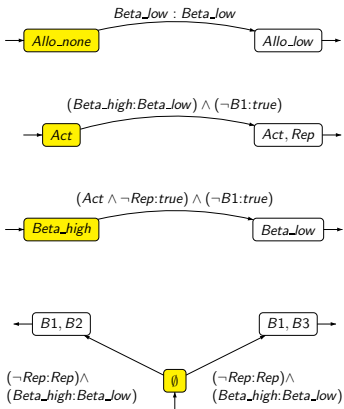
Theory
Practice
Example

Generalisation

Modular
verification of
pathways

Dynamic
systems

Conclusions



Semantics of synchronisation

Peter Drábik

Introduction

Sync-programs

Modular
verification

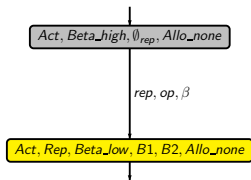
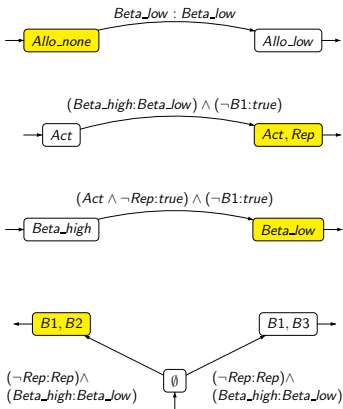
Theory
Practice
Example

Generalisation

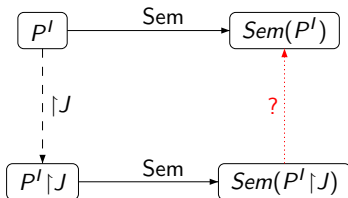
Modular
verification of
pathways

Dynamic
systems

Conclusions

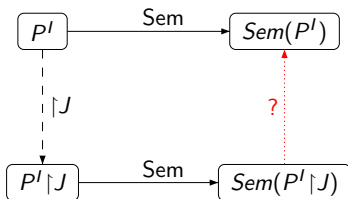


Property preservation



- Modular verification
 - properties of programs **inferred** from the properties of subprograms
- Property preservation
 - **Projection** – abstracting from unnecessary components
 - If a property holds in a model fragment obtained by projection, then it holds also in the whole model

Principle



- Sync-program **projection**
 - **removes** some **automata** and **references** to them in the conditions of the remaining automata
- Over-approximation implies property preservation
 - need to prove that a **computation** is **preserved by projection**
 - computation is a **fair path**

The role of fairness

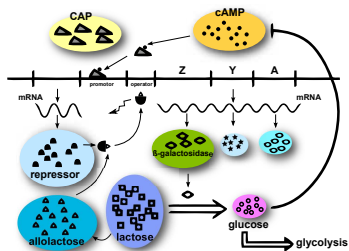
- Fairness
 - Additional constraint on paths in the LTS
 - Requires that **each component participates infinitely often** on a “fair” computation of the program
- Modular verification level
 - guarantees that a **projection of a computation is a valid computation of the part**
 - sufficient, not a necessary condition. Could be weakened.
- Modelling biological systems level
 - a computation has to **correspond to** an intuitively **correct** system **behaviour**
 - first to consider explicitly
 - meaningful in this setting
 - a stronger reaction-level fairness desirable

- Properties
 - **Universally quantified**, talking about all computations
 - Logic $ACTL^-$: $p \mid \neg p \mid f \wedge g \mid A[f U g] \mid A[f U_w g]$
- **Property preservation theorem**:
 - If $\mathcal{M}_J, s \models_{\Phi} f$ then $\mathcal{M}_I, s \models f$
- Interpretation
 - **Positive answer** carried over to the whole system
 - False negatives possible due to over-approximation

Modular verification – from theory to practice

- Translation of sync-programs into the input language of NuSMV
 - **NuSMV** is a well-established and efficient model checker
- The tool can be used to verify:
 - **ACTL** properties in a **modular** way
 - **CTL** properties (which includes ACTL) **monolithically**
- Challenge: change of concurrency paradigm, **no synchronisation** in NuSMV modules
 - introduce a protocol for synchronisation
 - need to introduce intermediate states
 - need to change the properties
- Formally proved the correctness of the translation

Case study: the *lac* operon regulation



Properties

We considered the following properties (all of which hold):

- (P1) “The allolactose bound to the repressor implies that the operon is repressed”
- (P2) “The increase of allolactose concentration can only be mediated by β -galactosidase in low concentration”
- (P3) “While lactose is inside the cell, the operon will necessarily oscillate between a repressed and an unrepressed state”
- (P4) “When glucose concentration drops and lactose is inside the cell, the lac operon will eventually be fully expressed”
- (P5) “When lactose concentration is low, the lac operon will eventually be unrepressed”
- (P6) “When glucose concentration is low, the lac operon will eventually be activated”

Example: property (P2)

“The increase of allolactose concentration can only be mediated by β -galactosidase in low concentration”.

