Virtually Timed Ambients: Formalisation and Analysis

Johanna B. Stumpf, Einar Broch Johnsen, and Martin Steffen

University of Oslo, Oslo, Norway
{johanbst, einarj, msteffen}@ifi.uio.no

1 Motivation

The ambient calculus is the process algebra of locations and domains, originally developed by Cardelli and Gordon [2] for distributed systems such as the Internet. We extend the ambient calculus with a notion of virtual time as a resource. The resulting calculus can be used for instance to model aspects of virtualization in cloud computing, where different locations, barriers between locations, and barrier crossing are important features, as well as elasticity which allows to provision virtual resources on-demand.

2 Previous work on timed process algebras

Algebraic concurrency theories such as ACP, CCS and CSP have been extended to deal with time-dependent behaviour in various ways (e.g., [1, 7, 4]). All these approaches describe speed as the absolute *duration* of processes, while in our approach speed describes the relative *processing power* of an ambient.

3 Preliminaries on mobile ambients

An ambient represents the location or domain where a process is running. Ambients can be nested, such that a surrounding *parental ambient* contains *subambients*, and the nesting structure can change dynamically. This is specified by three basic capabilities. The input capability $\mathit{in}\ n$ indicates the willingness of a process, respectivly its containing ambient, to enter an ambient named n, running in parallel outside, e.g., $k[\mathit{in}\ n.P] \mid n[Q] \to n[k[P] \mid Q]$. The output capability $\mathit{out}\ n$ enables an ambient to leave its surrounding ambient n, e.g., $n[k[\mathit{out}\ n.P] \mid Q] \to k[P] \mid n[Q]$. The third basic capability $\mathit{open}\ n$ allows to open an ambient named n which is on the same level as the capability, e.g., $k[\mathit{open}\ n.P \mid n[Q]] \to k[P \mid Q]$. This syntax, as well as the semantics we consider, is based on [6] and largely unchanged compared to [2].

4 Virtually timed mobile ambients

We extend mobile ambients with notions of virtual time and resource consumption. Virtual time is a resource, which is made available to a location by its parental location, similar to time slices that an operating system provisions to its processes. Interpreting the locations of ambients as a place of deployment, each timed ambient is modelled to have a certain computing power, determined by its deployment. Thus, our model of timed ambients uses a *local* notion of time, which, however, is *relative* to the computing power of the embedding, parental ambients. The basic model of virtually timed ambients is described in [5].

Timed systems A timed ambient contains one *local clock* and possibly other timed ambients or classic untimed ambients and processes. A *computing environment* is a timed ambient which contains *resources*, as explained below.

Local clocks To represent the outlined time model, each timed ambient is equipped with one local clock responsible for triggering timed behaviour and local resource consumption. Clocks have a *speed*, interpreted *relative* to the speed of the surrounding timed ambient. The speed s of a clock is given by the tuple (p, q), where p is the number of local time slices emitted for a number q of time slices received from the surrounding ambient. Time slices propagate from parental clocks to clocks in the subambients. Thus, the time in a nested ambient is relative to the global time, depending on the speeds of the clocks of the ambients it is nested in. We assume one universal outermost ambient with a *global clock* triggering the clocks of the local subambients recursively. When moving timed ambients, we must update the clocks to guarantee a correct propagation of time slices. As virtual time is made available to an ambient by its surroundings we have to ensure that a clock distributes time slices to all of its current subambients. Thus, we use an *update function* and define timed capabilities **in** n, **out** n, and **open** n for timed systems, corresponding to the similar untimed capabilites.

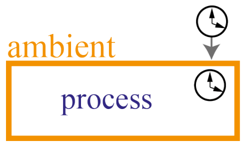

Figure 1: Figurative representation of a virtually timed ambient with a local clock.

Computing resources An ambient's processing power is defined by a *resource process* which transforms the time slices of the local clock into locally consumable resources. Processes expend the processing power of the ambient they are contained in by consuming resources. An ambient with a higher local clock speed produces more resources per parental time slice which in turn allows more work to be done for each parental time slice.

5 Weak bisimulation for timed ambients

We define weak timed bisimulation for virtually timed ambients in [5] as a conservative extension of weak bisimulation for mobile ambients as defined by Merro and Zappa Nardelli [6]. We then define a bisimulation for specific classes of processes which relaxes the condition on timing. This way we can determine if a system is faster than another and give a worst case approximation for this timing difference. We finally show that weak timed bisimulation for ambients completely characterises reduction barbed congruence for virtually timed ambients, extending a result from [6].

6 A Type System with Assumptions and Commitments

Type systems are a common technique to describe the important features of a calculus and provide a way to have the implementation of those features mechanically checked. A basic type system for mobile ambients was first defined in [3] and was mainly concerned with controlling communication and mobility. We are currently working on an assumption and commitment type system with coeffects [8] for virtually timed ambients, concerning time and resources. The type system enables the checking of constraints regarding the capacity of ambients as well as providing an upper bound for the resource usage. Additionally, we aim to obtain a *subject reduction result* for the type system for virtually timed ambients, showing that the upper bounds on resources and the number of subambients are preserved under reduction.

7 Concluding Remarks

Virtualization opens for new and interesting foundational models of computation by explicitly emphasizing deployment and resource management. We introduce virtually timed ambients, a formal model of hierarchical locations of execution with explicit resource provisioning. Resource provisioning for virtually timed ambients is based on virtual time, a local notion of time reminiscent of time slices for virtual machines in the context of nested virtualization. This way, the computing power of a virtually timed ambient depends on its location in the deployment hierarchy. To reason about timed behavior in this setting, we define weak timed bisimulation for virtually timed ambients as a conservative extension of bisimulation for mobile ambients, and show that the equivalence of bisimulation and reduction barbed congruence is preserved by this extension. Introducing an assumption and commitment type system with coeffects allows for the checking of timing and resource constraints on ambients and gives an upper bound on the resources used by a process.

References

[1] J. C. M. Baeten and J. A. Bergstra. *Real Time Process Algebra* . Technical Report CS-R 9053, Centrum voor Wiskunde en Informatica (CWI), 1990.

[2] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *Theor. Comput. Sci.*,, 240(1): 177–213, 2000.

[3] Luca Cardelli and Andrew D. Gordon. Types for the Ambient Calculus. In *Information and Computation*, 177(2:)160–194,2002.

[4] Matthew Hennessy and Tim Regan. A process algebra for timed systems. In *Information and Computation*, 117(2):221–239, 1995.

[5] E. B. Johnsen, M. Steffen, and J. B. Stumpf. A calculus of virtually timed ambients. In *Postproceedings of selected contributiions to the 23rd International Workshop on Algebraic Development Techniques (WADT 2016)*, 2017.

[6] Massimo Merro and Francesco Zappa Nardelli. Behavioral theory for mobile ambients. *J. ACM*, 52(6):961–1023, November 2005.

[7] Faron Moller and Chris Tofts. A temporal calculus of communicating systems. In J. C. M. Baeten and J. W. Klop, editors, *Proc. CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 401–415. Springer, 1990.

[8] T. Petricek, D. Orchard, and A. Mycroft. Coeffects: unified static analysis of context-dependence. In *Proceedings of International Conference on Automata, Languages, and Programming — Volume Part II*, ICALP 2013.