A Calculus of Mobility and Communication for Ubiquitous Computing

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

Nosheen Gul
Department of Computer Science
University of Leicester

May 2015

A Calculus of Mobility and Communication for Ubiquitous Computing

Nosheen Gul

Abstract

Ubiquitous computing makes various computing devices available throughout the physical setting. Ubiquitous computing devices are distributed and could be mobile, and interactions among them are concurrent and often depend on the location of the devices. Process calculi are formal models of concurrent and mobile systems. The work in this thesis is inspired by the calculus of Mobile Ambients and other process calculi such as Calculus of Communicating Systems which have proved to be successful in the modelling of mobility, communication and structure of systems.

We start by developing operational semantics for the calculus of Mobile Ambients and Push and Pull Ambient Calculus, and prove that the semantics are sound and complete with respect to the corresponding underlying reduction semantics. This thesis proposes a Calculus of Communication and Mobility, denoted by $\mathrm{CMC}_{\mathrm{PCA}}$, for the modelling of mobility, communication and context awareness in the setting of ubiquitous computing. $\mathrm{CMC}_{\mathrm{PCA}}$ is an ambient calculus with the *in* and *out* capabilities of Cardelli and Gordon as well the *push* and *pull* capabilities of Phillips and Vigliotti. $\mathrm{CMC}_{\mathrm{PCA}}$ has a new form of global communication similar to that in Milner's CCS. We define a new notion of behavioural equivalence for our calculus in terms of an observation predicate and action transitions. Thus, we define barbed bisimulation and congruence, and capability barbed bisimulation and congruence. We then prove that capability barbed congruence coincides with barbed congruence. We also include in the calculus a new form of context awareness mechanism that allows ambients to query their current location and surrounding. We then propose reduction semantics and operational semantics for the context awareness primitives, and show that the semantics coincide. Several case studies and a variety of small examples show the expressiveness and usefulness of our calculus.

· · · my loving mother and the sweet memories of my father

Acknowledgement

I would like to thank my supervisor Dr Irek Ulidowski for his guidance and invaluable support during my research. I believe that without his guidance none of this research would have been possible. Thank you very much Irek for showing tremendous patience in reading and evaluating my work.

I would also like to thank Prof Reiko Heckel, my second supervisor for his time to time valuable suggestions. A special thanks to Dr Emilio Tuosto, as being an expert in process calculi his suggestions have always been of great help.

I owe a great debt of thanks to Prof Thomas Erlebach for his dedication and guidance in resolving my professional and personal issues.

I cannot find words to express my gratitude to my dearest friends for their loving support throughout this journey. Thank you Ayesha for staying awake with me at nights during my migraines. Thank you very much Sab for your super-special support.

My very special thanks are due to my family, especially to my sweet parents, brothers and my lovely sister Seemrose, who have always been close to my heart throughout my stay in the UK, and they have never made me realise that I am away from my family. Very special thanks to my dear father (may his soul rest in peace) for his unconditional love and support throughout my life. I would also like to give special thanks to my dear uncle M. T. Abbasi for all his encouragement and guidance during my studies.

.....

I am particularly thankful to my two examiners Prof Maciej Koutny and Dr Emilio Tuosto for their useful suggestions and comments during and after the viva.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ubiquitous computing [79, 78] provides various computing devices available throughout the physical setting, and humans interact with networks of devices (dynamic, invisible and embedded in everyday objects). The idea behind ubiquitous computing is to surround ourselves with computing devices that are carefully adjusted to offer us hidden assistance as we navigate through our daily activities. In ubiquitous and mobile computing environment computing devices are distributed and could be mobile, and interactions among them are concurrent and often depend on the location of the devices. The idea of mobile computing deals with the computations carried out in mobile devices that are moved by the users, and communication in such setting could be global, which means that agents may interact with subagents inside other agents. Moreover, the notion of context aware computing [78, 56] is introduced as an emerging trait of ubiquitous computing where devices adapt according to the changing surroundings. Therefore, the aim of such setting is not to support global ubiquity; which is to connect all systems to form a general service domain, but rather to support context-based ubiquity.

We consider a scenario where a client intends to move from her current location to some target location inside a building. She picks up her digital device PDA, and sends a request to a server to provide a path between the two locations. The request contains the names of the two locations, and the server sends a path between these locations to the client via her PDA.

This scenario describes the main features of the ubiquitous computing setting that we aim to model in this thesis, namely

- physical mobility, both active and passive
- global communication, and
- location modelling.

Since the client moves around holding her digital device, we shall use two forms of mobility, namely mobility using the *in* and *out* capabilities as in [11] and mobility modelled by the *push* and *pull* primitives as in [54]. Moreover, the server and the client communicate globally and, hence, we introduce a new form of global communication similar to that in [43]. Location modelling has become an important aspect of the ubiquitous computing setting, where location awareness is considered an important feature that provide communication among various ubiquitous computing devices. We describe below in more detail how these features of ubiquitous computing will be modelled.

Process algebras are used to model formally concurrent systems, and provide specification for communication, interaction and synchronisation between collections of independent agents [43]. Structural Operational Semantics, SOS for short, [55, 51, 73, 71, 72] is used to define the behaviour of a system in terms of the behaviour of its parts. It is a standard approach of defining the semantics of a system by means of transition rules [42, 55]. Several process calculi were developed to model concurrency, communication and distributed systems, most notably CSP [29, 30], CCS [43, 42] and ACP [4]. These process calculi have no primitives to describe certain aspects of behaviours of discussed above scenario, for example mobility and locations. The idea of mobile code has been formalised by Milner in π-calculus [44, 49, 50]. π-calculus can model a changing connectivity of interactive systems. The calculus is an extension of CCS, and has been developed for modelling concurrent systems that pass around resources that can be reused later.

The aforementioned process calculi do not represent directly physical mobility of devices and their locations or surroundings. In the ubiquitous computing setting it is beneficial to model interactions among mobile agents that communicate globally. Communication in such settings could be global, which means that agents may interact with subagents inside other agents. Moreover, the structures in such settings may be active (that could move on their own) or passive (may only move when active structures carry them), and may need to have knowledge of their current location and surrounding.

This thesis continues research on Mobile Ambients [11, 10], MA for short, and process calculi which have proved to be useful in the modelling of mobility, communication and structure of systems. We develop a *Calculus of Mobility and Communication*, CMC_{PCA} for short. In our calculus mobility, global communication and location awareness are considered as first class entities. Therefore, we model locations and mobility as in Mobile Ambients. According to [11] MA was proposed to model mobility and locations that could not be modelled directly by other tradi-

tional calculi like CCS. The two aspects of mobility that MA models are; mobile computing (that deals with physical mobility of computing devices), and mobile computation (that deals with logical mobility of code between devices). In MA the communication between ambients is modelled by the *open* capability in combination with *in* and *out* capabilities. In our calculus, we drop the *open* capability since we introduce a new mechanism of direct and global communication.

We start by presenting a new calculus, *Calculus of Mobility* (CM for short), that inherits its mobility primitives from MA. Recently, a number of operational semantics have been developed for Mobile Ambients and its variants as, for example, in [16, 36, 37, 40, 38, 34, 8]. We develop an operational semantics for CM which is inspired by [36, 37]. Our SOS rules use concretions $\nu\tilde{m}\langle P\rangle Q$ as introduced by Milner [44] and later used in [36, 37, 40]. We show that the operational semantics of CM coincides with the standard reduction semantics.

In past few years, several variants of MA have been introduced [6, 53, 54, 41, 35, 36]. To the best of our knowledge, these ambient calculi do not support a direct communication of an agent with a subagent inside another agent. Communication may only happen between the two adjacent agents, namely communication between parent and child or between siblings. Therefore, we extend CM by adding a direct and global form of communication as in Milner's CCS [43], and hence we obtain Calculus of Mobility and Communication(CMC). CMC introduces A to ambient structure which shall be useful for the global communication between agents. We change ambient name m in CM to m_A where A is the set of ports that the ambient communicates on. In order to illustrate how our global communication works, we consider process

$$(\overline{a}.0 \mid b.0) \mid m_A[n_B[a.0] \mid k_C[\overline{b}.0]], \text{ for } a, b \in A \; a, b \in B \text{ and } b \notin C.$$

Here, ambient n_B communicates globally via port a since $a \in A, B$, whereas k_C cannot communicate via b since $b \notin C$. We obtain these intuitive transitions

$$(\overline{a}.0 \mid b.0) \mid m_A[n_B[a.0] \mid k_C[\overline{b}.0]] \xrightarrow{\tau} (0 \mid b.0) \mid m_A[n_B[0] \mid k_C[\overline{b}.0]]$$

$$(\overline{a}.0 \mid b.0) \mid m_A[n_B[a.0] \mid k_C[\overline{b}.0]] \stackrel{\tau}{\nrightarrow} (\overline{a}.0 \mid 0) \mid m_A[n_B[a.0] \mid k_C[0]].$$

Phillips and Vigliotti introduced the Push and Pull Ambient Calculus [54], PAC for short, that allows modelling of certain security issues. The calculus models mobility in a different way, namely ambients can push away or pull in other ambients by the $push_m$ n and $pull_m$ n capabilities. In our setting we also aim at modelling

the behaviour of passive mobile structures as in [54]. We extend CMC by adding additional mobility primitives to CMC and obtain CMC_P. We add $push(m_A)\ n_B$ and $pull(m_A)\ n_B$ capabilities to move passive ambients around. The capability $push(m_A)\ n_B$ allows an ambient m_A to move n_B out of its boundary, whereas $pull(m_A)\ n_B$ allows m_A to pull n_B inside its body. The calculus now models both forms of active and passive mobile structures which may communicate globally. Consider

$$\overline{a}.0 \mid k_C[client_A[pull(client_A)\ device_B.a.0] \mid device_B[0]], \text{ for } a \in A, B, C.$$

Here, $client_A$ may only communicate globally via a if he picks up $device_B$. The ambient $client_A$ therefore pulls $device_B$ inside its scope by $pull(client_A)\ device_B$ capability. This is shown by the following transition

$$\overline{a}.0 \mid k_C[client_A[pull(client_A)\ device_B.a.0] \mid device_B[0]] \xrightarrow{\tau}$$
$$\overline{a}.0 \mid k_C[client_A[a.0 \mid device_B[0]]]$$

Now communication may take place as shown below:

$$\overline{a}.0 \mid k_C[client_A[a.0 \mid device_B[0]] \xrightarrow{\tau} 0 \mid k_C[client_A[0 \mid device_B[0]]$$

Furthermore, technologies have made it possible to detect a user's presence or a position or other attributes concerning the user. Therefore, context-awareness and location-awareness have become important features of ubiquitous computing environments. The idea of context aware computing has originated in [78]. It enables an application to adapt to the changes in its environment and location. In smart indoor settings, location is considered an important entity for providing communication among various portable and static structures.

We consider location as one of the most typical forms of context. Context aware applications basically use location of people and computing devices as their main source of contextual information so that the personalised services are executed accordingly. We further proceed to extend our calculus to also include some of the basic mechanism of context awareness. We add $ploc(x).P$ and $sloc(x).P$ to CMC_P and obtain CMC_PCA. The construct $ploc(x)$ enquires the parent's name, whereas $sloc(x)$ enquires the sibling's name. These primitives enable agents to acquire the name of their parent and sibling, and pass it as x to P. We develop operational semantics for the extended calculus, and show that the operational semantics corresponds to the reduction semantics. The context features of CMC_PCA empower ambients to start interaction in some scenarios, and adapt to the changes as required. For example,

consider

$$server[P_s] \mid building[r_1[client_A[P_c] \mid dev_B[P_d]] \mid r_2[P_r]]$$

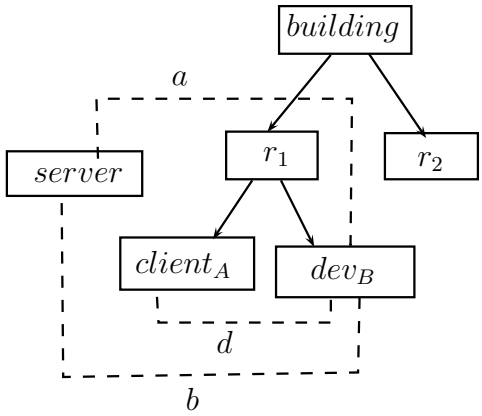
$$P_c \stackrel{def}{=} pull(client_A)\ dev_B.ploc(x).\overline{a}(x, r_2).c(z_1).0$$

$$P_d \stackrel{def}{=} a(x_1, y_1).\overline{b}(x_1, y_1).d(z).\overline{c}(z).0$$

$$P_s \stackrel{def}{=} b(x_2, y_2).\overline{d}(path(T, x_2, y_2)).0$$

P_r is the agent running inside r_2

A graphical representation of this setting is given below.