Example: property (P2)

“The increase of allolactose concentration can only be mediated by β -galactosidase in low concentration”.

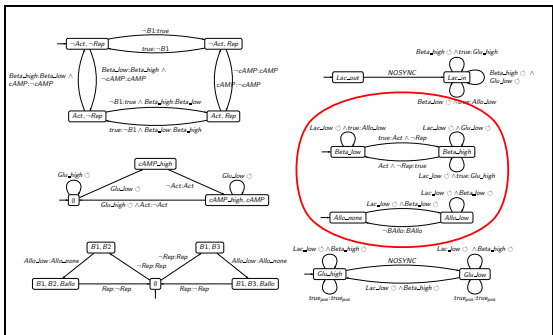
- formula

$AG(Allo_none \wedge Beta_high \rightarrow A(\neg Allo_low \cup Beta_low))$

Example: property (P2)

“The increase of allolactose concentration can only be mediated by β -galactosidase in low concentration”.

- formula
 $AG(Allo_none \wedge Beta_high \rightarrow A(\neg Allo_low \cup Beta_low))$
- can be **verified on the part** consisting only of automata representing components **Beta** and **Allo**



Example: property (P3)

“While lactose is inside the cell, the operon will necessarily oscillate between a repressed and an unrepressed state”.

Example: property (P3)

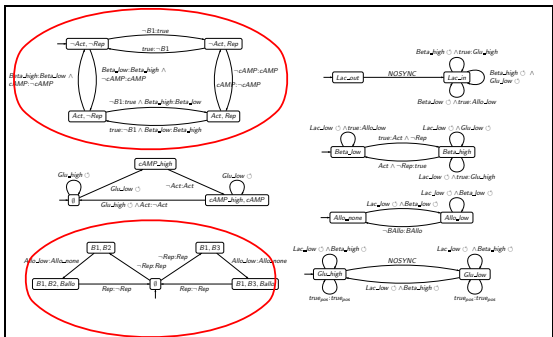
“While lactose is inside the cell, the operon will necessarily oscillate between a repressed and an unrepressed state”.

- formula $AG((rep \rightarrow AF \neg rep) \wedge (\neg rep \rightarrow AF rep))$

Example: property (P3)

“While lactose is inside the cell, the operon will necessarily oscillate between a repressed and an unrepressed state”.

- formula $AG((rep \rightarrow AF\neg rep) \wedge (\neg rep \rightarrow AF rep))$
- false in the part consisting of Op
- true in the part consisting of Op and Repr



		Monolithic		Modular	
Prop.	#Comp.	Time	BDD size	Time	BDD size
(P1)	2/7	0.4s	174750 n.	<0.1s	3271 n.
(P2)	2/7	0.4s	180278 n.	<0.1s	1991 n.
(P3)	4/7	1.2s	326908 n.	0.1s	35269 n.
(P4)	7/7	0.9s	283614 n.	0.9s	283614 n.
(P5)	4/7	0.7s	256112 n.	0.1s	29193 n.
(P6)	4/7	0.6s	222511 n.	0.1s	21442 n.

- [CS2Bio 2010, SACS 2011]

Generalisation

- Analyse the **class of formalisms** we could apply the modular verification technique to
- Generalise the **synchronisation schema**, each automata move explicitly indicates all moves of other automata without which it is not willing to be performed
- Synchronisation seen as a feature of semantics, it permits simultaneous execution of moves that synchronise
- Modular verification requires a **specific type** of synchronisation – conditions identified

Applicability to other formalisms

- Semantics has the **complete synchronisation** schema
 - ① moves from distinct automata
 - ② for each move its condition is satisfied
 - ③ each move is requesting synchronisation with all the participating moves
- To avoid synchronisation of two components through a third party that could be removed by projection operation
- Generalise the principles to other formalisms
- **CSP** even stronger condition execute whenever an action is in the alphabet
- CCS complementary channels, always in pairs, not applicable. More sophisticated projection needed.
- [CS&P 2011, subm. FI 2011]

