Journal of Logical and Algebraic Methods in Programming 94 (2018) 109–127

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Virtually timed ambients: A calculus of nested virtualization [☆]

Einar Broch Johnsen, Martin Steffen, Johanna Beate Stumpf [*]

University of Oslo, Oslo, Norway

ARTICLE INFO

Article history:
Received 28 February 2017
Accepted 5 October 2017

Keywords:
Process calculi
Ambient calculi
Models of distributed systems
Models of nested virtualization
Weak timed bisimulation

ABSTRACT

Nested virtualization enables a virtual machine, which is a software layer representing an execution environment, to be placed inside another virtual machine. Nested virtual machines form a location hierarchy where virtual machines at every level in the hierarchy compete with other processes at that level for processing time. With nested virtualization, the computing power of a virtual machine depends on its position in this hierarchy and may change if the virtual machine moves. This paper introduces the calculus of virtually timed ambients, a formal model of hierarchical locations for execution with explicit resource provisioning, motivated by these effects of nested virtualization. Resource provisioning in this model is based on virtual time slices as a local resource. To reason about timed behavior in this setting, weak timed bisimulation for virtually timed ambients is defined as an extension of bisimulation for mobile ambients. We show that the equivalence of contextual bisimulation and reduction barbed congruence is preserved by weak timed bisimulation. Simulation with time relaxation is defined to express that a system is slower than another system up to a given time bound. The calculus of virtually timed ambients is illustrated by examples.

1. Introduction

Virtualization technology enables the resources of an execution environment to be represented as a software layer, a so-called *virtual machine*. Application-level processes are agnostic to whether they run on such a virtual machine or directly on physical hardware. Since a virtual machine is a process, it can be executed on another virtual machine. Technologies such as VirtualBox, VMWare ESXi, Ravello HVX, Microsoft Hyper-V, and the open-source Xen hypervisor increasingly support running virtual machines inside each other in this way. This *nested virtualization*, originally introduced by Goldberg [16], is necessary to host virtual machines with operating systems which themselves support virtualization [5], such as Microsoft Windows 7 and Linux KVM. Nested virtualization has many uses, for example for end-user virtualization for guests, in development, and in deployment testing. Nested virtualization is also a crucial technology to support the hybrid cloud, as it enables virtual machines to migrate between different cloud providers [37].

To study the logical behavior of virtual machines in the context of nested virtualization, this paper develops a calculus of virtually timed ambients with explicit resource provisioning. Previous work on process algebra with resources typically

[☆] The work was supported by the Research Council of Norway through the CUMULUS project (project number: 240614) and through the SIRIUS Centre for Scalable Data Access (www.sirius-labs.no).

[*] Corresponding author.
E-mail addresses: einarj@ifi.uio.no (E.B. Johnsen), msteffen@ifi.uio.no (M. Steffen), johanbst@ifi.uio.no (J.B. Stumpf).

https://doi.org/10.1016/j.jlamp.2017.10.001
2352-2208/© 2017 Elsevier Inc. All rights reserved.

focuses on binary resources such as locks (e.g., [22,29]) and previous work on process algebra with time mainly considers timeouts (e.g., [31,4,28,17,6]). In contrast, time and resources in virtually timed ambients are *quantitative* notions: a process which gets *more resources* typically executes *faster*. Virtually timed ambients can be understood as locations for the deployment of processes; the resource requirements of processes executing at a location are matched by resources made available by the virtually timed ambient. The amount of resources made available by a virtually timed ambient constitutes its computing power. This amount is determined by the time slices the virtually timed ambient receives from its parent ambient. A virtually timed ambient that shares the time slices of its parent ambient with another process has less available time slices to execute its own processes.