In this setting, *server* delivers services to $client_A$ based on its current location. We assume $a, b, c, d \in A$ and $a, b, c, d \in B$. We write $m[P]$ instead of $m_A[P]$ whenever ambients allow communication on all visible ports, so $server, building, r_1, r_2$ may communicate on any ports. The ambient $client_A$ first pulls the device by $pull(client_A)$ dev_B capability. Then next $client_A$ acquires its parent's name r_1 by its $ploc(x)$ capability and sends it (r_1) and the target location r_2 to *server* via dev_B. Based on the two locations *server* calculates the path *out* r_1.*in* r_2 using $path(T, x_2, y_2)$, where T represents the tree structure of our setting, and delivers it to $client_A$ via dev_B. This enables $client_A$ to move from r_1 to r_2 by following the appropriate sequence of capabilities. This example models the scenario that we have presented at the beginning of this chapter.

In this thesis several case studies and a variety of small examples are given that show the expressiveness and usefulness of our calculi. For example, the intelligent hospital setting case study, where services follow mobile ambients uses global communication and the *in* and *out* capabilities, and the interactive shopping mall case study illustrates the usefulness of global communication, *push* and *pull*, and *ploc(x)* and *sloc(x)* features of $\mathrm{CMC}_{\mathrm{PCA}}$.

1.1 Contributions

The first contribution is the Calculus of Mobility (CM) which models the mobile structures in the setting of ubiquitous computing. The thesis builds on recent research in the area of Mobile Ambients and other process calculi that have proved useful in the modelling of mobility, communication and structure of systems. The calculus CM inherits its syntax and reduction semantics from Mobile Ambients [10]. An operational semantics for CM is developed and is proved sound and complete with respect to the standard reduction semantics.

We then add a new form of global communication similar to that in Milner's CCS [43] to CM. This gives us a Calculus of Mobility and Communication (CMC). In CMC we modify $m[P]$ to $m_A[P]$ by adding a set of ports A to it that allows agents executing inside ambients to communicate on. The calculus is developed with real-world applications in mind and its usefulness is illustrated in case studies and small examples. A new notion of behavioural equivalence for CMC is introduced in terms of α-transitions ($\xrightarrow{\alpha}$) and observation predicate. We define barbed bisimulation congruence and capability barbed bisimulation congruence, and show that the congruence relations of the two forms of barbs imply each other.

We proceed to extend CMC with additional mobility primitives, namely the *push* and *pull* capabilities, and thus obtain CMC_P. Intuitively the extended calculus models passive and active mobile structures in the setting of ubiquitous computing. We develop a new operational semantics, and the first such operational semantics as far as we know, for CMC_P and prove that the semantics is sound and complete with respect to the standard reduction semantics.

We finally extend CMC_P to a new calculus CMC_PCA by adding context awareness primitives. We add basic forms of context awareness mechanism, namely location awareness, that gives the parent name of an ambient and sibling awareness, that gives the name of a sibling ambient. We develop reduction and operational semantics for the additional features of our calculus. We show that the operational semantics is sound with respect to the reduction semantics.

The expressiveness and usefulness of the calculus is exemplified by several case studies where the relevant constructs are used to model various features of the calculus. For example, in our *Path* case study in Section 5.3.1, server calculates path as a sequence of *in* and *out* capabilities between two locations inside a building and delivers it to the mobile agents to move from one location to another whenever required. Furthermore, the intelligent hospital case study in Section 5.3.2 models that services follow doctor while he moves around the building and deals with the patients. Server communicates with doctor globally via fixed or mobile computing

devices that are distributed around the building, and delivers services to the appropriate device provided that the doctor is in the same room as the device. We also consider scenarios to show the requirement of passive mobile structures in the setting of ubiquitous and mobile computing, namely a mobile device sends a message to its user and the user cannot view the message unless he picks up the device. This is modelled by the *pull* capability and ambients tagged with an appropriate set of communication actions. We also provide an interactive shopping mall case study in Section 7.5.1 that illustrates the usefulness of global communication, *push* and *pull*, and *ploc(x)* and *sloc(x)* features of CMC_{PCA}. The mall consists of a number of retail outlets, clients and computing devices. There is a server that delivers services to clients on requests via devices which are distributed inside the mall. Moreover, we device a small case study in Section 7.5.2 where smart devices automatically switch their *ON* and *OFF* modes depending on their location and the users who are using them. The constructs *push*, *pull* and *ploc(x)* are considered more relevant in combination with global communication.

This thesis presents a number of case studies that illustrate expressiveness of the calculus. The systematic addition of new primitives smoothly increases the usefulness of the calculus. Summarising, due to the expressiveness power of CMC, the calculus could be used to model a variety of features that could be illustrated in the settings ubiquitous computing. These features are agents' mobility, system changing structures, location, global communication between agents and context awareness. The final calculus CMC_{PCA} is saturated with a number of primitives which makes the calculus more powerful, that is, CMC_{PCA} is useful to model other scenarios in general.

For example, *A Firewall Access* scenario, taken from [11], represents an access authorisation of a mobile ambient by means of *pilot* and *wrapper* ambients. The ambient firewall keeps its name secret, and uses a *pilot* ambient to share its name with the target agent. The setting where a single target agent gets access across the firewall is defined as follows:

$$Firewall \stackrel{def}{=} (\nu w)w[k[out\ w.in\ k'.in\ w] \mid open\ k'.open\ k''.P])$$
$$Agent \stackrel{def}{=} k'[open\ k.k''[Q]]$$

In this example an agent *Agent* crosses a firewall w, and after a number of reductions the agent gains access to the firewall contents. It is not guaranteed which particular agent may enter the firewall but the access authorisation is granted by the firewall by sharing its name with the intended agent. These agents may be composed in

parallel as follows:

$$Agent \mid Firewall \equiv (\nu w)k'[open\ k.k''[Q]] \mid w[k[out\ w.in\ k'.in\ w] \mid open\ k'.open\ k''.P])$$

where $fn(P) \cup fn(Q) \cap \{k, k', k''\} = \phi$ and $w \notin fn(Q)$.

In this setting, k, k', k'' are passwords that are previously agreed between the two interacting agents, where k and k' act as *pilot* and *wrapper* ambients, and k'' encloses the agent Q. The *open* capabilities dissolve the ambients k, k', k'', hence finally, we obtain $(\nu w)w[Q \mid P]$.

Such a scenario can be rewritten in CMC_{PCA} as follows:

$$Network \stackrel{def}{=} \nu(a,b)(net_A[w_B[ploc(x).\overline{a}(x) \mid \overline{b}.P'] \mid P''], \text{ for } a,b \in B, a \in A, b \notin A$$

$$Agent \stackrel{def}{=} \nu(a,b)(m_C[a(y).in\ y \mid b.Q'], \text{ for } \{a,b\} \in C$$

$$Network \mid Agent \equiv \nu(a,b)(net_A[w_B[ploc(x).\overline{a}(x) \mid \overline{b}.P'] \mid P''] \mid m_C[a(y).in\ y \mid b.Q'])$$

In this example, initially mobile ambient m_C and firewall w_B cannot communicate globally via b, since $b \notin A$. The network allows only the authorised agents to interact with the enclosing contents. The access authorisation is assigned by the firewall which controls the incoming agents based on previously set agreement. The primitive $ploc(x)$ allows the firewall to have the name of its parent (net_A in this case). It then shares the network name net_A with the target agent. The ambient m_C after receiving the value via a becomes able to enter the network by its $in\ net_A$ capability.

Now communication between m_C and the firewall w_B may take place as shown below:

$$\nu(a,b)(net_A[w_B[ploc(x).\overline{a}(x) \mid \overline{b}.P'] \mid P''] \mid m_C[a(y).in\ y \mid b.Q']) \xrightarrow{\tau}$$

$$\nu(a,b)(net_A[w_B[\overline{a}(net_A) \mid \overline{b}.P'] \mid P''] \mid m_C[a(y).in\ y \mid b.Q']) \xrightarrow{\tau}$$

$$\nu(a,b)(net_A[w_B[\overline{b}.P'] \mid P''] \mid m_C[in\ net_A \mid b.Q']) \xrightarrow{\tau}$$

$$\nu(a,b)(net_A[m_C[b.Q'] \mid w_B[\overline{b}.P'] \mid P'']).$$

This example shows the usefulness of CMC_{PCA} primitives in translating such scenarios. The communication ports can conveniently be used for global communication between ambients. This may also place a restriction at the level of ambients where the communication is not intended. Similarly, the other constructs of CMC_{PCA} are suitable in modelling and translating such scenarios if applied appropriately.

1.2 Thesis Outline

The rest of the thesis has been organised as follows:

2. *Background and Related Work*

   This chapter describes background information and related work.

3. *Towards a Calculus of Mobility*

   In this chapter we discuss the syntax and semantics of MA. We modify the definition of ambient and write $m_A[P]$, for some set of actions A, instead of $m[P]$ and call it as, a Calculus of Mobility (CM). We develop operational semantics for CM and prove that the semantics is sound with respect to the standard reduction semantics. We show by presenting three examples that the proposed operational semantics is not complete with respect to the standard reduction semantics.

4. *Operational Semantics of a Calculus of Mobility*

   We develop a new better operational semantics for CM. The SOS rules use concretions $\nu\tilde{m}\langle P\rangle Q$ [36, 37, 40]. We prove that the operational semantics is sound and complete with respect to the standard reduction semantics.

5. *The Calculus of Mobility and Communication*

   We add global communication to CM, and obtain thus a Calculus of Mobility and Communication (CMC). We develop an operational semantics for CMC which include new SOS rule for global communication. The usefulness of CMC is illustrated in two bigger case studies. We also define a new notion of behavioural equivalence for CMC, in terms of observation predicate and action transitions.

6. *Operational Semantics for Push and Pull Ambient Calculus*

   In this chapter CMC is extended with additional passive mobility primitives to give CMC_P. We introduce the *push* and *pull* primitives, and develop a new operational semantics for our calculus. The operational semantics is proved sound and complete with respect to the standard reduction semantics. A number of small examples show the usefulness of the extended calculus.

7. *Context-Awareness: Location and Surrounding*

   In this chapter we introduce a basic form of context awareness. We extend CMC_P by adding the *ploc* and *sloc* primitives that help ambients to gain a

knowledge of their parent's and sibling's identity. We develop an operational semantics for the final calculus $\mathrm{CMC}_{\mathrm{PCA}}$ which is sound with respect to the standard reduction semantics. The usefulness of $\mathrm{CMC}_{\mathrm{PCA}}$ is shown by several examples.

8. *Conclusion and Future Work*

   The last chapter summarises our results and gives an evaluation of the work done. We then give several directions to future work.

Chapter 2

Background and Related Work

This chapter presents the background information that is needed to understand the formal specification and behaviour of mobile and communicating agents.

2.1 Process Algebra

Process algebras are used to formally model concurrent systems, and provides formalisms for specifying communications, interactions, and synchronizations between collections of independent agents. Structural Operational Semantics (SOS for short) [55, 51, 73] is used to define the behaviour of a system in terms of the behaviour of its parts. In particular, SOS specification defines the behaviour of a program in terms of a (set of) transition relation(s). Practical introduction to the formal specification of concurrent systems is given by Robin Milner in his book *Communication and Concurrency* [43]. The precise notion of communicating systems is given by providing operational meanings to the syntactic constructions. Theoretical aspects of the concurrency are illustrated by a variety of examples in [43, 42, 15]. This helps the readers to combine both aspects (practical and theoretical) according to their particular tastes and requirements. Robin Milner has given an excellent introduction to the mobile communicating systems with more emphasis on applications and less upon the behavioural theory in his book *"Communicating and Mobile Systems: The π-calculus"* [44]. He clearly describes the applications of mobile systems with practical examples. π-calculus is presented as a model of the changing connectivity of the interactive systems.

2.1.1 Structural Operational Semantics and Transition Rules for CCS

In this section we shall consider the Calculus of Communicating Systems, its syntax, and then the transition rules for CCS operators. We shall also present some expressions and inference trees showing the application of CCS transition rules. CCS shall be used in Chapter 5, therefore we assume that the reader is familiar with CCS and the description will be brief.

Process algebra is used to formally model concurrent systems. Operational semantics allow us to describe communication, interaction and synchronisation between a collection of independent agents or processes [43]. Concurrent systems can be best described as networks of agents or processes. Several process calculi have been developed to model concurrent interactions among distributed agents. Since concurrency and interactions are considered the basic features that we intend to model, CCS is a suitable formalism. The calculus was introduced by Robin Milner [43, 42] in the early 1980s with the aim of modelling concurrent behaviour of communicating systems.

The Syntax of CCS

The Calculus of Communicating Systems focuses on a very simple paradigm of synchronous handshakes. Two processes $\overline{on}.P$ and $on.Q$, when composed in parallel $\overline{on}.P \mid on.Q$ may execute by synchronising via port on and after the execution proceed as P and Q respectively, as shown by the transition $\overline{on}.P \mid on.Q \xrightarrow{\tau} P \mid Q$.

The syntax of CCS as in [43, 70] is as follows: We assume that $\mathcal{A}$ is an infinite set of *names*, which is ranged over by a, b,c,.... The set of *co-names* is denoted by $\overline{\mathcal{A}}$, and is ranged over by $\overline{a}$, $\overline{b}$, $\overline{c}$, ... We set $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$, where $\mathcal{L}$ is the set of *labels*, ranged over by l, l'. We extend complementation to the whole of $\mathcal{L}$, so that $\overline{\overline{a}} = a$. We shall need one more *special* action, namely the *internal, silent* action τ, which is not a member of $\mathcal{L}$. An infinite set *Act* comprises all possible actions that an agent can perform and α, β range over it. *Act* also includes τ, which is a single completed action of composite agents. So $Act = \mathcal{L} \cup \{\tau\}$. In the basic calculus, labels have no value parameters. We further assume

- $\mathcal{X}$ – set of *agent variables*, ranged over by $X, Y, \ldots,$
- $\mathcal{K}$ – set of *agent constants*, ranged over by $A, B, \ldots,$
- We use I or J for *indexing sets*, for example $\{E_i : i \in I\}$ is a family of expressions indexed by I.