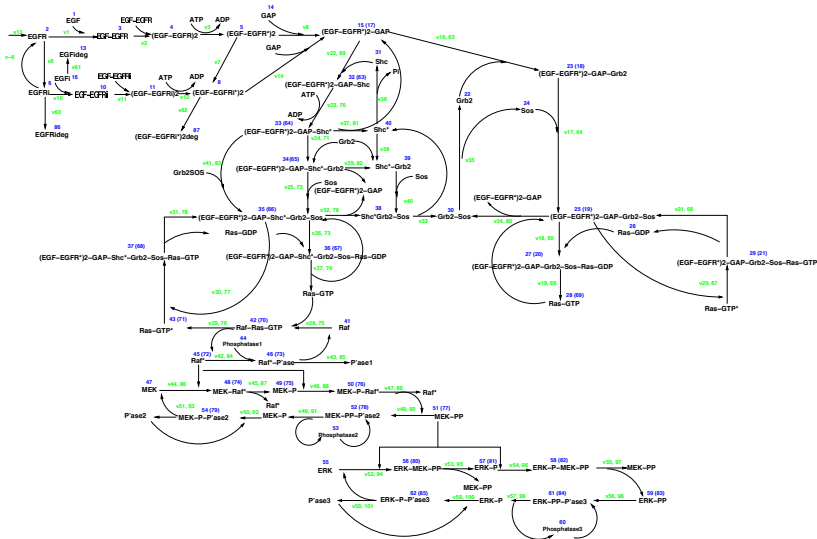
Pathways

- Purpose
 - proof of concept for the **scalability** of the approach
 - show how to verify a particular **class** of biological systems
- Pathway
 - **series of biochemical reactions** occurring within a cell
 - modelled as a set of reactions
- Syntactic assumptions
 - reaction $r_1, \dots, r_n \rightarrow p_1, \dots, p_n \{c_1, \dots, c_m\}$
 - species: reactants, products and catalysts
 - as many reactants as products, positional correspondence
 - species = part of the state, reaction = synchronous change
- Semantic assumptions / abstractions
 - species are present or absent
 - uncatalyzed reaction does not consume reactants
 - catalyzed reaction consumes all reactants, all catalysts need to be present

Pathway decomposition and implementation

- Sync-automata **identification**
 - $r_1, r_2 \rightarrow p_1, p_2\{c\}$
 - three constituents that synchronise
- Specification of automata
 - **states** based on species present
 - **moves** by reactions: with/without catalysers
 - add **self loops** due to fairness
- Translation **directly to NuSMV**
 - Standard translation possible
 - Ad-hoc translation – more efficient for this class

Case study: MAPK pathway



Case study: MAPK pathway

- MAP kinase cascade activated by surface and internalised EGF receptors
- 143 species and 80 reactions.
- Automatically 14 automata are identified

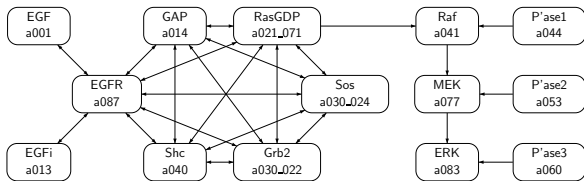


Figure: Interaction graph

Experiments

The final product of the pathway (ERK-PP) is always reachable.

- $AF(ERK-PP \vee ERK-PPi)$ (E1)

Raf* is a necessary predecessor of MEK-PP.

- $\neg E(\neg Raf^* U ERK-PP)$ (E2)

The precursor Raf* is always reachable.

- $AF(Raf^* \vee Raf^*i)$ (E3)

The precursor Raf* is never produced.

