Static Analysis for Systems Biology

Flemming Nielson, Hanne Riis Nielson
Technical University of Denmark

Corrado Priami, Debora Schuch da Rosa
University of Trento

Abstract. This paper shows how static analysis techniques can help understanding biological systems. Based on a simple example we illustrate the outcome of performing three different analyses extracting information of increasing precision. We conclude by reporting on the potential impact and exploitation of these techniques in systems biology.

1 Introduction

Systems biology is a new approach to study biological phenomena. The main idea is to lift the reductionist paradigm that drives the current research practice in biology to a global view of biological systems that can be investigated at different level of abstractions. The new paradigm is hypothesis driven, iterative and global. The main focus is no more on the structure of biological components (as it was in the human genome project for instance), but on the functions that these components have in the dynamic evolution of biological systems. More abstractly we can say that biologists are passing from production of knowledge to organization of knowledge so that structure (well reported in many public data bases) must be related to behavior (still largely not organized in suitable representations).

From computer science we know that passing from representations of structures to representations of the corresponding dynamic evolution means that the complexity grows exponentially. Since the amount of data available on the structure is huge, we can easily infer that we need smart techniques to investigate the dynamic behavior of biological systems. A good candidate is the use of static analysis techniques (note that here static means that the description of a system – the structure – is analyzed to discover dynamic properties – the behavior or evolution) that often are associated with polynomial time algorithms. This gain in efficiency is paid in terms of precision of the analysis that we can carry out: in fact, the study of dynamic properties is carried out on approximations of the possible evolution of systems.

In this paper we investigate control flow analysis, a promising static techniques, that is surveyed in the next section. We then report the potential impact and the exploitation of the technique in systems biology.

2 Static analysis

Static analysis techniques [2] have their origin in the field of compiler optimisation; today the application area also include validation of safety and security properties of programs and systems. The hallmark of static analysis techniques is their ability to statically extract complex information about the dynamic behaviour of programs – note that this does *not* involve executing the programs but only systematic inspection of the program text. A static analysis is designed for a particular property of interest and

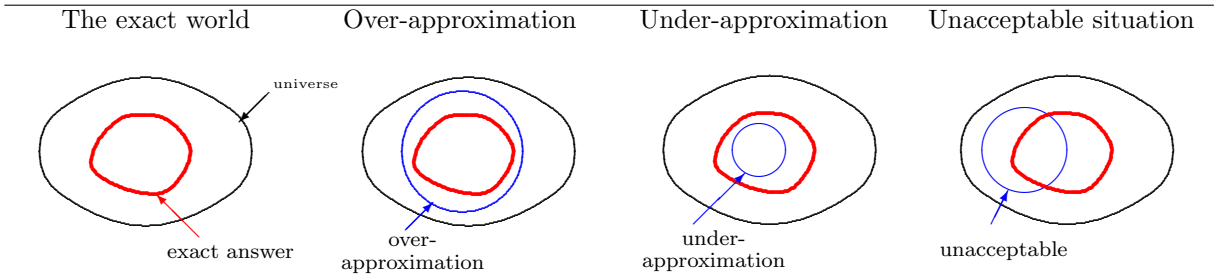

Figure 1: The nature of approximation.

for a particular specification language and it can be applied to all programs of that language. The information extracted for a program is guaranteed to be a correct description of the behaviour of the program. Unfortunately, for most interesting properties it is impossible to obtain *exact* information so typically static analysis techniques are approximative.

The notion of approximation is crucial. As illustrated on Figure 1 we may be interested in over-approximations as well as under-approximations. When we have an over-approximation to the exact behaviour of a program we can guarantee that certain events will *never* happen – namely those not included in the analysis result. When we have an under-approximation we can guarantee that certain events *will indeed* happen – namely those included in the analysis result. Results that neither are over- nor under-approximations are uninteresting as we cannot interpret the results meaningfully. Obviously, it is trivial to construct uninformative over- and under-approximations so the challenge is to obtain approximations striking the right balance between the precision and computational costs – the more precision we require the more costly will the analysis be.