The set $\mathcal{E}$ of *agent expressions* is the *smallest set* which includes $\mathcal{X}$ and $\mathcal{K}$ and contains the following expressions, where E, E_i are already in $\mathcal{E}$:

1. $\alpha.E$, a Prefix ($\alpha \in Act$)
2. $\sum_{i \in I} E_i$, a Summation (I an indexing set)
3. $E_1|E_2$, a Composition
4. $E\backslash L$, a Restriction ($L \subseteq \mathcal{L}$)
5. $E[f]$, a Relabelling (f is a relabelling function)

Example 2.1. Consider a simple vending machine originally proposed by Hoare [30]. We consider a big chocolate costs 2p, little chocolate costs 1p, and only these coins are accepted. One way of defining the behaviour of vending machine V is in terms of its interaction with external environment.

$$V \stackrel{def}{=} 2p.big.collect.V + 1p.little.collect.V$$

This expression shows the behaviour of vending machine in terms of interaction with environment using ports $1p, 2p, collect, big$ and $little$. The behaviour of the system is quite restricted, for example, to get big chocolate one cannot insert two 1p coins, or machine will not give little chocolate if 2p coin is inserted.

Labelled Transition Semantics of CCS

Operational semantics is a formal way of defining the meaning of various agent expressions in terms of all their possible transitions [28, 70, 80]. Structural Operational Semantics is used to define the behaviour of a system in terms of the behaviour of its parts [55]. The behaviour of an agent is represented in terms of a graph or tree or, more formally, by a Label Transition System (LTS) [43]. Formally, an LTS is a tuple $(S, T, \{\stackrel{t}{\rightarrow} : \ t \in T\})$ where S is a set of states, T is a set of transition labels, and $\stackrel{t}{\rightarrow} \subseteq S \times S$, for $t \in T$, is a family of transition relations. The transition rules, or SOS rules, for CCS operators are given below:

$$\text{(Act)}\quad \frac{}{\alpha.E \xrightarrow{\alpha} E} \qquad \text{(Sum}_j\text{)}\quad \frac{E_j \xrightarrow{\alpha} E'_j}{\sum_{i\in I} E_i \xrightarrow{\alpha} E'_j}\ (j \in I)$$

$$\text{(Com}_1\text{)}\quad \frac{E \xrightarrow{\alpha} E'}{E|F \xrightarrow{\alpha} E'|F} \qquad \text{(Com}_2\text{)}\quad \frac{F \xrightarrow{\alpha} F'}{E|F \xrightarrow{\alpha} E|F'}$$

$$\text{(Com}_3\text{)}\quad \frac{E \xrightarrow{l} E' \quad F \xrightarrow{\overline{l}} F'}{E|F \xrightarrow{\tau} E'|F'} \qquad \text{(Res)}\quad \frac{E \xrightarrow{\alpha} E'}{E\backslash L \xrightarrow{\alpha} E'\backslash L}\ (\alpha,\ \overline{\alpha} \notin L)$$

$$\text{(Rel)}\quad \frac{E \xrightarrow{\alpha} E'}{E[f] \xrightarrow{f(\alpha)} E'[f]} \qquad \text{(Con)}\quad \frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'}\ (A \overset{def}{=} P)$$

(Act) - $\alpha.E$ shows that it has a transition from the initial state $\alpha.E$ to state E without any hypotheses (contain several transitions of the components).

(Sum$_j$) - $\sum_{i\in I} E_i$ has an action α if any one summand E_j (where $j \in I$) has an action α. For $I = \emptyset$, there is no rule so we define 0 as $\sum_{i\in\phi} E_i$. This means that 0 does not have any transitions.

(Com1, Com2) - In expression $E|F$ or $F|E$ the components E and F can act concurrently with, and independently of, each other.

(Com3) - It represents that components E and F of $E|F$ may communicate with each other through complementary actions (ports).

(Res) - When a port l ($\in L$) is restricted in an agent expression E, there will be no interaction between E through port l with any other agent outside E. Thus, E cannot perform a restricted action l.

(Con) - Finally, if $A \overset{def}{=} a.A'$, then, by using transition rule Con, we can infer $A \xrightarrow{a} A'$ from $a.A' \xrightarrow{a} A'$.

Inference Trees

Transition rules are used to justify the validity of transitions. For example, the transition $((a.E + b.0) \mid \overline{a}.F)\ a \xrightarrow{\tau} (E \mid F)\backslash a$ is derived as follows:

Inference trees demonstrate the relationship between transition semantics and the transition graphs. The SOS rules are used to prove or disprove the validity of transitions of agent expressions. We consider the agent expression

$$((a.0 + \overline{b}.c.0) \mid (a.b.0 + c.d.0)[d/a][b/d])\backslash b$$

$$\dfrac{\dfrac{\dfrac{}{a.E \xrightarrow{\alpha} E}\,\text{Act}}{a.E + b.0 \xrightarrow{\alpha} E}\,\text{Sum}_1 \quad \dfrac{}{\overline{a}.F \xrightarrow{\alpha} F}\,\text{Act}}{(a.E + b.0) \mid \overline{a}.F \xrightarrow{\tau} E \mid F}\,\text{Com}_3}{((a.E + b.0) \mid \overline{a}.F)\backslash a \xrightarrow{\tau} (E \mid F)\backslash a}\,\text{Res}$$

and construct the inference tree that proves valid τ-transition of the expression. The inference tree is given in Figure 2.1.

$$\dfrac{\dfrac{\dfrac{}{\overline{b}.c.0 \xrightarrow{\overline{b}} c.0}\,\text{Act}}{a.0 + \overline{b}.c.0 \xrightarrow{\overline{b}} c.0}\,\text{Sum}_2 \quad \dfrac{\dfrac{\dfrac{\dfrac{}{a.b.0 \xrightarrow{a} b.0}\,\text{Act}}{a.b.0 + c.d.0 \xrightarrow{a} b.0}\,\text{Sum}_1}{(a.b.0 + c.d.0)[d/a] \xrightarrow{d} b.0[d/a]}\,\text{Rel}}{(a.b.0 + c.d.0)[d/a][b/d] \xrightarrow{b} b.0[d/a][b/d]}\,\text{Rel}}{(a.0 + \overline{b}.c.0) \mid (a.b.0 + c.d.0)[d/a][b/d] \xrightarrow{\tau} (c.0 \mid b.0[d/a][b/d]}\,\text{Com}_3}{((a.0 + \overline{b}.c.0) \mid (a.b.0 + c.d.0)[d/a][b/d])\backslash b \xrightarrow{\tau} (c.0 \mid b.0[d/a][b/d])\backslash b}\,\text{Res}$$

Figure 2.1: Inference tree

2.1.2 Mobile Ambients

The Calculus of Mobile Ambients, MA for short, has been introduced by Cardelli and Gordon as a model for mobile computations that are distributed [12, 11]. Mobile Ambients is a concurrent process calculus, where the notion of ambient is used to model various structures that are distributed and mobile. According to [11], MA was developed to model two different aspects of mobility, namely mobile computing; concerning computation that is carried out in mobile devices, and mobile computation; concerning mobile code that moves between computing devices. An ambient is a bounded place in which computations occur [12, 11, 10], it has a tree structure possibly containing sub-ambients. Furthermore, MA aims at describing all these aspects of mobility within a single uniform framework consisting of mobile agents, interactions among them, and ambients' mobility. The authors describe in [11] that the inspiration for the MA comes from the potential for mobile computation over the World-Wide Web and the main difficulty with mobile computation on the Web is the handling of administrative domains. Therefore, at the most fundamental level MA captures notions of locations, of mobility and of authorization to move [11, 12]. Ambients are named terms of the form $n[P]$ where n is a name and P a process. Ambients may enter or exit named ambients by their *in* n and *out* n capabilities. The ambient's *open* capability dissolves its boundary so that the communication may

take place. Processes can be composed in parallel as in $P \mid Q$, represent restricted names as in $(vm)P$, exercise a capability as in $C.P$ or do nothing as in 0.

In our calculus mobility, locations and global communication are considered as first class entities, so in order to model locations and mobility, Mobile Ambients is a suitable formalism. Therefore, we review Mobile Ambients, including its syntax and reduction semantics, in Chapter 3.

2.1.3 Boxed Ambients

The Calculus of Boxed Ambients, BA for short, [6] is a variant of Mobile Ambients that inherits mobility primitives, namely *in* and *out* capabilities, from MA and drops *open* capability to avoid certain risks. For example, *open* action completely dissolves the boundary of an ambient, merging the processes executing inside the ambient with outer environment. According to [6], Boxed Ambients proposes a new communication mechanism where the additional communication primitives complement the existing constructs of MA in an effective manner. For example, communication between ambients can be local as in MA, the new constructs allow direct communication between a parent and a child, which is across the ambients boundaries. The additional syntax given in [6] for the Boxed Ambient is given below:

Pattern input	$(x)^{\eta}P$
Synchronous output	$\langle M\rangle^{\eta}\, P$
Input from child ambient n	$(x)^{n}P$
Output to the parent ambient	$\langle M\rangle^{\uparrow}P$

For mobility, reduction rules are inherited from Mobile Ambients [10], while communication may be local as in Mobile Ambient, namely $(x)\, P|\, \langle M\rangle\, Q \rightarrow P\,\{x := M\}\,|Q$. Furthermore, BA allows an ambient to pull an input from a child n via $((x)^{n})$. A child can also pull an input via $((x)^{\uparrow})$ from a parent, and correspondingly with input/output swapped. This is represented by the following reductions.

(input n)	$(x)^{n}\, P|n\,[\langle M\rangle\, Q|R] \rightarrow P\,\{x := M\}\,|n\,[Q|R]$
(input $\uparrow$)	$\langle M\rangle P|n\left[(x)^{\uparrow}\, Q|R\right] \rightarrow P|n\,[Q\,\{x := M\}\,|R]$
(output n)	$\langle M\rangle^{n}\, P|n\,[(x)\, Q|R] \rightarrow P|n\,[Q\,\{x := M\}\,|R]$
(output $\uparrow$)	$(x)\, P|n\left[\langle M\rangle^{\uparrow}\, Q|R\right] \rightarrow P\,\{x := M\}\,|n\,[Q|R]$

Several extensions of BA have been introduced [53, 41, 7, 8]. In particular, Safe Boxed Ambients in [41] uses ambients co-capabilities that help in controlling ambients access across the boundaries. Channel Ambient Calculus in [53] uses channels

as first class entities. It allows ambients to move in and out over the channels.

2.1.4 Channel Ambient Calculus

Channel Ambient Calculus [53] is a variant of Boxed Ambients in which channels are defined as first class entities. It allows ambients to move in and out over the channels and interact using named channels. The constructs of Channel Ambients are explained in the following example taken from [53]:

$$Network \mid client\boxed{C \mid \nu login(server.request\,\langle client, login\rangle \mid \overline{in}\,login.P)}$$
$$\mid server\boxed{S \mid !\overline{out}\,logout \mid !request^{\uparrow}(c,x).service\boxed{out\ logout.inc.x.Q}}$$

Here, client creates a new *login* channel ($\nu login$) and sends its name and *login* to *server* on *request* channel. At the same time *client* allows an ambient to enter through *login* channel ($\overline{in}\,login$). In parallel *server* is ready to receive request, it creates new service ambient that leaves server through *logout* channel and enters client through *login* channel. Processes P of *client* and Q of *service* start execution once *service* enters *client*. Additionally, in parallel *Network* can contain multiple clients, and *server* can handle multiple requests simultaneously.

Some of the additional primitives given by Channel Ambient Calculus in [53] are:

$in\ a.x$	Entering an ambient through channel x
$out\ x$	Leaving an ambient through channel x
$\overline{in}\,x$	Accepting an ambient through channel x
$\overline{out}\,x$	Releasing an ambient through channel x

Another syntactical difference is that $!\alpha.P$ has been for replicated actions instead of general replication $!P$.