The time model used to realize this kind of resource provisioning for virtually timed ambients is here called *virtual time*. Virtual time is provided to a virtually timed ambient by its parent ambient, similar to the time slices that an operating system provisions to its processes. When we consider many levels of nested virtualization, virtual time becomes a *local* notion of time which depends on a virtually timed ambient's position in the location hierarchy. Virtually timed ambients are mobile, reflecting that virtual machines may migrate between host virtual machines. Observe that such migration affects the execution speed of processes executed in the virtually timed ambient which moves, in the virtually timed ambients it leaves, and in the virtually timed ambient it enters. The model of resource provisioning in virtually timed ambients is inspired by Real-Time ABS [20], but extended to address nested virtualization in our calculus.

To formalize nested virtualization, notions of location mobility and nesting are essential. The calculus of mobile ambients, originally developed by Cardelli and Gordon [9], captures processes executing at distributed locations in networks such as the Internet. Mobile ambients model both location mobility and nested locations, which makes this calculus well-suited as a starting point for our work. Combining these notions from the ambient calculus with the concepts of virtual time and resource provisioning, the calculus of virtually timed ambients can be seen as a model of nested virtualization. To capture migration, virtually timed ambients will have capabilities reminiscent of those for mobile ambients, but the capabilities of virtually timed ambients need to deal with virtual time and the corresponding changes to the resource provisioning. Thus different locations, barriers between locations, barrier crossing, and their relation to virtual time and resource provisioning are important for the virtually timed ambients; the number and position of virtually timed ambients available for processing tasks influences the overall processing time of a program. This allows the effects of, e.g., load balancing and scaling to be observed using weak timed bisimulation.

Contributions. To study the effects of nested virtualization, the main contributions of this paper can be summarized as follows:

- we define a calculus of *virtually timed ambients*, to the best of our knowledge the first process algebra capturing notions of virtual time and resource provisioning for nested virtualization;
- we define *weak timed bisimulation* for virtually timed ambients, and show that weak timed bisimulation is equivalent to reduction barbed congruence [25] with time;
- we define *time relaxation* for virtually timed ambients as a simulation relation allowing deviation by a bounded amount of time.

A short version of this paper appeared in the proceedings of WADT 2016 [21].

2. Preliminaries on mobile ambients

Mobile ambients [9] have originally been introduced to represent "administrative domains" for processes. The syntax, as well as the semantics we consider, is based on [25] and largely unchanged compared to [9]. The main difference compared to [9] lies in the separation of processes into two levels, as *processes* and *systems*. This distinction is used to simplify proofs in the bisimulation section. Systems characterize the outermost layer of an ambient structure.

The syntax in Table 1 represents the basic calculus for mobile ambients. The inactive process $\mathbf{0}$ does nothing. The parallel composition $P \mid Q$ allows both processes P and Q to proceed concurrently, where the binary operator $\mid$ is commutative and associative. The restriction operator $(\nu n)P$ creates a new and unique name with process P as its scope. In the calculus, administrative domains for processes, called *ambients*, are represented by names. A process P located in an ambient named m is written $m[P]$.

Ambients can be nested, and the nesting structure can change dynamically. A change of the nesting structure is specified by prefixing a process with a *capability*. There are three basic capabilities. The input capability $in\, n$ indicates the willingness of a process (respectively its containing ambient) to enter an ambient named n, running in parallel with its own ambient; e.g., $k[in\, n.P] \mid n[Q] \rightarrowtriangle n[k[P] \mid Q]$. The output capability $out\, n$ enables an ambient to leave its surrounding (or parental) ambient n; e.g., $n[k[out\, n.P] \mid Q] \rightarrowtriangle k[P] \mid n[Q]$. The open capability $open\, n$ allows an ambient named n at the same level as the capability to be opened; e.g., $k[open\, n.P \mid n[Q]] \rightarrowtriangle k[P \mid Q]$.

2.1. Syntax

We use the following notational conventions. Parallel composition has the lowest precedence among the operators and the process $E.F.P$ is read as $E.(F.P)$. For names, the ν-operator acts as a binder and the sets of free names fn of a process