A biological perspective. Although static analysis is developed in a completely different setting we believe that it will be an excellent tool for analysing biological systems. To illustrate the potentials let us consider a simple program written as a BioAmbient process [6, 7]:

$$\begin{array}{ll}(c)(cell_1)(cell_2)(cell_3) & [\ \mathsf{rec}\, X.\ (\mathsf{enter}\ cell_1.\ X + \mathsf{exit}\ cell_2.\ X + c\hat{}?\{x\}.\ \mathsf{expel}\ x.\ X) \\ & \quad \mid [\mathsf{exit}\ cell_3.\ 0\,]^{D}\ ]^{mol} \\ & \mid [\mathsf{rec}\, X.\ (\mathsf{accept}\ cell_1.\ X + \mathsf{expel}\ cell_2.\ X + c\_!\{cell_3\}.\ X)\,]^{cell}\end{array}$$

The process consists of three ambients *cell*, *mol* and D where D is inside *mol*; *cell* and *mol* are at the top-level. The behaviour of the process is governed by so called capabilities that make use of the four names c, $cell_1$, $cell_2$ and $cell_3$. The ambients *cell* and *mol* contain processes that will repeat themselves whereas the process of D only will be executed once. We write "+" for the choice between various actions and "." for sequencing of actions.

The behaviour of the above process is as follows (recall that when a capability is consumed it is no more available until the recursive process is re-activated in the flow of control by reaching the control variable of the recursive definition):

1. First *mol* is ready to execute the enter $cell_1$ capability and *cell* is ready to execute the accept $cell_1$ capability; as a result *mol* will move into *cell*. The ambient D is not influenced by this; it is still inside *mol*.
2. After this there are two possibilities:
   (a) *mol* is ready to execute exit $cell_2$ and *cell* is ready to execute expel $cell_2$; as a result *mol* may leave *cell* and we are back in the initial configuration. Thus *mol* is ready to enter *cell* again and in fact it may move in and out of *cell* any number of times.

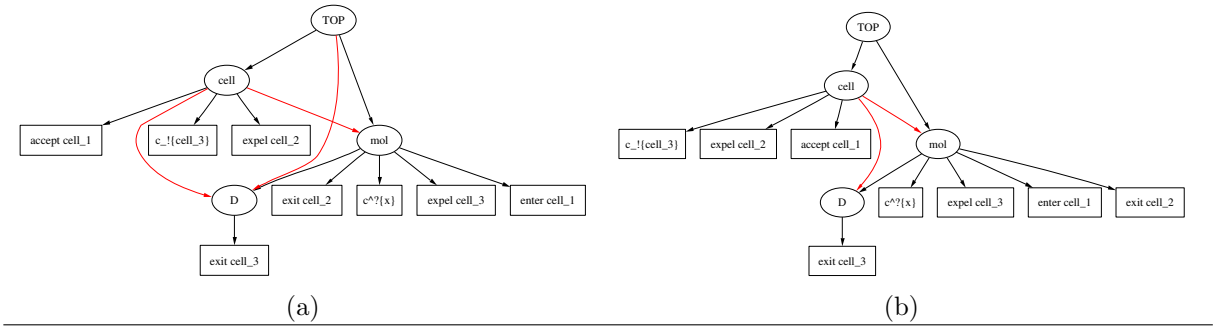

Figure 2: (a) Simple control flow analysis. (b) Context dependent analysis.

(b) *cell* is ready to send the name $cell_3$ on c and *mol* is ready to receive a name on c and bind it to x. Once that has happened the capability expel x of *mol* will really be expel $cell_3$ and since D is ready to execute the corresponding exit $cell_3$ we see that D will leave *mol* and become a subambient of *cell*. At this stage *mol* can leave *cell* using the exit $cell_2$ and expel $cell_2$ capabilities and subsequently it may reenter and leave any number of times.

Static analysis can be used to discover a good part of this behaviour without actually running the process on a simulator. We shall now illustrate this for three different static analyses that we are developing for BioAmbients. They all compute over-approximations to the exact behaviour but they differ in their precision (and computational cost) and thereby the kind of insights they give.

A simple control flow analysis. Our first analysis is described in detail in [1]. It computes an over-approximation to the *father-son relationship* between ambients and capabilities and in doing so it ignores the order in which the capabilities are executed and also it does not take the context into account.