2.1.5 Push and Pull Ambient Calculus

Push and Pull Ambient Calculus (PAC) has been developed by Phillips and Vigliotti [54, 77]. The basic idea of PAC relies on the $push_m\ n$ and $pull_m\ n$ capabilities instead of the $in\ m$ and $out\ m$ capabilities of Cardelli and Gordon's Mobile Ambients. Unlike in Mobile Ambients, in PAC, it is not possible for any ambient to enter or leave other, but an ambient has the control to pull in or push away other ambients. More specifically, PAC is defined with the intention to allow better modelling of

certain security issues. The usefulness of PAC as discussed in [54] is illustrated by the following example:

$$Client\boxed{EnterServer} \mid Server\boxed{Program}$$

reduces to:

$$Server\boxed{Client\boxed{EnterServer} \mid Program}$$

In the above settings the client enters the server and the server cannot avoid the client if it is a malicious agent. This scenario is rewritten in [54] as follows:

$$Client\boxed{Program'} \mid Server\boxed{PullClient \mid Program}$$

reduces to:

$$Server\boxed{Client\boxed{Program'} \mid PulledClient \mid Program}$$

Now the host ambient decides to pull in or push away ambients whenever required. The reduction relations for $push_m\ n$ and $pull_m\ n$ capabilities given in [54, 77] are as follows:

$$m[pull_m\ n.P \mid Q] \mid n[R] \rightarrow m[P \mid Q \mid n[R]],$$

where ambient m pulls n inside its boundary, and

$$m[push_m\ n.P \mid n[Q] \mid R] \rightarrow m[P \mid R] \mid n[Q],$$

where ambient m pushes n out of its boundary.

2.2 Context Awareness

Context awareness is an emerging feature in the setting of ubiquitous and mobile computing. The idea of context aware computing has been originated from Weiser's vision of ubiquitous computing [78]. There are various types of contexts [31] that may be used in ubiquitous computing setting, namely physical contexts (location, space), environmental contexts (temperature, light), human contexts (mood, health), system contexts (network traffic) and many more. In [31] context-based infrastructure has been proposed that allow applications to specify different behaviours in different contexts easily.

Poslad in his book *Ubiquitous Computing: smart devices, environments and interaction* [56] addressed number of theoretical concepts in the context of ubiquitous

computing. The design aspect of context-awareness is about how to know the users state and surrounding and how to modify behaviour of the system according to the users' requirements. The aim of ubiquitous computing is to support context-based ubiquity. Context-based ubiquity includes many benefits, namely to limit the resources needed to deliver the ubiquitous services, limiting the choice of access from all possible services to only useful services, avoiding overburdening the user with too much information. A key design issue for context-aware system is to balance the degree of user control and awareness of their environments. Contexts may exhibit a range of spatial characteristics, alternative representations, generate huge volumes of data and reduce the users privacy.

Pervasive computing settings allow computing devices to disappear in the background and perform computations. These computations could be mobile and context-aware [17, 18, 52, 58, 57]. The applications executing in pervasive computing setting need to be context sensitive so that they behave rapidly according to the changing environment. The context aware model in [58] has been developed for ubiquitous computing setting using first order logic. The model allows complex rules involving contexts to be written such that the applications behave rapidly in different contexts. Siewe, Cau and Zedan proposed a Calculus of Context Aware Ambients (CCA) [17] which is based on the Calculus of Boxed Ambients [6]. The CCA describes the context-awareness requirements of the mobile systems. It introduces the notion of context expression that constraints the ambient capability. The context guarded capability has the form $k?M$, where k is a context expression and M is a capability. This capability can only be performed if the environment satisfies its guard. The rest of the syntax of the CCA is similar to that of Boxed Ambients.

2.3 Location Modelling

This section describes the applications of location awareness in the setting of ubiquitous and mobile computing. Location is one of the most typical forms of context. Context aware applications basically use location of people and computing devices as their main source of contextual information so that the personalised services are executed accordingly. Satoh has researched spatial organisation of systems [63, 62, 64, 65] and concluded that technological advancements have enabled computing devices to become aware of their surroundings. Location awareness has turned out to be useful in many applications, in particular, in determining position, navigation, routing, tracking, monitoring of ubiquitous computing devices, and many more. In [64], a general location based model is provided which is independent of

application-specific services and particular sensors. The modelling is done in such a way that existing services have been constructed in an ad-hoc manner; they have been designed for particular sensing systems, namely GPSs and RFID-tags. Furthermore, the main focus is on application-specific services, e.g., user navigation for visualizing locations on maps and the information relevant to the users current location. So far, such models have been studied and developed by many researchers but most of the existing models cannot be used in ubiquitous computing environments because these often need to be maintained in database systems, and ubiquitous computing environments must be managed at run-time in an ad-hoc manner and cannot always use static database systems. According to [63, 62, 64] the proposed model can be used in ubiquitous computing environments, and the framework provides the modelling of both physical entities and space among them. For example, when a person enters a place, personalized services should be provided from his or her portable terminal.

Leonhardt [33] proposed a taxonomy of location models and distinguished them into two major categories, namely geometric and symbolic models. In geometric models locations are represented as coordinates systems, whereas symbolic location models use the notion of place and rely on abstract symbols, namely naming the entities or labelling the locations. In this dissertation we use the notion of place to model location, and represent the structure of our system by a hierarchical space tree. The nodes represent the places, objects or computing devices, whereas the edges represent the containment relations between objects. Each node or object is represented by a named ambient, which may contain nested ambients.

2.4 Other Related Work

Graphs provide a simple and powerful framework for the modelling of many computer science problems [26, 27]. Many visual notations have been developed, including State Diagrams, Structural Analysis, Control Flow Graphs, Architectural Description Languages, and UML family of languages. By using these notations, models are created that can easily be seen as graphs and thus graph transformations are involved. Rules are defined for graphical interpretations, and implementation techniques are done accordingly.

The notion of bigraph has been introduced by Milner in [48] with the idea of presenting two independent structures on the same set of nodes. A bigraph is a mathematical structure consisting of a place graph and a link graph with common nodes. Place graph is limited to the tree structure while link graph can be hy-

pergraph. The theory of bigraphical reactive systems [45] is based on a graphical model of mobile computation that emphasizes both locality and connectivity. Object locations and connections can be represented simultaneously by combing the two graphs. A bigraphical reactive system in [46] is presented as a mobile computation model in which both location and connectivity are considered as first class entities. The model presents a fully graphical view that involves that how bigraphs compose (nesting of nodes that represents the locality), and location independent nodes connectivity. Process calculi and behavioural equivalences have led to an approach in bigraph theory somewhat different from the well-known tradition of graph rewriting [47].

Petri net [14] is a modelling approach used to model dynamic and distributed systems. Petri nets due to their graphical nature and expressiveness power could be used to model many of ubiquitous computing features, namely process mobility, dynamic structures, location, interactions between components, and context-awareness. Recently, some Petri nets models have been introduced that could be applied to model ubiquitous computing features, namely, Elementary Object Systems are considered in [74]. They represent process mobility by means of transitions.

In recent years several models based on Petri nets have been proposed for the modelling of ubiquitous computing features [13, 61, 75, 60]. In [13], a simple two-level model called Ubiquitous nets is presented for the modelling of ubiquitous systems. The model is based on traditional Petri nets, and is defined in terms of processors that supply services, processes that request services, and the mobility of processes. Similarly, [60] presents a dynamically amendable Petri net based model for ubiquitous computing. The model captures both locality and mobility of software components at different locations in a well synchronised manner. The devices and software components are modelled as coloured Petri nets, and local communication is formalised by means of token firing from one net to another. Authors, in [22], have modelled communication in ubiquitous computing systems based on Algebraic Higher Order Nets with Individual Token [32], a special type of Petri nets. The model has formalised both synchronous and asynchronous communication between components via shared channels and publish/subscribe scheme respectively. Furthermore, while considering context awareness as a key feature of ubiquitous computing, [25] discussed a number of Petri net based context modelling methodologies that could be applied in the setting of ubiquitous computing. These models are [59, 5, 24].

Frank Stajano in [67] addressed some of the risks associated with the ubiquitous computing settings. Security is crucial for most of the computer science applications.

However, a lot of research in recent years has been directed at solving the unique security problems raised by distributed systems. In ubiquitous computing setting computing is omnipresent and devices that do not look like computers are endowed with computing capabilities. However, the more difficult problem of protecting the security of the ubiquitous-computing (like, modelling securely and possible authentic interactions among components and with the environment) remains an open research topic.

Some modelling ideas have been discussed in [23] to create basic infrastructure of spatial model for location and context aware services in the setting of ubiquitous computing. The model describes the components, their functionalities, and mobility and interactions of the components with each other and with their environments. It follows the hierarchical tree pattern, containment relationships, and components migration. A comprehensive set of mathematical definitions is provided for moving from the notion of graph to the possible tree structure. *Aura* is defined for each component of the spatial model showing ranges of the corresponding entities, as inspired by [63], and possible interactions are based on the Aura of the components. Different notations are used to show objects and their relationships. Several transformation rules are defined and conditions to the rules are given in mathematical way. Based on these rules graph transformation is shown by describing different scenarios against ubiquitous computing health setting (proposed case-study).

Chapter 3

Towards a Calculus of Mobility

In this chapter we review the calculus of Mobile Ambients, MA for short, [11] including its syntax and reduction semantics. We shall reuse only the mobility part of MA and hence call the calculus, a Calculus of Mobility (CM). We present a new operational semantics for CM and prove that the semantics is sound with respect to the standard reduction semantics. We then analyse the completeness of our operational semantics. Completeness ensures that for every valid τ-transition of a CM term there is a valid reduction of the term, and the targets of the τ-transitions and the reductions are the same. We show with the help of some examples that the operational semantics is not complete in general. These examples help us to develop a new complete operational semantics in Chapter 4.

The calculus of Mobile Ambients has been introduced by Cardelli and Gordon, [11] as a concurrent process calculus where the notion of ambient is used to model various structures that are distributed and mobile. MA describes two aspects of mobility, namely (a) mobile computing that concerns physical mobility of computing devices, and (b) mobile computation that concerns with logical mobility where code moves between devices. Ambients are named terms of the form $n[P]$ where n is a name and P a process. Ambients may enter or exit named ambients by their *in* n and *out* n capabilities. The ambient's *open* capability dissolves its boundary so that the communication may take place. Recently, a number of variants of MA have been introduced, most notably Boxed Ambients [6], Safe Boxed Ambients [41] and Channel Ambients [53]. These calculi inherit their primitives from MA with some modifications. CM inherits mobility primitives, namely *in* and *out* capabilities from MA. We drop *open* capability as CM proposes a new form of global communication in Chapter 5.

Our labelled transition semantics is inspired by that in [36, 37], where ambients co-capabilities are used which are exercised by the target computation space and

$Names:$	$m, n, k... \in \mathcal{N}$					
$Processes:$	$P, Q ::=$	0	$\mid$	$C.P$	$\mid$	$m[P]$
	$\mid$	$P \mid Q$	$\mid$	$(\nu m)P$		
$Capabilities:$	$C ::=$	$in\ n$	$\mid$	$out\ n$		

Table 3.1: Syntax of CM

they help in controlling ambients mobility across the boundaries. Also [36, 37] use *concretions* of the form $\nu\tilde{m}\langle P\rangle Q$ in their operational semantics, where P is the migrating agent, Q the residual code and $\tilde{m}$ is the set of shared names. Our semantics does not have the co-capabilities, hence preserving the standard MA semantics, and as the first attempt we have developed our transition rules without using concretions; (see Section 4.1 and [37] for the definition of concretions).

The chapter is organised as follows: We introduce CM in Section 3.1 where its syntax, structural congruence and reduction semantics are given. Section 3.2 includes the structural operational semantics for CM. In Section 3.3 we show that our operational semantics is sound w.r.t reduction semantics. We analyse completeness of the semantics in Section 3.4, and then conclude the chapter.

3.1 The Syntax of Calculus of Mobility

The syntax of CM consists of ambient names, for example, $m, n, k \in \mathcal{N}$, processes P, Q, and capabilities C as presented in Table 3.1. The syntax is the same as in [11]. The deadlock agent 0 is the agent that does nothing. In $C.P$, the process P cannot start execution until the prefix capability C is performed. The term $m[P]$ represents an ambient, where m is the ambient name and P an executing process. Parallel composition is given in terms of a binary operator, $P \mid Q$. An ambient restriction $(\nu m)P$ executes process P with a private ambient named m. The capability $in\ n$ tries to move the surrounding ambients into a sibling ambient with the name n, whereas $out\ n$ moves the surrounding ambient out of its parent ambient with the name n.

Definition 3.1. We denote the set of all names occurring free in process P by $fn(P)$. We define free names for CM in Table 3.2.

$fn(0)$	$\stackrel{def}{=}$	ϕ
$fn(C.P)$	$\stackrel{def}{=}$	$fn(C) \cup fn(P)$
$fn(m[P])$	$\stackrel{def}{=}$	$\{m\} \cup fn(P)$
$fn(P \mid Q)$	$\stackrel{def}{=}$	$fn(P) \cup fn(Q)$
$fn(\nu m(P))$	$\stackrel{def}{=}$	$fn(P) - \{m\}$
$fn(in\ n)$	$\stackrel{def}{=}$	$\{n\}$
$fn(out\ n)$	$\stackrel{def}{=}$	$\{n\}$

Table 3.2: Free names

3.1.1 Structural Congruence

Structural Congruence relation, denoted as $\equiv$, rearranges the term P to yield Q. Relation $\equiv$ between the two agents represents that they are equal.

Definition 3.2. *Structural Congruence*, $\equiv$, over CM processes is the smallest congruence relation that satisfies the axioms:

$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$P \mid 0 \equiv P$	(Struct Zero Par)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Struct Res Res)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin fn(P)$	(Struct Res Par)
$(\nu n)(m[P]) \equiv m[(\nu n)P]$ if $n \neq m$	(Struct Res Amb)

By this definition we get the axioms and the rules for $\equiv$ in Table 3.3. We briefly explain them as follows:

(1-3) An agent is structurally congruent to itself. The two structurally congruent agents possess the equivalence property of symmetry and transitivity.

