CALCULI FOR CONCURRENT OBJECTS

Cosimo Laneve

Dipartimento di Scienze dell'Informazione, Bologna
laneve@cs.unibo.it

Concurrent object-oriented languages, such as POOL [1], ABCL [12], Obliq [3] or Java [7], usually structure the program into *objects*, model the parallelism with *threads* and the mutual exclusion with *locks* (or some variant of them). These attempts to integrate concurrency and object-oriented features are somehow naive and no rigorous motivation for the design choices is given. As a result, the semantics is often crisp and one stumbles into anomalies (*cfr.* inheritance anomaly [8]) or typing flaws [10].

On the other hand the concepts of processes and objects seem strongly related. The formers are entities that interact with the environment, possibly with a dynamically changing interface (*mobile processes*). The latters are units with a set of operations that may affect an internal state. Relating process interfaces to operations and interactions to operation invocations is quite natural as much as relaxing the sequential constraint of standard object calculi. It is therefore not surprising that foundational studies for bringing objects and processes together have recently arisen a broad interest.

The theory of concurrent objects has been developed according to the tradition of functional and concurrent calculi. Several researchers have in fact defined foundational languages for such paradigm, equipping them with precise and clear semantics, simple type systems and equational theories. To date two approaches have emerged. One consists on recasting some well-established object-calculus and enriching it with concurrent primitives (parallel composition and the operation of scope restriction) [9, 2, 6]. The second attempts the other way around, trying to extend process calculi with objects by introducing record types [11, 4].

There is an apparent distance among these calculi and concurrent object languages which jeopardizes their utility. The focus in the latters is on classes and inheritance of class definitions while the formers, being object-based calculi, definitely overlook these issues. Regrettable, forgetting about classes eludes any treatment of inheritance, which is a crucial notion in object-oriented programming and even more when concurrency is added. Some progress in this direction is done in [5], where a class based calculus

P. Ciancarini et al. (eds.), *Formal Methods for Open Object-Based Distributed Systems*

has been developed out of a well known process calculus – the *join-calculus* – and a formal analysis of inheritance and possible anomalies is undertaken.

References

[1] Pierre America. Issues in the design of a parallel object-oriented language. *Formal Aspects of Computing*, 1(4):366–411, 1989.

[2] Paolo Di Blasio and Kathleen Fisher. A calculus for concurrent objects. In Ugo Montanari and Vladimiro Sassone, editors, *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR '96)*, LNCS 1119, pages 406–421, 1996.

[3] Luca Cardelli. Obliq A language with distributed scope. SRC Research Report 122, Digital Equipment, June 1994.

[4] Silvano Dal-Zilio. Quiet and bouncing objects: Two migration abstractions in a simple distributed blue calculus. In Hans Hüttel and Uwe Nestmann, editors, *Proceedings of the Worshop on Semantics of Objects as Proceedings (SOAP '98)*, pages 35–42, June 1998.

[5] Cédric Fournet, Cosimo Laneve, Luc Maranget and Didier Rémy. Inheritance in the join-calculus. ftp://ftp.cs.unibo.it/pub/laneve/obj.ps.gz, October 1998.

[6] Andrew D. Gordon and Paul D. Hankin. A concurrent object calculus: reduction and typing. In Uwe Nestmann and Benjamin C. Pierce, editors, *HLCL '98: High-Level Concurrent Languages*, volume 16(3) of *entcs*, Nice, France, September 1998. To appear.

[7] James Gosling, Bill Joy, and Guy Steele. *The Java Language Specification.* Addison Wesley, 1996.

[8] Satoshi Matsuoka and Akinori Yonezawa. Analysis of inheritance anomaly in object-oriented concurrent programming languages. In Gul Agha, Peter Wegner, and Akinori Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming*, chapter 4, pages 107–150. The MIT Press, 1993.

[9] Oscar Nierstrasz. Towards an object calculus. In O. Nierstrasz M. Tokoro and P. Wegner, editors, *Proceedings of the ECOOP'91 Workshop on Object-Based Concurrent Computing*, LNCS 612, pages 1–20, 1992.

[10] Vijay Saraswat. Java is not Type Safe. Technical report, AT&T research, 1997. http://www.att.research.com/ vi.

[11] Vasco T. Vasconcelos. Typed concurrent objects. *Lecture Notes in Computer Science*, 821:100–117, 1994.

[12] Akinori Yonezawa, Jean-Pierre Briot, and Etsuya Shibayama. Object-oriented concurrent programming in ABCL/1. *ACM SIGPLAN Notices*, 21(11):258–268, November 1986. Proceedings of OOPSLA '86.