The analysis takes its starting point in the initial father-son relationship as illustrated by the black edges on Figure 2 (a); the oval nodes represent ambients, the square nodes represent capabilities and an edge from one node to another means that the source node is the father of the target node. The analysis mimicks the semantics of the BioAmbients and include more and more information about father-son relationships until no further information can be deduced – note that this process will always terminate. In order to give a proper treatment of the communication capabilities, the analysis will also keep track of the potential name bindings (this part of the analysis result is not illustrated on the figure).

For our example the red edges on Figure 2 (a) represent the information added in this stage of the analysis. The (red) edge from *cell* to *mol* is added because the analysis correctly discovers that *mol* moves into *cell* (and hence *cell* becomes the father of *mol*) and similarly the edge from *cell* to D is added because D moves into *cell*. The edge from *TOP* to D really should not have been added but this cannot be discovered by the analysis. The edge is added for the following reason: the analysis observes that D can be inside *cell* and both *mol* and *cell* can be inside *TOP*; furthermore D contains the exit $cell_3$ capability and *cell* contains the expel $cell_3$ capability (since x can be bound to $cell_3$). These observations will cause the analysis to conclude that D can move to the top-level. The "mistake" made by the analysis is that it does not discover that these capabilities never will be executed when D has moved into *cell*.

It is important to stress that the analysis result is indeed correct: it is an over-approximation to the precise father-son relationship. And it does provide useful information about our example – in particular, we learn that *cell* will neither move into *mol* nor into D.

Context dependent analysis. A more precise analysis can be obtained by taking context into account [3]. To illustrate this let us consider a variant of the analysis where it is required that the grandfather and great-grandfather information is consistent in order for the father-son relationship to be updated.

