Type disciplines for systems biology

Livio Bioglio

INSERM, UMR-S 1136, iPLESP, Paris, France

Systems Biology is a discipline that aims to study complex biological systems by means of computational models. Because of the complexity of biological behaviors, the formalisms used in this field are usually designed ad-hoc for the biological topic of interest, or they need to be tuned by a long set of evolution rules. Here we present a different approach: we define biological properties through a type discipline, leaving the formalisms as general as possible. We explore three different kinds of Type Systems: a static one, that limits the model that can be written by modelers; a dynamic one, that limits the evolution of the model at run-time; and an hybrid combination of the previous ones.

1 Static type system

Homogeneous biological entities are classified according to their behavior. In order to reproduce such classification, we propose a Minimal Object-Oriented Core Calculus for term-rewriting formalisms. A rewrite system is composed by a term, representing the structure of the modeled system, and a series of reduction rules, representing the possible evolutions of the system: depending on the formalism, these rules can be embedded in terms, like in P Systems [7], or defined in a separate part, like in the Calculus of Looping Sequences [1] (CLS for short). The objective of the OO Core Calculus is to facilitate the organizations of rules, also improving their re-use, and to check the correctness of the model at compile time. In our core calculus it is possible to define classes, that contain methods (encapsulation) and extend another class (subtyping), inheriting all its methods (inheritance). Methods are formed by a sequence of variables, the arguments, and a sequence of reduction rules containing these variables. They are called on symbols of the model, representing biological entities, with a sequence of values as arguments (method invocation). A method invocation is replaced by the reduction rules of the method, in which the variables are replaced by the values used as arguments. These reduction rules are then used for the evolution of the model. The syntax, definitions and rules of the OO Core Calculus are inspired by the ones proposed by Igarashi, Pierce and Wadler for Featherweight Java [6], a minimal core calculus for modeling the Java Type System. For more details, see [2].

2 Dynamic type system

In Biology and Chemistry there can be found several examples of repellency, such as hydrophobicity (the physical property of a molecule that is repelled from a

mass of water), the behavior of anions and cations, or, at a different level of abstraction, the behavior of the rh antigen for the different blood types. As a counterpart, there may be elements, in nature, which always require the presence of other elements: for example, it is difficult to find a lonely atom of oxygen, they always appear in the pair O_2. We bring these aspects at their maximum limit, and, by abstracting away all the phenomena which give rise/arise to/from repellency (and its counterpart), we assume that for each kind of element of our reality we are able to fix a set of elements which are required by the element for its existence, and a set of elements whose presence is forbidden by the element. We enrich the basic CLS with a type discipline which guarantees the soundness of reduction rules with respect to some relevant properties of biological systems deriving from the required and excluded kinds of elements. The key technical tool we use is to associate to each reduction rule the minimal set of conditions an instantiation must satisfy in order to assure that applying this rule to a "correct" system we get a "correct" system as well. We also propose a type inference algorithm, based on the machinery of principal typing [8], and show its soundness and completeness. The required/excluded elements properties modeled here assure, through type inference, that only compatible elements are combined together in the different environments of the biological system took in consideration. Thus the type system intrinsically yields a notion of correct (well-behaving) system according to the expressed requirements. The detailed Type Discipline can be found in [5].
There are cases in which the request/repellency model cannot reflect the behavior of a biological system. An example is homeostasis, the property of a system that regulates its internal environment and tends to maintain stable conditions that are optimal for survival: when this equilibrium is disturbed, built-in regulatory devices respond in order to restore the balance. Different living organisms employ homeostatic mechanisms to maintain some conditions in specific ranges: the human body, like in all the warm-blooded animals, maintain a near-constant body temperature using mechanisms such as vasodilation and vasoconstriction; microorganisms maintain the iron presence above a minimum level to maintain life but up to a maximum level to avoid iron toxicity. For this reason, we propose an extension of the previous Type Disciplines, where we assume that for each element of our system we can fix the minimum and the maximum number of other elements it requires. We enrich CLS with a type discipline and typed reductions that guarantee the soundness of reduction rules with respect to the properties of biological systems deriving from the minimum and the maximum requested numbers of elements, and a type inference algorithm for inferring the type of rewriting rules. Our contribution appeared in [3].

3 Hybrid type system

We present the variant of the Calculus of Looping Sequences with global and local rewrite rules (CLSLR, for short). Global rules are the usual rules of CLS, and they can be applied anywhere in a given term wherever their patterns match

the portion of the system under investigation, while local rules can only be applied in the compartment in which they are defined. Terms written in CLSLR are thus syntactically extended to contain explicit local rules within the term, on different compartments. Local rules can be created, moved between different compartments and deleted. Having a calculus in which we can model the dynamic evolution of the rules describing the system allows to study emerging properties of complex systems in a more natural and direct way. As it happens in nature, where data and programs are encoded in the same kind of molecular structures, we insert rewrite rules within the terms modeling the system under investigation. On the other hand, some rule may represent general behaviors, common to the whole system: global rules are used for avoiding the repetition of such rules in each compartment. Since in this framework the focus is put on local rules, we define a set of features that can be associated to each one. Features may define general properties of rewrite rules or properties which are strictly related to the model under investigation. We define a membrane type for the compartments of our model and develop a type systems enforcing the property that a compartment must contain only local rules with specific features. Our framework has been presented in [4].

References

1. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo and A. Troina. A Calculus of Looping Sequences for Modelling Microbiological Systems. *Fundamenta Informaticæ*, 72(1–3):21–35, 2006.
2. L. Bioglio. A Minimal OO Calculus for Modelling Biological Systems. *Computational Models for Cell Processes (CompMod) 2011*, EPTCS 67:50–64, 2011.
3. L. Bioglio. Enumerated Type Semantics for the Calculus of Looping Sequences. *RAIRO - Theoretical Informatics and Applications*, 45(01):35–58, 2011.
4. L. Bioglio, M. Dezani-Ciancaglini, P. Giannini and A. Troina. A Calculus of Looping Sequences with Local Rules. *7th Workshop on Developments in Computational Models (DCM'11)*, EPTCS 88:43–58, 2011.
5. L. Bioglio, M. Dezani-Ciancaglini, P. Giannini and A. Troina. Type Directed Semantics for the Calculus of Looping Sequences. *International Journal of Software and Informatics*, to appear.
6. A. Igarashi, B. Pierce and P. Wadler. Featherweight Java: a Minimal Core Calculus for Java and GJ. *ACM Transactions on Programming Languages and System*, 23:396–450, 2001.
7. G. Păun. *Membrane Computing. An Introduction.* Springer, 2002.
8. J. B. Wells. The Essence of Principal Typings. *International Colloquium on Automata, Languages and Programming (ICALP'02)*, LNCS 2380:913–925, 2002.