(4-7) Congruence equations.

(8-9) Agents P and Q are structurally congruent up-to re-ordering of parallel compositions. The parallel composition of the two agents are commutative and associative.

(10) The null agent can be composed in parallel with a given agent P.

$P \equiv P$	(Struct Refl)	(1)
$P \equiv Q \Rightarrow Q \equiv P$	(Struct Symm)	(2)
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	(Struct Trans)	(3)
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Struct Res)	(4)
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	(Struct Par)	(5)
$P \equiv Q \Rightarrow n[P] \equiv n[Q]$	(Struct Amb)	(6)
$P \equiv Q \Rightarrow C.P \equiv C.Q$	(Struct Capability)	(7)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)	(8)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)	(9)
$P \mid 0 \equiv P$	(Struct Zero Par)	(10)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Struct Res Res)	(11)
$(\nu n)0 \equiv 0$	(Struct Zero Res)	(12)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin fn(P)$	(Struct Res Par)	(13)
$(\nu n)(m[P]) \equiv m[(\nu n)P]$ if $n \neq m$	(Struct Res Amb)	(14)

Table 3.3: Structural congruence

(11-12) The reordering of restricted names does not effect the agents congruence. The successive restricted names can be re-ordered. The null agent with restricted name is equal to the null agent.

(13-14) The scope of a name n restricted over two parallel agents P and Q can be lowered to the agent Q providing that n is not a free name in P. The scope of a name n restricted inside an ambient m can be extended outside the ambient if the ambient name is different from the restricted name.

3.1.2 Reduction Semantics for CM

The reduction $P \rightarrow Q$ says that P may evolve in one computation step to yield Q.

Definition 3.3. Reduction relation, $\rightarrow$, over CM processes is the smallest relation that satisfies the rules and axioms in Table 3.4.

$$m[in\ n.P \mid Q] \mid n[R] \rightarrow n[m[P \mid Q] \mid R] \qquad \text{(Red In)}$$

$$n[m[out\ n.P \mid Q] \mid R] \rightarrow m[P \mid Q] \mid n[R] \qquad \text{(Red Out)}$$

$$P \equiv Q,\ Q \rightarrow Q',\ Q' \equiv P' \Rightarrow P \rightarrow P' \qquad \text{(Red } \equiv\text{)}$$

$$P \rightarrow Q \ \Rightarrow\ (\nu n)P \rightarrow (\nu n)Q \qquad \text{(Red Res)}$$

$$P \rightarrow Q \ \Rightarrow\ n[P] \rightarrow n[Q] \qquad \text{(Red Amb)}$$

$$P \rightarrow Q \ \Rightarrow\ P \mid R \rightarrow Q \mid R \qquad \text{(Red Par)}$$

Table 3.4: Reduction rules and axioms

We now give some examples to show reductions of simple ambients. Initially, we assume that our agent is of the form,

$$m[in\ n.P] \mid n[R] \mid S,\ \text{for some } P,\ R \text{ and } S.$$

In this setting the ambient m intends to enter a sibling ambient with the name n. The successful reduction transforms m the sibling of ambient n into a child of n. By Red In in Table 3.4, the term $m[in\ n.P] \mid n[R]$ reduces to $n[m[P] \mid R]$. Now, by Red Par we get

$$m[in\ n.P] \mid n[R] \mid S \rightarrow n[m[P] \mid R] \mid S$$

Hence the above reductions transform m the sibling of n into a child of n, while the process S is still executing in parallel.

Similarly, to exercise the *out* capability we consider a system

$$n[m[out\ n.P] \mid R] \mid S$$

In this setting, the prefix $out\,n$ instructs the ambient m to exit its parent ambient n. Agent R is running in parallel with ambient m, and agent S is running in parallel with the ambient n. After a successful reduction m becomes a sibling of n. By Red Out, the term $n[m[out\ n.P] \mid R]$ reduces to $m[P] \mid n[R]$. Thus, by Red Par we get,

$$n[m[out\ n.P] \mid R] \mid S \rightarrow m[P] \mid n[R] \mid S$$

The above given relation shows that the code inside m is also moved out of the surrounding ambient n. Now the three agents m, n and S are siblings and exist at the same level.

Finally, we consider a system

$$n[(\nu k)(m[in\ k.0 \mid out\ n.0]) \mid P] \mid Q.$$

In this setting, we restrict the scope of private name k to agent m, and the process P executes in parallel with m inside n. The two capabilities inside the ambient m enable m to either leave parent ambient n or enter sibling ambient k. Since k is restricted we assume m leaves n by *out* n and we get

$$\begin{aligned} & n[(\nu k)(m[in\ k.0 \mid out\ n.0]) \mid P] \mid Q \\ & \quad \equiv n[\nu k(m[in\ k.0 \mid out\ n.0] \mid P)] \mid Q, \qquad \textit{if } k \notin \textit{fn}(P) \qquad \text{Struct Res Par} \\ & \quad \equiv \nu k(n[m[in\ k.0 \mid out\ n.0] \mid P]) \mid Q, \qquad \textit{if } k \neq n \qquad \text{Struct Res Amb} \end{aligned}$$

Since by Red Out we get

$$n[m[in\ k.0 \mid out\ n.0] \mid P] \longrightarrow m[in\ k.0 \mid 0] \mid n[P]$$

thus, by Red Res, we obtain

$$\nu k(n[m[in\ k.0 \mid out\ n.0] \mid P]) \longrightarrow \nu k(m[in\ k.0 \mid 0] \mid n[P]).$$

Finally, by Red Par, we obtain

$$\nu k(n[m[in\ k.0 \mid out\ n.0] \mid P]) \mid Q \longrightarrow \nu k(m[in\ k.0 \mid 0] \mid n[P]) \mid Q.$$

Here, after reductions, the scope of k is extended to outside m if $k \neq n$ and $k \notin \textit{fn}(P)$.

3.2 Labelled Transition System Semantics for CM

In this section we present an operational semantics for CM.

Definition 3.4. A *Labelled Transition System (LTS)* is a tuple $(S, L, \{\xrightarrow{\ell} : \ \ell \in L\})$ which consists of

- a set S of *states* (or *nodes*),
- a set L of (transition) *labels*, and
- a family of *transition* relations: $\xrightarrow{\ell} \subseteq S \times L \times S$, for $\ell \in L$.

The LTS for CM is as follows: the set of processes of CM is the set of states, the set of labels ℓ as in Table 3.5 is the set of transition labels, and the transition

Ambient Prefixes :	$\mu ::=$	$in\, n$	$\mid$	$out\, n$		
Labels :	$\ell ::=$	μ	$\mid$	τ		
	$\mid$	$enter1\, n$	$\mid$	$move1\, n$	$\mid$	$exit1\, n$
	$\mid$	$enter2\, n$	$\mid$	$move2\, n$	$\mid$	$exit2\, n$

Table 3.5: Prefixes and labels for CM

relations $\xrightarrow{\ell}$ are defined by Plotkin's SOS [55] rules in Tables 3.6 and 3.7. In our semantics $P \xrightarrow{\tau} Q$ represents mobility of ambients by means of their *in* n and *out* n capabilities. In order to model mobility by τ-transitions additional labels are used in Table 3.5. These labels are: $enter1\ n$, $enter2\ n$, $exit1\ n$, $exit2\ n$, $move1\ n$ and $move2\ n$. Our labelled transition semantics is inspired by that in [36, 37, 40]. The main difference is that we do not use the co-capabilities, hence preserving the standard Mobile Ambients semantics. Also, in [36, 37, 40] concretions of the form $\nu\tilde{m}\langle P\rangle Q$ are used in the LTS, whereas we have developed our transition rules without using concretions to keep them simple.

Now we explain the transition rules in Tables 3.6 and 3.7. Each transition is of the form $P \xrightarrow{\ell} P'$ where a process P evolves to a process P' by performing an action labelled ℓ.

Act - $\mu.P$ performs the action μ, and then executes as process P. The action represents ambients' migration, namely $in\, n$ and $out\, n$. These prefixes enables ambients to move around, for example, the prefix $in\, n$ induces a capability in ambient to migrate into n.

Enter1 - An ambient $m[P]$ has an action $enter1\ n$ if the process P exhibits a capability $in\ n$. This rule allows the ambient m to enter the ambient with the name n. After the transition the moving ambient $m[P]$ evolves to a new state $m[P']$. The resultant state $m[P']$ is surrounded by the parent ambient with the name n. Here, the term $m[P']$ is the migrated agent.

Enter2 - The ambient $m[P]$ has the action $enter2\ n$, if the process P evolves to P' after exercising the capability $in\, n$. Here, the term 0 represents the residual code. When ambient $m[P]$ migrates into ambient n, there is nothing left behind.

Co-Enter1 - The ambient n has a transition action $move1\, n$, and evolves to the state P from $n[P]$ without having any hypothesis. The process P represents

$$\text{(Act)}\ \frac{}{\mu.P \xrightarrow{\mu} P}$$

$$\text{(Enter1)}\ \frac{P \xrightarrow{in\,n} P'}{m[P] \xrightarrow{enter1\,n} m[P']} \qquad \text{(Enter2)}\ \frac{P \xrightarrow{in\,n} P'}{m[P] \xrightarrow{enter2\,n} 0}$$

$$\text{(Co-Enter1)}\ \frac{}{n[P] \xrightarrow{move1\,n} P} \qquad \text{(Co-Enter2)}\ \frac{}{n[P] \xrightarrow{move2\,n} 0}$$

$$\text{(τ-In)}\ \frac{Q \xrightarrow{move1\,n} (\nu\tilde{q})Q' \qquad Q \xrightarrow{move2\,n} (\nu\tilde{q})Q'' \qquad P \xrightarrow{enter1\,n} (\nu\tilde{p})P' \qquad P \xrightarrow{enter2\,n} (\nu\tilde{p})P''}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p})(\nu\tilde{q})(n[P' \mid Q'] \mid P'' \mid Q'')}\ (*)$$

$$\text{(Exit1)}\ \frac{P \xrightarrow{out\,n} P'}{m[P] \xrightarrow{exit1\,n} 0} \qquad \text{(Exit2)}\ \frac{P \xrightarrow{out\,n} P'}{m[P] \xrightarrow{exit2\,n} m[P']}$$

$$\text{(τ-Out)}\ \frac{P \xrightarrow{exit1\,n} (\nu\tilde{m})P' \quad P \xrightarrow{exit2\,n} (\nu\tilde{m})P''}{n[P] \xrightarrow{\tau} (\nu\tilde{m})(n[P'] \mid P'')}\ (**)$$

$(*)(fn(P') \cup fn(P'')) \cap \tilde{q} = (fn(Q') \cup fn(Q'')) \cap \tilde{p} = \phi$

$(**)(fn(P') \cup fn(P'')) \cap \tilde{m} = \phi$

Table 3.6: Transition rules for CM capabilities

what must stay inside the ambient n. By this rule, the ambient n allows other ambients to migrate in.

Co-Enter2 - An ambient n allows other ambient to move in, and involves no hypothesis. The 0 at the right side of the conclusion represents that there exists nothing outside the ambient n.

τ-In - Agent P wishes to enter agent Q by performing $in\,n$, and agent Q contains an ambient n. We chose $\tilde{p}$ and $\tilde{q}$ as sets of ambient names that are private in processes P and Q respectively. If there are no private names in the given agents, then these sets are assumed to be empty. We intuitively split the process P into two parts, one is P' and other is P''. The process P' represents the agent that moves in, whereas the process P'' represents the residual agent. Here $\nu\tilde{p}$ represents that after emigration the name $\tilde{p}$ is private in the two sub-agents P' and P''. Similarly, We split the process Q into two parts, one is Q' and other is Q''. The process Q' is the receiving agent, whereas the process Q'' represents what is left outside once Q receives

$$\text{(Res-Amb)}\ \frac{P \xrightarrow{\ell} P'}{(\nu u)P \xrightarrow{\ell} (\nu u)P'}\ (u \notin fn(\ell)) \qquad \text{(Par)}\ \frac{P \xrightarrow{\ell} P'}{P \mid Q \xrightarrow{\ell} P' \mid Q}\ (*)$$

$$\text{(Par-Enter1)}\ \frac{P \xrightarrow{enter1\, n} P'}{P \mid Q \xrightarrow{enter1\, n} P'} \qquad \text{(Par-Enter2)}\ \frac{P \xrightarrow{enter2\, n} P'}{P \mid Q \xrightarrow{enter2\, n} P' \mid Q}$$

$$\text{(Par-Move1)}\ \frac{P \xrightarrow{move1\, n} P'}{P \mid Q \xrightarrow{move1\, n} P'} \qquad \text{(Par-Move2)}\ \frac{P \xrightarrow{move2\, n} P'}{P \mid Q \xrightarrow{move2\, n} P' \mid Q}$$

$$\text{(Par-Exit1)}\ \frac{P \xrightarrow{exit1\, n} (\nu\tilde{m})P'}{P \mid Q \xrightarrow{exit1\, n} (\nu\tilde{m})(P' \mid Q)} \qquad \text{(Par-Exit2)}\ \frac{P \xrightarrow{exit2\, n} (\nu\tilde{m})P'}{P \mid Q \xrightarrow{exit2\, n} (\nu\tilde{m})P'}$$

$$\text{(Struct)}\ \frac{P \equiv Q \quad Q \xrightarrow{\ell} Q' \quad Q' \equiv P'}{P \xrightarrow{\ell} P'} \qquad (\tau\text{-Amb})\ \frac{P \xrightarrow{\tau} P'}{n[P] \xrightarrow{\tau} n[P']}$$

$(*)\quad \ell \notin \{exit1, exit2, enter1, enter2, move1, move2\}$

Table 3.7: Transition rules for other operators of CM

the sibling ambient. Here, $\nu\tilde{q}$ represents that after emigration the names $\tilde{q}$ are private in the two sub-agents Q' and Q''.