- $AG\neg(Raf^* \vee Raf^*i)$ (E4)

Prop.	Absent automata	Modular verification			Monolithic ver.	
		Projection	Result	Time	Result	Time
(E3)	-	9/14	true	492.7s	true	1129.0s
(E3)	EGFR	9/14	false	45.6s	false	84.7s
(E4)	EGFR	9/14	true	51.5s	true	81.5s
(E3)	GAP	9/14	false	48.2s	false	85.6s
(E4)	GAP	9/14	true	43.4s	true	82.7s
(E3)	Grb2	9/14	false	50.6s	false	86.4s
(E4)	Grb2	9/14	true	51.7s	true	82.9s
(E3)	RasGDP	9/14	false	51.7s	false	176.6s
(E4)	RasGDP	9/14	true	51.8s	true	161.2s
(E3)	Shc	9/14	true	51.9s	true	164.9s
(E3)	Sos	9/14	false	51.6s	false	90.1s
(E4)	Sos	9/14	true	49.4s	true	81.7s

Table: Comparison of verification times

- [In preparation]

Peter Drábik

Introduction

Sync-programs

Modular
verification

Theory
Practice
Example

Generalisation

Modular
verification of
pathways

Dynamic
systems

Conclusions

- **Automatic** tool for **analysis of pathways**
- Causality, reachability analysis
- For **biologists**, hides the model checking
- For a certain class of analyses more **efficient**
- [In preparation]

Dynamic sync-programs

- Biologically-motivated **dynamic extension** of sync-programs
 - new copies of system components can be **created dynamically**
 - more **instances** of the same component allowed
- Generative reactions can be dealt with (e.g. $gene_A \rightarrow gene_A + prot_A$)
- Automata transitions can generate new automata (from a given finite set of types of automata)
- Some moves labelled by **creation conditions**

Dynamic vs. static

- Semantics based on multisets
- Logic **DACTL⁻**, dealing with instances
- Need to adjust the modular verification
- Used it to model (a fragment of) the EGF pathway
- Translation of dynamic sync-programs into **Petri nets**
- **Unboundedness** problem – inherent to infinite state systems
 - but Petri net approaches are possible to be applied
- [NCMA 2010]

Contributions

- An automata-based **formalism** for qualitative modelling of biological systems, such as metabolic pathways, signalling pathways and gene regulatory networks
- A **modular verification technique** for this formalism, allows properties expressed in the ACTL⁻ to be verified on fragments of models
- We have shown that the **class of the properties** preserved includes several useful and **biologically relevant** properties
- Encode the formalism into synchronisation skeletons, prove correctness. Thus a **tool** can be used for verification.
- Investigated the possibility of **application to other existing formalisms**
- Evaluated the method on a **realistic example** of a metabolic pathway. Automatically identify the component of a pathway. Modular verification of pathways.
- **Extension** of the formalism by allowing **dynamic** creation of components

Discussion and response to reviewers

- Unconditional **fairness**
 - reasonable for applications in Systems Biology
 - even though it is not generally possible to extend unfair paths into fair ones, in some cases we can do it semi-automatically in an empirical way
 - strong fairness for needed for pathways
- ACTL properties – **biological relevance**
- **Scalability**: example with 143 species and 80 reactions, shows significant difference in the time and space necessary for verification
- Decomposition is not automatic. However we can do it automatically in certain cases (pathways)

Further work and open problems

- Dynamic sync-programs
 - more experiments and implementation
 - relation to Petri nets, subclass induced
- Pathways
 - reaction-level fairness
 - modelling inhibitors
- Automatic assume-guarantee reasoning
- Quantitative extension

References I



Drábik, P., A. Maggiolo-Schettini and P. Milazzo.
*Modular Verification of Interactive Systems with an
Application to Biology.*

Electronic Notes in Theoretical Computer Science, 268 :
61-75, 2010.



Drábik, P., A. Maggiolo-Schettini and P. Milazzo.
*Modular Verification of Interactive Systems with an
Application to Biology.*

Scientific Annals of Computer Science XXI, pp. 39-72,
2011.

References II



Drábik, P., A. Maggiolo-Schettini and P. Milazzo.
Dynamic Sync-programs for Modular Verification of Biological Systems.

In 2nd Int. Workshop on Non-Classical Models of Automata and applications (NCMA10), Jena, Germany, 2010.



Drábik, P., G. Scatena.
An Application of Model Checking to Epidemiology (Extended Abstract).

In 1st Int. Workshop on Applications of Membrane computing, Concurrency and Agent-based modelling in POPulation biology (AMCA-POP 2010), Jena, Germany, 2010.

References III



Drábik, P., A. Maggiolo-Schettini and P. Milazzo.
*Application of a modular verification technique in Systems
Biology (abstract).*

In Proceedings of BITS11, Pisa, Italy, 2011.



Drábik, P., A. Maggiolo-Schettini and P. Milazzo.
*On Conditions for Modular Verification in Systems of
Synchronising Components.*

In Proceedings of Proceedings of CS&P 2011, Pultusk,
Poland, 2011.