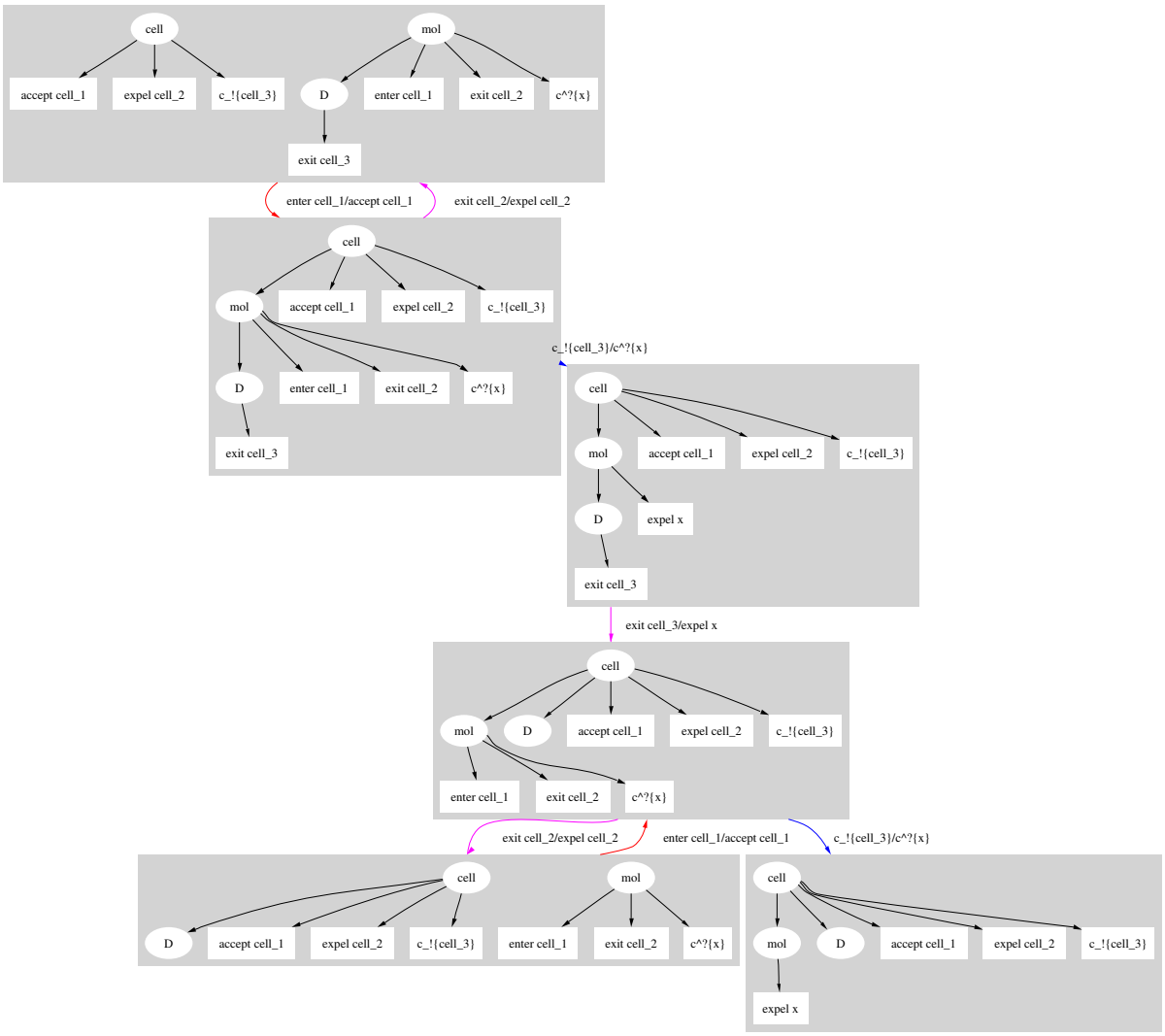

Figure 3: Pathway analysis.

Also this analysis will be approximative: informally speaking the semantics requires that *all* ancestors of the two components involved in an action are consistent whereas the analysis discussed here only insist that the two most recent ancestors are. In general the computational requirements of the analysis grow exponentially with the number of ancestors taken into account.

As mentioned earlier the analyses also keep track of the name bindings and also here we shall take context information into account. The simple control flow analysis above discovered that x could be bound to $cell_3$; the context dependent analysis will additionally discover that this information is indeed only valid inside *mol*. So in particular it is *not* valid inside *cell* and this is the reason why the analysis no longer makes the (false) conclusion that D may move to the top-level. This is illustrated on Figure 2 (b).

One may observe that the context dependent analysis gives a precise account of the father-son relationship of the example; for more complex processes this need not to be the case. However, it is also clear that the analysis result does not give a precise picture of *how* the execution of the system takes place; a much more powerful analysis is needed for that.

Pathway analysis. In work in progress [4] we are developing a static analysis that gives a fairly detailed but still approximative picture of the executing of a process. The central observation is that at a certain stage only some of the capabilities can be executed and only some of the ambients can be reached from the top-level. In the pathway analysis, a configuration will include information about the father-son relationship between *exposed* ambients and capabilities.

We can now mimick the execution of a matching pair of exposed capabilities and create a new configuration with an updated version of the father-son relationship; this is much as in the simple control flow analysis but with the extra twist that now we also add information about ambients and capabilities that become exposed as a result of executing the selected capabilities. And we introduce an extra component to the analysis recording that a transition was made from the first configuration to the second while executing the two capabilities.

The result of analysing our example program using this technique is shown in Figure 3. The initial configuration is shown at the top and as expected we can observe that the enter/accept and exit/expel capabilities will cause the system to toggle between two configurations, one where *mol* is at the top-level and one where it is inside *cell*. Once the communication over c has happened there is no way back and the next step will be the movement of D from *mol* to *cell*. As we already pointed out when explaining the semantics it is possible for *mol* to move in and out of *cell* any number of times as reflected by the cyclic structure at the bottom of the graph. However the analysis also reveals that the system may enter a stuck configuration if it decides to perform the communication once more – at this stage *mol* does not contain the ambient D so there is no way for the execution to proceed.

3 Perspectives

Molecular biology has accumulated data for many years on components of living systems by exploiting their structure and their position (a brilliant example is the human genome project), but very little information is available on the functions that such components have. A challenge of the next years is to implement a paradigm shift from structure to functions/behaviour yielding functional genomics.