More specifically, the agent $P \mid Q$ performs a τ transition, if the process P performs the actions $enter1\, n$ and $enter2\ n$ and transforms to the states P' and P'' respectively, and the process Q transforms to the states Q' and Q'' by exercising $move1\ n$ and $move2\, n$ actions respectively.

Exit1 - The ambient $m[P]$ has the action $exit1\ n$ if the process P evolves to P' after exercising the capability $out n$. Here, the term 0 represents the residual code, when ambient $m[P]$ migrates out of an ambient n, there is nothing left behind.

Exit2 - An ambient $m[P]$ has an action $exit2\ n$ if the process P contains a capability $out\, n$. This rule allows the ambient m to leave the ambient with the name n. After the transition the moving ambient $m[P]$ evolves to a new state $m[P']$. We assume that the resultant state $m[P']$ becomes the sibling of an ambient with the name n. The term $m[P']$ represents that what must move out of an ambient n.

τ-Out - An agent $n[P]$ performs a τ transition, if the process P performs the actions $exit1n$ and $exit2\ n$ to evolve the states P' and P'' respectively. More

generally, the agent P wishes to leave the agent n by performing $out\, n$. We chose $\tilde{m}$ as the set of ambient names that is private in process P. If there are no private names in P, then $\tilde{m}$ is assumed to be empty. Furthermore, we split the process P into P' and P''. The process P' represents the residual agent after exercising the *out* capability, whereas P'' represents the agent that moves out. Here, $\nu\tilde{m}$ represents that after emigration the names $\tilde{m}$ are private in the two sub-agents P' and P''.

Res-Amb - $(\nu u)P$ represents that the name u is private in P. This agent can perform a transition if the agent P evolves to P' by performing the same transition providing that $u \notin fn(\ell)$.

We now show some examples that illustrate how to use the SOS rules discussed above. Initially we consider a system

$$m[in\, n.P] \mid Q$$

The ambient m has the capability to enter an ambient with the name n, where ambient n must exist in parallel with the moving ambient. So to exercise this capability agent Q must have the form $n[Q'] \mid Q''$, for some processes R and S. If there is no such sibling ambient n, then m cannot move. To exercise this capability of m, τ-In in Table 3.6 is used. We assume that there are no private names in agent P, Q, R and S. The resulting transition is as follows:

$$m[in\, n.P] \mid n[Q'] \mid Q'' \xrightarrow{\tau} n[m[P'] \mid Q'] \mid Q''$$

The inference tree that proves the validity of this transition is given in Figure 3.1.

Next, we consider a situation where agents share private names. For example, we assume

$$(\nu k)(m[in\, n.0 \mid in\, k.0] \mid k[P]) \mid n[R]$$

where ambients m and k execute in parallel and they share a private name k. The ambient m has the capabilities to either enter k or n. We consider the ability of m to enter n which is running in parallel with m, where the two agents share no private names. To exercise this capability of agent m, we included constructs $(\nu\tilde{p})$ and $(\nu\tilde{q})$ in τ-In rule. These constructs maintain privacy and sharing of the restricted names in the agents after they move around. The application of the τ-In rule specifies how this capability of m is achieved. The resulting transition is as follows, where

$$\text{(Act)}\ \frac{}{in\ n.P \xrightarrow{in\ n} P} \qquad \text{(Act)}\ \frac{}{in\ n.P \xrightarrow{in\ n} P}$$

$$\text{(Enter1)}\ \frac{}{m[in\ n.P] \xrightarrow{enter1\ n} m[P]} \qquad \text{(Enter2)}\ \frac{}{m[in\ n.P] \xrightarrow{enter2\ n} 0} \qquad \text{(Co-Enter1)}\ \frac{}{n[Q'] \xrightarrow{move1\ n} Q'} \qquad \text{(Co-Enter2)}\ \frac{}{n[Q'] \xrightarrow{move2\ n} 0}$$

$$\text{(τ-In)}\ \frac{}{m[in\ n.P] \mid n[Q'] \xrightarrow{\tau} n[m[P] \mid Q'] \mid 0}$$

$$\text{(Par)}\ \frac{}{m[in\ n.P] \mid n[Q'] \mid Q'' \xrightarrow{\tau} n[m[P] \mid Q'] \mid 0 \mid Q'' \equiv n[m[P] \mid Q'] \mid Q''}$$

$$\text{(Struct)}\ \frac{}{m[in\ n.P] \mid n[Q'] \mid Q'' \xrightarrow{\tau} n[m[P] \mid Q'] \mid Q''}$$

Figure 3.1: Inference tree for enter capability

$k \notin fn(R)$:

$$(\nu k)(m[in\ n.0 \mid in\ k.0] \mid k[P]) \mid n[R] \xrightarrow{\tau} (\nu k)(\nu \tilde{q})(n[m[in\ k.0] \mid R] \mid k[P])$$

The inference tree that proves the validity of this transition is given in Figure 3.2. If there exists a set of ambient names private in an agent, we avoid using another level of restriction with a name that is already private in the agent, as given in the inference tree for ambients' enter capability using restriction in Figure 3.2. We show this in Lemma 3.1 as follows:

Lemma 3.1. $(\nu k)(\nu k(P)) \equiv \nu k(P)$

Proof.

$(\nu k)(\nu k(P))$	$\equiv (\nu k)(\nu k(P) \mid 0)$		Struct Zero Par
	$\equiv \nu k(P) \mid \nu k(0)$	$k \notin fn((\nu k)P)$	Struct Res Par
	$\equiv \nu k(P) \mid 0$		Struct Zero Res
	$\equiv \nu k(P)$		Struct Zero Par

□

Next, to exercise the *out* capability we propose a set of exit rules in Table 3.6. For the application of exit rules we consider a system,

$$m[out\ n.P \mid Q]$$

Here, the prefix *out* n instructs the ambient m to exit its parent ambient with the name n. P and Q are the two agents running in parallel inside m. To exercise this capability the ambient m must be a child of the ambient n. If the parent named n does not exist then the ambient m cannot move out. Furthermore, we assume the agents R, S run in parallel with the ambients m and n respectively, so our system is of the form:

$$n[m[out\ n.P \mid Q] \mid R] \mid S$$

In this example, we assume that there are no private names in P, Q, R and S. The exit transition transforms m the child of n to a sibling of n. When this capability is exercised, the processes inside m will also move out of the surrounding ambient n. Now the three agents m, n and S exist at the same level. The application of τ-Out specifies how this capability of m is achieved. The resulting transition is:

$$n[m[out\ n.P \mid Q] \mid R] \mid S \xrightarrow{\tau} m[P \mid Q] \mid n[R] \mid S$$

The inference tree that proves the validity of the above given transition is in Fig-

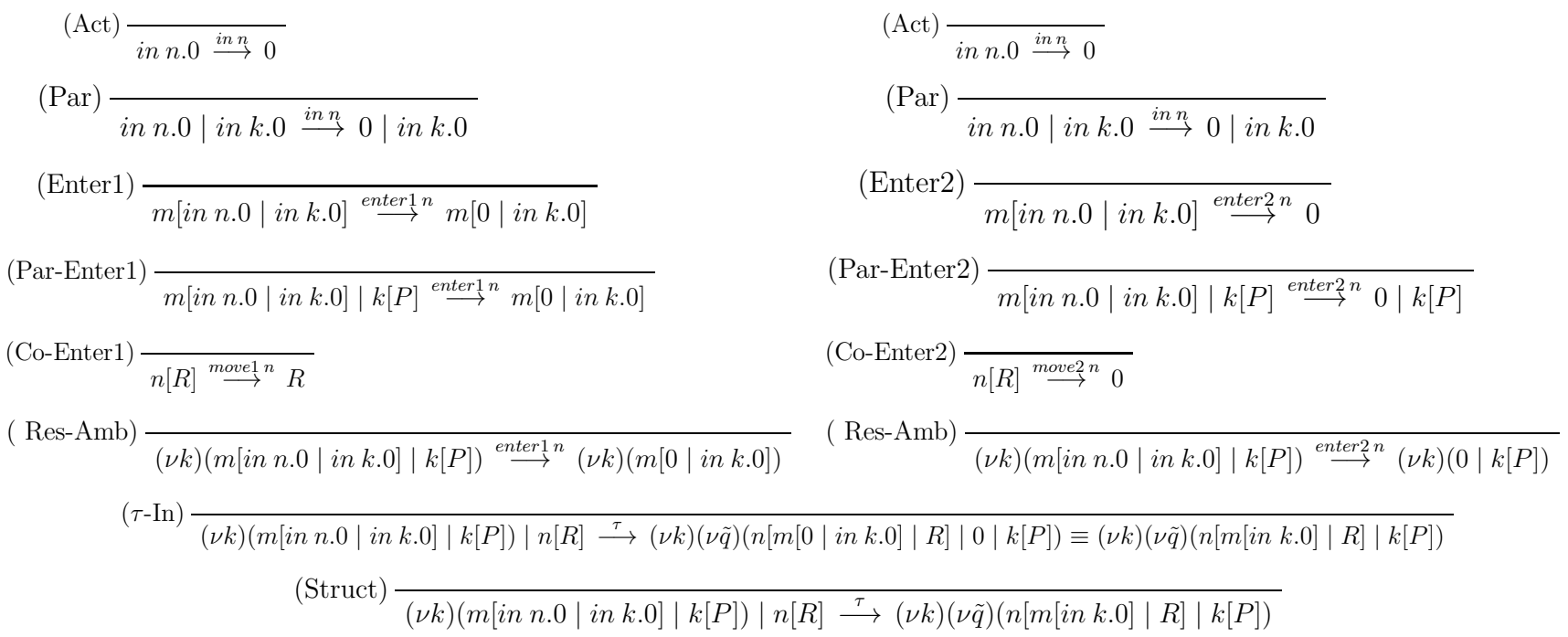


Figure 3.2: Inference tree for enter capability using restriction

ure 3.3.

$$\text{(Act)}\ \frac{}{out\ n.P \xrightarrow{out\ n} P} \qquad \text{(Act)}\ \frac{}{out\ n.P \xrightarrow{out\ n} P}$$

$$\text{(Par)}\ \frac{}{out\ n.P \mid Q \xrightarrow{out\ n} P \mid Q} \qquad \text{(Par)}\ \frac{}{out\ n.P \mid Q \xrightarrow{out\ n} P \mid Q}$$

$$\text{(Exit1)}\ \frac{}{m[out\ n.P \mid Q] \xrightarrow{exit1\ n} 0} \qquad \text{(Exit2)}\ \frac{}{m[out\ n.P \mid Q] \xrightarrow{exit2\ n} m[P \mid Q]}$$

$$\text{(Par-Exit1)}\ \frac{}{m[out\ n.P \mid Q] \mid R \xrightarrow{exit1\ n} 0 \mid R} \qquad \text{(Par-Exit2)}\ \frac{}{m[out\ n.P \mid Q] \mid R \xrightarrow{exit2\ n} m[P \mid Q]}$$

$$\text{(τ-Out)}\ \frac{}{n[m[out\ n.P \mid Q] \mid R] \xrightarrow{\tau} n[0 \mid R] \mid m[P \mid Q]}$$

$$\text{(Par)}\ \frac{}{n[m[out\ n.P \mid Q] \mid R] \mid S \xrightarrow{\tau} n[0 \mid R] \mid m[P \mid Q] \mid S \quad (n[0 \mid R] \mid m[P \mid Q] \mid S \equiv n[R] \mid m[P \mid Q] \mid S)}$$

$$\text{(Struct)}\ \frac{}{n[m[out\ n.P \mid Q] \mid R] \mid S \xrightarrow{\tau} n[R] \mid m[P \mid Q] \mid S}$$

Figure 3.3: Inference tree for exit capability

Similarly, in order to exercise the *out* capability we consider a setting where ambients share private names:

$$(\nu k)(n[m[in\ k.0 \mid out\ n.0] \mid k[P]])$$

In this example the name k is private and shared in the agents n, m and k. In principle, there could be other shared and private names in agent P, but we assume that there is no shared name that is private in the agent P. There are two capabilities executing inside m. These capabilities instruct m to either leave parent ambient n, or enter sibling ambient k. We assume that exit capability of m occurs. We added $\nu\tilde{m}$ to our τ-Out rule as discussed earlier in this section. The rule τ-Out specifies how this capability of the agent is achieved. The resulting transition is:

$$(\nu k)(n[m[in\ k.0 \mid out\ n.0] \mid k[P]]) \xrightarrow{\tau} (\nu k)(n[k[P]] \mid m[in\ k.0])$$

