Spatial Types for Processes
(abstract of talk)

Luís Caires
Departamento de Informática
FCT/UNL, Portugal

António Ravara
Departamento de Matemática
IST/UTL, Portugal

February 13, 2004

1 Introduction

Several type-systems for concurrent systems have been proposed recently, building upon a general notion of behavioral typing [6, 9, 10]. Behavioral types aim at capturing more refined properties of processes, namely safety or liveness properties of processes like race-freedom and deadlock-freedom, rather than simpler properties like absence of communication errors, that can be handled by more traditional sorting systems. However, there is a growing interest in using types to ensure non-behavioral properties of processes, for instance, properties related to resource usage and spatial distribution of capabilities.

The aim of our work is to develop a general notion of type based on spatial logic for concurrent processes, expressive enough to capture properties of the kind already ensured by behavioral typing systems, but also able to establish more "intensional" spatial / structural properties. In fact, many interesting properties of distributed systems are inherently spatial, for instance connectivity, stating that there is always an access route between two different sites, unique handling, stating that there is at most one server process listening on a given channel name, or resource availability, stating that a bound exists on the number of channels that can be allocated at a given location.

Spatial logics support the specification not only of behavioral properties but also of structural properties of concurrent distributed systems, in a fairly integrated way. Essentially, spatial logics are modal logics that can talk about the internal structure of each world. The interpretation of each world as a structured space, and moreover as a space seen as a certain kind of resource [8], distinguishes spatial logics among modal logics. Spatial logics have been used in the definition of several core languages, calculi, and data models [5, 7, 2, 4]. In particular, in [4], spatial types where already used to type programs in a functional language for semistructured data. Moreover, proof systems for verifying [3] and model-checking [1] spatial properties of concurrent processes have been already defined. The challenge we want to consider now is to design a type system (and thus a decidable proof system) able to assign an interesting class of spatial / behavioral types to an interesting class of processes.

Types of our type system include spatial types such as $A \mid B$ (*cf.*, the composition operator of spatial logic), capturing spatial decomposition, and behavioral types such as $\{a_i.A_i\}$, characterizing the potential of interaction of a process. An example of a derivable typing judgment in our type system is

$$\Gamma \vdash (a\langle\rangle \mid (\textbf{rec}\ \mathcal{X}.a().(a\langle\rangle \mid a\langle\rangle))) : \Box(\overline{0} \mid \overline{0})$$

The type $\Box(\overline{0} \mid \overline{0})$ says that the process will always present at least two separate threads. Much of the deductive power of our type system is encapsulated in a subtyping relation $A :> B$, which is interpreted as logical entailment (since types are interpreted as formulas of spatial logic). This is not unexpected, but contrasts with the subtyping relations introduced in behavioral type systems [6, 9], where types are understood as abstract processes, equipped with a form of abstract operational semantics, and subtyping corresponds to a form of simulation.

References

[1] L. Caires. Behavioral and Spatial Properties in a Logic for the Pi-Calculus. In I. Walukiwicz, editor, *Proc. of Foundations of Software Science and Computation Structures'2004*, Lecture Notes in Computer Science. Springer Verlag, 2003.

[2] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). *Information and Computation*, 186(2):194–235, 2003.

[3] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part II). *Theoretical Computer Science*, to appear.

[4] L. Cardelli, P. Gardner, and G. Ghelli. Manipulating Trees with Hidden Labels. In A. D. Gordon, editor, *Proc. of Foundations of Software Science and Computation Structures (FoSSaCS '03)*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[5] L. Cardelli and A. D. Gordon. Anytime, Anywhere. Modal Logics for Mobile Ambients. In *27th ACM Symp. on Principles of Programming Languages*, pages 365–377. ACM, 2000.

[6] A. Igarashi and N. Kobayashi. A Generic Type System for the Pi-Calculus. *Theoretical Computer Science*, 1–3(311):121–163, 2004.

[7] S. Ishtiaq and P. O'Hearn. BI as an Assertion Language for Mutable Data Structures. In *28th ACM Symp. on Principles of Programming Languages*, 2001.

[8] P. O'Hearn and D. Pym. The Logic of Bunched Implications. *The Bulletin of Symbolic Logic*, 5(2):215–243, 1999.

[9] S. K. Rajamani and J. Rehof. A behavioral module system for the pi-calculus. In *SAS'01: 8th International Static Analysis Symposium (Paris, France)*, LNCS. Springer, 2001.

[10] A. Ravara and V. Vasconcelos. Typing non-uniform concurrent objects. In C. Palamidessi, editor, *CONCUR'00*, volume 1877 of *Lecture Notes in Computer Science*, pages 474–488. Springer-Verlag, 2000. Extended version published as a Department of Mathematics, Instituto Superior Técnico Research Report.