The new paradigm, however, needs new computer science tools as a support in addition to the ones developed so far in the so-called bioinformatics field that are mainly related to storage of data and query tools. Recently, some interest in computer science, especially from concurrency and programming language theory, has emerged and some researchers are active in trying to adapt process calculi with their theory to a biological applicative domain [5].

The main techniques that are investigated are based essentially on simulation of behaviour and sometimes on equivalence checking based on bisimulation. These techniques, however, suffer the large size of complex systems and hence the computational power needed to have usable results in systems biology. To overcome this problem we suggest the use of static analysis that, with its origin in compiler technology, originally was developed to handle large programs.

The properties that we can investigate for systems can be an order of magnitude larger than the ones that we can handle with dynamic tools, and they are concerned for instance with the localization components within structure, their possible interactions or migration (e.g., translocation to the nucleus and subsequent transcription), extraction of positive or negative feedback loops from large pathway networks.

The side effect of relying on the computer science tools to study biological systems is that it would force researchers to adopt a hypothesis-driven, iterative and integrated approach to biology. Indeed a computer scientist first needs to build a model of the software or hardware that he/she wants to study relying on a set of integrated automatic facilities. Then a prototype is built out of the model and then tuned to the expected behavior by iterating the sequence of steps "change the model" and "build a new prototype". In this scenario the prototype for biologists would be experimentation. Most of the tools mentioned above where built by computer scientists to reduce as much as possible the number of iterations needed to reach

a satisfiable product.

To make the approach feasible in systems biology it is necessary to hide as much mathematical details as possible from biologists so that they can continue using their usual tools getting precise analysis for free. But this step requires an automatic translation of the biological specification of systems into mathematical formalisms and therefore biological specifications must be non ambiguous and quite standard. Notable work is done in this direction by the group of Roux-Rouquiè at the Institut Pasteur by relying on a standard computer science modeling language (UML) to model biological systems [8]. Even if this language could be still too mathematical for biologists it is easy to imagine it as an intermediate level into which classical data base (e.g., KEGG) representations are mapped and possibly completed with lacking information by the user on a by need basis.

To implement the hiding of details we suggest to follow the approach introduced and developed by the EU project DEGAS in the FET global computing proactive initiative (`http://www.omnys.it/degas`). The main idea is splitting the architecture of our environment into two parts: an interface environment and a kernel environment. The interface allows the user to specify the model and to select the kind of analysis s/he wants to carry out. The kernel contains the machinery to perform formal static analysis. The interface is built upon UML so that the profile for systems biology developed by Roux-Roquie and her group can be easily incorporated in the overall architecture.

Finally, this interdisciplinar approach is useful for both sciences. Once the complete behavior of cells is well understood, we can think of programming biological entities. This should be an enormous breakthrough because it would solve the problem of limited resources while performing computations and so new problems could be automatically solved on the computer science side. On the systems biology side this should definitely lead to predictive and preventive medicine.

Acknowledgement. This research has been funded in part by the DEGAS project (number IST-2001-32072) funded by the European Union and by the LoST project (number 21-02-0507) funded by the Danish Natural Science Research Council.

References

[1] F. Nielson, H. Riis Nielson, C.Priami, and D. Schuch da Rosa. Control Flow Analysis for BioAmbients. 2003.

[2] F. Nielson, H. Riis Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 1999.

[3] F. Nielson, H. Riis Nielson, H. Pilegaard, C.Priami, and D. Schuch da Rosa. Context Dependent Analysis of BioAmbients. Work in progress, 2003.

[4] H. Riis Nielson and F. Nielson et al. Pathway analysis. Work in progress; 2003.

[5] C. Priami, editor. *Proceedings of Computational Methods in Systems Biology, LNCS 2062*. Springer, 2003.

[6] A. Regev. *Computational system biology: A calculus for biomolecular knowledge*. PhD thesis, Tel Aviv University, 2003.

[7] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: An abstraction for biological compartments. 2003. Manuscript available from http://www.luca.demon.co.uk/.

[8] M. Roux-Rouquie and J.L. Le Moigne. The systemic paradigm and its relevance to modeling biological functions. *C.R. Biologies*, 325:419–430, 2002.