Furthermore, we consider a system,

$$n[(\nu k)(m[in\ k.0 \mid out\ n.0]) \mid P] \mid Q$$

In this example we restrict the scope of private name k to agent m. The ambient m and an agent P executes in parallel inside ambient n, but the name k is private

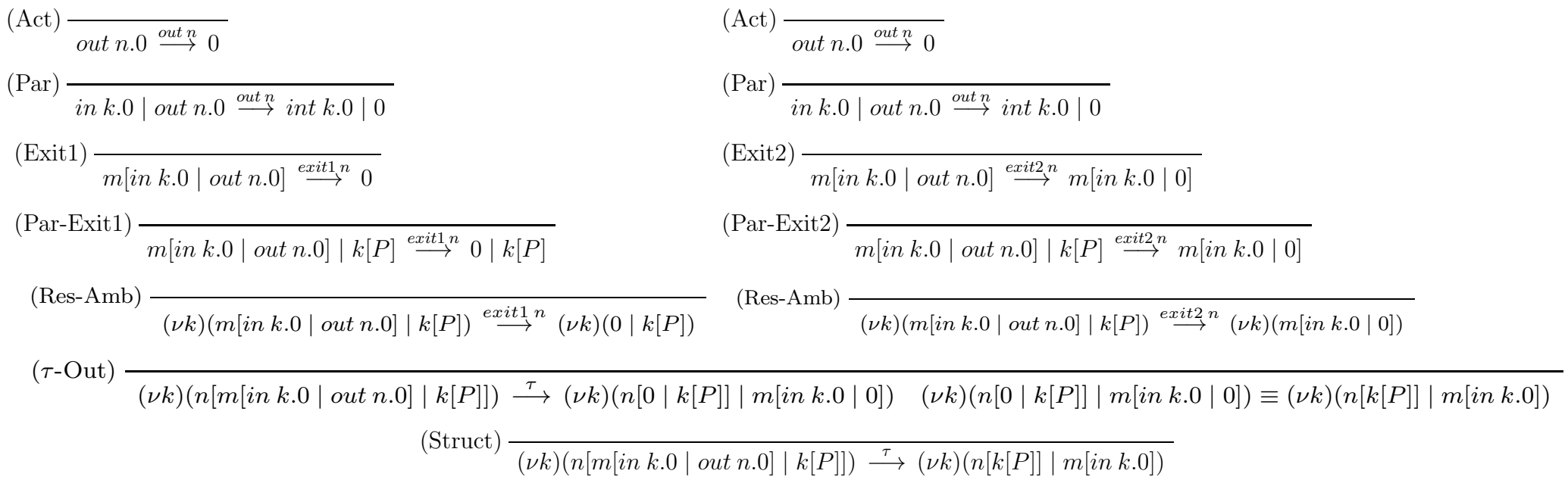


Figure 3.4: Inference tree for exit capability using restriction

$$\text{(Act)}\ \frac{}{out\ n.0 \xrightarrow{out\ n} 0} \qquad \text{(Act)}\ \frac{}{out\ n.0 \xrightarrow{out\ n} 0}$$

$$\text{(Par)}\ \frac{}{in\ k.0 \mid out\ n.0 \xrightarrow{out\ n} int\ k.0 \mid 0} \qquad \text{(Par)}\ \frac{}{in\ k.0 \mid out\ n.0 \xrightarrow{out\ n} in\ k.0 \mid 0}$$

$$\text{(Exit1)}\ \frac{}{m[in\ k.0 \mid out\ n.0] \xrightarrow{exit1\ n} 0} \qquad \text{(Exit2)}\ \frac{}{m[in\ k.0 \mid out\ n.0] \xrightarrow{exit2\ n} m[in\ k.0 \mid 0]}$$

$$\text{(Res-Amb)}\ \frac{}{\nu k(m[in\ k.0 \mid out\ n.0]) \xrightarrow{exit1\ n} \nu k(0)} \qquad \text{(Res-Amb)}\ \frac{}{\nu k(m[in\ k.0 \mid out\ n.0]) \xrightarrow{exit2\ n} \nu k(m[in\ k.0 \mid 0])}$$

$$\text{(Par-Exit1)}\ \frac{}{\nu k(m[in\ k.0 \mid out\ n.0]) \mid P \xrightarrow{exit1\ n} \nu k((0) \mid P)} \qquad \text{(Par-Exit2)}\ \frac{}{\nu k(m[in\ k.0 \mid out\ n.0]) \mid P \xrightarrow{exit2\ n} \nu k(m[in\ k.0 \mid 0])}$$

$$\text{(τ-Out)}\ \frac{}{n[\nu k(m[in\ k.0 \mid out\ n.0]) \mid P] \xrightarrow{\tau} \nu k(n[0 \mid P] \mid m[in\ k.0 \mid 0])}$$

$$\text{(Par)}\ \frac{}{n[\nu k(m[in\ k.0 \mid out\ n.0]) \mid P] \mid Q \xrightarrow{\tau} (\nu k(n[0 \mid P] \mid m[in\ k.0 \mid 0]) \equiv \nu k(n[P] \mid m[in\ k.0]) \mid Q)}$$

$$\text{(Struct)}\ \frac{}{n[\nu k(m[in\ k.0 \mid out\ n.0]) \mid P] \mid Q \xrightarrow{\tau} \nu k(n[\nu k(P)] \mid \nu k(m[in\ k.0])) \mid Q}$$

Figure 3.5: Inference tree for exit capability using restriction example 2

only in agent m. We assume that there are no private names in agents P and Q, and $k \notin fn(P)$. The application of τ-Out rule specifies how the out capability of the agent is achieved. The resulting transition is:

$$n[(\nu k)(m[in\ k.0 \mid out\ n.0]) \mid P] \mid Q \xrightarrow{\tau} (\nu k)(n[P] \mid m[in\ k.0]) \mid Q$$

The τ-transition shows that m which is a child of n, becomes after the transition a sibling of n. The scope of k is extended to also include P since $k \notin fn(P)$. If $k \in fn(P)$ then the above scope extrusion becomes invalid.

3.3 Soundness of Operational Semantics

Soundness ensures that for every reduction $P \to P'$ of a CM term P, for some P', there is a valid τ-transition of P, namely $P \xrightarrow{\tau} Q$, for some Q in CM, and the target of the τ-transition is congruent to the target of the reduction, namely $Q \equiv P'$.

Theorem 3.1. $\forall P, R \in CM.\ P \to R \Longrightarrow \exists\, Q \in CM.\ P \xrightarrow{\tau} Q \equiv R.$

Proof. We prove Theorem 3.1 by using induction on the structure of P.

1. Base case: (Constant)
   We show that our statement holds when we choose the simplest term of CM, namely the deadlocked agent 0:

   $$0 \to R \Longrightarrow 0 \xrightarrow{\tau} R$$

   There is no rule defined to show the reduction of 0, so $0 \to R$ is false, and hence, the implication above is true.

2. Induction Hypothesis:
   We assume that our theorem holds for all the sub-processes P' of P, such that if $P' \to R \Longrightarrow P' \xrightarrow{\tau} R$, for all R.

3. Induction Step:

   (a) $P = (\nu m)P'$, an ambient name m private in P'.
   In this case we show

   $$(\nu m)P' \to R \Longrightarrow (\nu m)P' \xrightarrow{\tau} R \tag{3.1}$$

   We assume $(\nu m)P' \to R$.

The only reduction rule that can be used to derive a reduction relation of $(\nu m)P'$ is Red-Res in Table 3.4.

Since the term $(\nu m)P'$ reduces to some process R, by the reduction rule Red-Res, so we deduce that the reduction $P' \rightarrow Q$ is also valid for some Q, such that $(\nu m)Q = R$. Since $P' \rightarrow Q$ is valid, so by the inductive hypothesis we get $P' \xrightarrow{\tau} Q$.

Since $P' \xrightarrow{\tau} Q$ is a valid transition, by transition rule Res-Amb in Table 3.6, we deduce that the transition $(\nu m)P' \xrightarrow{\tau} (\nu m)Q$ is valid (as $m \notin fn(\tau)$).

Now using the structural congruence rule Struct in Table 3.7, we obtain $(\nu m)P' \xrightarrow{\tau} (\nu m)Q$ and $(\nu m)Q \equiv R$, as equality is the subset of our congruence ($=\subset\equiv$). Hence, the conclusion of the rule is valid, that is: $(\nu m)P' \xrightarrow{\tau} R$ as required.

(b) $P = n[P']$, an ambient with name n for some P'.

In this case we show

$$n[P'] \longrightarrow R \Longrightarrow n[P'] \xrightarrow{\tau} R \tag{3.2}$$

To prove statement 3.2, we assume $n[P'] \rightarrow R$, for some R. There are two reduction rules Red-Amb and Red-Out in Table 3.4, that can be used to derive a reduction of $n[P']$. So, to apply each rule separately, we divide this case into two sub-cases.

i. Red-Amb:
Since using Red-Amb, $n[P'] \rightarrow R$ is a valid reduction, so $P' \rightarrow Q$ is also valid for some Q. By this rule the agent R is of the form $n[Q]$, such as, $R = n[Q]$. As $P' \rightarrow Q$ is valid, so by inductive hypothesis we get $P' \xrightarrow{\tau} Q$.
Since $P' \xrightarrow{\tau} Q$ is a valid transition, so by transition rule τ-Amb, we deduce that the conclusion of the rule is also valid, that is $n[P'] \xrightarrow{\tau} n[Q]$.
Now by Struct, $n[P'] \xrightarrow{\tau} n[Q]$, and $n[Q] \equiv R$. Hence, we obtain $n[P'] \xrightarrow{\tau} R$ as required.

ii. Red-Out:
By using the rule Red-Out, we deduce that P is of the form $m[out\ n.P_1 \mid P_2] \mid Q$ for some P_1, P_2 and Q. Hence, the reduction $n[m[out\ n.P_1 \mid P_2] \mid Q] \rightarrow R$ is valid by Red-Out, where $R = m[P_1 \mid P_2] \mid n[Q]$.
We derive the $\xrightarrow{\tau}$ of agent $n[m[out\ n.P_1 \mid P_2] \mid Q]$ by constructing

the inference tree in Figure 3.6. Hence, by applying the transition rule τ-Out in Table 3.6 we obtain the resulting transition

$$n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} m[P_1 \mid P_2] \mid n[Q]$$

$$\text{(Act)}\ \frac{}{out\ n.P_1 \xrightarrow{out\ n} P_1} \qquad \text{(Act)}\ \frac{}{out\ n.P_1 \xrightarrow{out\ n} P_1}$$

$$\text{(Par)}\ \frac{}{out\ n.P_1 \mid P_2 \xrightarrow{out\ n} P_1 \mid P_2} \qquad \text{(Par)}\ \frac{}{out\ n.P_1 \mid P_2 \xrightarrow{out\ n} P_1 \mid P_2}$$

$$\text{(Exit1)}\ \frac{}{m[out\ n.P_1 \mid P_2] \xrightarrow{exit1\ n} 0} \qquad \text{(Exit2)}\ \frac{}{m[out\ n.P_1 \mid P_2] \xrightarrow{exit2\ n} m[P_1 \mid P_2]}$$

$$\text{(Par-Exit1)}\ \frac{}{m[out\ n.P_1 \mid P_2] \mid Q \xrightarrow{exit1\ n} 0 \mid Q} \qquad \text{(Par1-Exit2)}\ \frac{}{m[out\ n.P_1 \mid P_2] \mid Q \xrightarrow{exit2\ n} m[P_1 \mid P_2]}$$

$$\text{(τ-Out)}\ \frac{}{n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} n[0 \mid Q] \mid m[P_1 \mid P_2] \quad (n[0 \mid Q] \mid m[P_1 \mid P_2] \equiv n[Q] \mid m[P_1 \mid P_2])}$$

$$\text{(Struct)}\ \frac{}{n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} n[Q] \mid m[P_1 \mid P_2]}$$

Figure 3.6: Inference tree for τ-Out transition

Now by Struct, $n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} m[P_1 \mid P_2] \mid n[Q]$ and $m[P_1 \mid P_2] \mid n[Q] \equiv R$. Hence, we obtain $n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} R$ as required.

(c) $P = P' \mid Q$, parallel composition of the processes.
In this case we show

$$P' \mid Q \longrightarrow R \Longrightarrow P' \mid Q \xrightarrow{\tau} R \tag{3.3}$$

Assume $P' \mid Q \to R$ for some R.

There are two reduction rules Red-In and Red-Par in Table 3.4, that can be used to deduce this reduction. To apply each rule separately, we divide this case into two sub-cases.

i. Red-Par:
Since $P' \mid Q \to R$ is valid by Red-Par, we deduce that the reduction $P' \to S$ is valid, for some S, such that $S \mid Q = R$. Since $P' \to S$ is valid, so by inductive hypothesis $P' \xrightarrow{\tau} S$ is valid.
Since $P' \xrightarrow{\tau} S$ is valid, by the transition rule Par in Table 3.7, we deduce that the conclusion of the rule is also valid, namely $P' \mid Q \xrightarrow{\tau} S \mid Q$.

By Struct we have $P' \mid Q \xrightarrow{\tau} S \mid Q$ and $S \mid Q \equiv R$, hence we get $P' \mid Q \xrightarrow{\tau} R$ as required.

ii. Red-In:
Assume that P' and Q are of the form $m[in\ n.P_1 \mid P_2]$ and $n[Q_1]$ respectively. By parallel composition of the two agents we obtain by the reduction rule Red-In

$$m[in\ n.P_1 \mid P_2] \mid n[Q_1] \rightarrow R$$

We further deduce that $R = n[m[P_1 \mid P_2] \mid Q_1]$.
Now we derive the $\xrightarrow{\tau}$ of agent $m[in\ n.P_1 \mid P_2] \mid n[Q_1]$ by applying the SOS rule τ-In given in Table 3.6. This is supported by the inference tree in Figure 3.7. Hence, the resulting transition is

$$m[in\ n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q_1]$$

$$\dfrac{\dfrac{\dfrac{}{in\ n.P_1 \xrightarrow{in\ n} P_1}(\text{Act})}{in\ n.P_1 \mid P_2 \xrightarrow{in\ n} P_1 \mid P_2}(\text{Par}) \quad \dfrac{\dfrac{}{in\ n.P_1 \xrightarrow{in\ n} P_1}(\text{Act})}{in\ n.P_1 \mid P_2 \xrightarrow{in\ n} P_1 \mid P_2}(\text{Par})}{m[in\ n.P_1 \mid P_2] \xrightarrow{enter1\ n} m[P_1 \mid P_2]}(\text{Enter1}) \quad \dfrac{}{m[in\ n.P_1 \mid P_2] \xrightarrow{enter2\ n} 0}(\text{Enter2}) \quad \dfrac{\dfrac{}{n[Q_1] \xrightarrow{move2\ n} 0}(\text{Co-Enter2})}{n[Q_1] \xrightarrow{move1\ n} Q_1}(\text{Co-Enter1})$$

$$\dfrac{}{m[in\ n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q_1] \mid 0 \equiv n[m[P_1 \mid P_2] \mid Q_1]}(\tau\text{-In})$$

$$\dfrac{}{m[in\ n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q_1]}(\text{Struct})$$

Figure 3.7: Inference tree for τ-In transition

Now by Struct, $m[in\ n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q_1]$ and $n[m[P_1 \mid P_2] \mid Q_1] \equiv R$. Hence, we obtain $m[in\, n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} R$ as required.

□

3.4 On Completeness of Operational Semantics

Completeness ensures that transition semantics of CM can correctly match all possible reductions, namely for every valid τ-transition of a CM term there is a valid reduction of the term, and the targets of the τ-transitions and the reductions are

the same. The statement $\forall P, P' \in CM.\ P \xrightarrow{\tau} P' \Longrightarrow P \rightarrow P'$ states that the operational semantics of CM is consistent with the reduction semantics given in Section 3.1.2. But this statement does not hold for some elaborate examples. For example, consider a setting (building), where two ambients (rooms) have the same name. Our SOS rules allow us to derive a τ-transition between two ambients such that there is no reduction between the ambients. This is because the SOS rules do not have the ability to distinguish between the two occurrences of the same ambient name. In practice, this is quite unusual for the two rooms with the same name, but to ensure completeness we consider all possible transitions.

We list four statements in Table 3.8 that need to be true to ensure completeness. These statements do not hold in general for our current operational semantics of CM. The following counter examples show why they do not hold.

Example 3.1. Consider the process

$$(m[in\ n.S] \mid m[in\ n.Q]) \mid (n[R])$$

In this example two ambients with the same name m want to enter ambient n by exercising the capability $in\ n$. To exercise this capability of m, the τ-In rule in Table 3.6 is applied. The resulting transition is as follows:

$$(m[in\ n.S] \mid m[in\ n.Q]) \mid (n[R]) \xrightarrow{\tau} n[m[S] \mid R] \mid m[in\ n.S]$$

The inference tree that proves the validity of the above given transition is in Figure 3.8. There exists no reduction relation that matches this τ-transition. We show this case as follows.

Let $P = m[in\ n.S] \mid m[in\ n.Q]$ is the agent that performs $in\ n$ capability, where $P' = m[S]$ is the part of P that is moved in and the agent $P'' = m[in\ n.S] \mid 0$ is the part of P which is left behind, after the τ-In rule is applied. Since $P \xrightarrow{enter1\ n} \nu\tilde{p}\ (P')$ and $P \xrightarrow{enter2\ n} \nu\tilde{p}\ (P'')$, so we need to check if the structure of P, P', P'' obtained from these transitions agrees with the structure stated in statement 2 in Table 3.8. We have $P = \nu\tilde{p}(m[in\ n.S] \mid m[in\ n.Q])$, where $P_1 = S,\ P_2 = m[in\ n.Q]$ and $P_3 = 0$, where $\tilde{p}$ is the empty set of names. So, $P' = m[S]$ and $P'' = m[in\ n.S]$. We now have $P'' = m[in\ n.Q]$ by statement 2 in Table 3.8, and we also obtain $P'' = m[in\ n.S]$ by the given example. The structures of the two agents do not match, so the result of statement 2 in Table 3.8 is not true.

1. $\forall P, P' \in CM.\ P \xrightarrow{\mu} P',$ where $\mu \in \{in\ n, out\ n\} \Longrightarrow \exists\ \tilde{p}, P_1, P_2$with $n \notin \tilde{p}$ such that $P \equiv \nu\tilde{p}\ (\mu.P_1 \mid P_2)$ and $P' \equiv \nu\tilde{p}\ (P_1 \mid P_2),$ where $\tilde{p}$ is the set of ambient names private in P.
2. $\forall P, P', P'', \tilde{p} \in CM.\ P \xrightarrow{enter1\ n} \nu\tilde{p}\ (P')$ and $P \xrightarrow{enter2\ n} \nu\tilde{p}\ (P'') \Longrightarrow \exists\ P_1, P_2,$ P_3, m, n with $n, m \notin \tilde{p}$ such that $P \equiv \nu\tilde{p}\ (m[in\ n.P_1 \mid P_3] \mid P_2),$ $P' \equiv m[P_1 \mid P_3]$ and $P'' \equiv P_2,$ where $\tilde{p}$ is the set of ambient names private in P.
3. $\forall Q, Q', Q'' \in CM.\ Q \xrightarrow{move1\ n} \nu\tilde{q}(Q')$ and $Q \xrightarrow{move2\ n} \nu\tilde{q}(Q'') \Longrightarrow \exists Q_1, Q_2, Q_3,$ n with $n \notin \tilde{q}$ such that $Q \equiv \nu\tilde{q}\ (n[Q_1 \mid Q_3] \mid Q_2), Q' \equiv Q_1 \mid Q_3$ and $Q'' \equiv Q_2,$ where $\tilde{q}$ is the set of ambient names private in Q.
4. $\forall P, P', P'' \in CM.\ P \xrightarrow{exit1\ n} \nu\tilde{m}(P')$ and $P \xrightarrow{exit2\ n} \nu\tilde{m}(P'') \Longrightarrow \exists\ P_1, P_2, P_3,$ k with $n, k \notin \tilde{m},$ such that $P \equiv \nu\tilde{m}\ (k[out\ n.P_1 \mid P_2] \mid P_3),\ P' \equiv P_3$ and $P'' \equiv k[P_1 \mid P_2],$ where $\tilde{m}$ is the set of ambient names private in P.

Table 3.8: Statements to ensure completeness

Example 3.2. Consider

$$(m[in\ n.P]) \mid (n[S] \mid n[R])$$

In this example the ambient m has the capability to enter the ambient n, and there exist two ambients with the same name n in parallel with the moving ambient. To exercise this capability τ-In rule in Table 3.6 is applied. The resulting transition is as follows:

$$(m[in\ n.P]) \mid (n[S] \mid n[R]) \xrightarrow{\tau} n[m[P] \mid S] \mid n[S]$$

The inference tree that proves the validity of this τ-transition is given in Figure 3.9. There exists no reduction relation that could match this τ-transition. We show this case as follows:

In this example we have, $Q = (n[S] \mid n[R])$ is an agent that allows any ambient to enter n, $Q' = S$ is the part of agent that stays inside the host ambient n and $Q'' = n[S] \mid 0$, stays outside the host ambient, after the τ-In rule is applied. Since $Q \xrightarrow{move1\ n} \nu\tilde{q}(Q')$ and $Q \xrightarrow{move2\ n} \nu\tilde{q}(Q'')$, so we want to check if the structure of Q, Q', Q'' obtained from these transitions agrees with the structure stated in statement 2 in Table 3.8. We have $Q = \nu\tilde{q}(n[S] \mid n[R])$, where $Q_1 \mid Q_3 = S,\ Q_2 = n[R]$ and $\tilde{q}$ is the empty set of names. So, $Q' = S$ and $Q'' = n[S]$. We now have $Q'' = n[R]$ by statement 3 in Table 3.8, and we also obtain $Q'' = n[S]$ from this example. Since the two agents do not match, so there is no reduction relation matching the transition $(m[in\ n.P]) \mid (n[S] \mid n[R]) \xrightarrow{\tau} n[m[P] \mid S] \mid n[P]$. Hence, the result of

$$\text{(Act)}\ \frac{}{in\ n.S \xrightarrow{in\ n} S} \qquad \text{(Act)}\ \frac{}{in\ n.Q \xrightarrow{in\ n} Q}$$

$$\text{(Enter1)}\ \frac{}{m[enter1\ n.S] \xrightarrow{in\ n} S} \qquad \text{(Enter2)}\ \frac{}{m[enter2\ n.Q] \xrightarrow{in\ n} 0}$$

$$\text{(Co-Enter2)}\ \frac{}{n[R] \xrightarrow{move2\ n} 0}$$

$$\text{(Par-Enter1)}\ \frac{}{m[in\ n.S] \mid m[in\ n.Q] \xrightarrow{enter1\ n} m[S]} \qquad \text{(Par-Enter2)}\ \frac{}{m[in\ n.S] \mid m[in\ n.Q] \xrightarrow{enter2\ n} m[in\ n.S] \mid 0} \qquad \text{(Co-Enter1)}\ \frac{}{n[R] \xrightarrow{move1\ n} R}$$

$$(\tau\text{-In})\ \frac{}{(m[in\ n.S] \mid m[in\ n.Q]) \mid (n[R]) \xrightarrow{\tau} n[m[S] \mid R] \mid m[in\ n.S] \mid 0 \mid 0 \equiv n[m[S] \mid R] \mid m[in\ n.S]}$$

$$\text{(Struct)}\ \frac{}{(m[in\ n.S] \mid m[in\ n.Q]) \mid (n[R]) \xrightarrow{\tau} n[m[S] \mid R] \mid m[in\ n.S]}$$

Figure 3.8: Inference tree for Example 3.1

$$\text{(Act)}\ \frac{}{in\ n.P \xrightarrow{in\ n} P} \qquad \text{(Act)}\ \frac{}{in\ n.P \xrightarrow{in\ n} P} \qquad \text{(Co-Enter1)}\ \frac{}{n[S] \xrightarrow{move1\ n} S} \qquad \text{(Co-Enter2)}\ \frac{}{n[R] \xrightarrow{move2\ n} 0}$$

$$\text{(Enter1)}\ \frac{}{m[in\ n.P] \xrightarrow{enter1\ n} m[P]} \qquad \text{(Enter2)}\ \frac{}{m[in\ n.P] \xrightarrow{enter2\ n} 0} \qquad \text{(Par-Move1)}\ \frac{}{(n[S] \mid n[R]) \xrightarrow{move1\ n} S} \qquad \text{(Par-Move2)}\ \frac{}{(n[S] \mid n[R]) \xrightarrow{move2\ n} n[S] \mid 0}$$

$$(\tau\text{-In})\ \frac{}{(m[in\ n.P]) \mid (n[S] \mid n[R]) \xrightarrow{\tau} n[m[P] \mid S] \mid 0 \mid n[S] \mid 0 \equiv n[m[P] \mid S] \mid n[S]}$$

$$\text{(Struct)}\ \frac{}{(m[in\ n.P]) \mid (n[S] \mid n[R]) \xrightarrow{\tau} n[m[P] \mid S] \mid n[S]}$$

Figure 3.9: Inference tree for Example 3.2

statement 3 in Table 3.8 is not always true.

Example 3.3. Consider

$$n[m[out\ n.S] \mid m[out\ n.T]]$$

In this example two sibling ambients with same name m want to exit the ambient n by exercising the capability *out* n. To exercise this capability, τ-Out rule given in Table 3.6 is applied. The resulting transition is as follows:

$$n[m[out\ n.S] \mid m[out\ n.T]] \xrightarrow{\tau} n[m[out\ n.T]] \mid m[T]$$

The inference tree given in Figure 3.10 proves the validity of this τ-transition. There exists no reduction relation that matches this τ-transition. We show this case as follows:

Let $P = m[out\ n.S] \mid m[out\ n.T]$ is the agent that performs *out* n capability, where $P'' = m[T]$ is the part of P that is moved out, and $P' = 0 \mid m[out\ n.T]$ is the part of P which is left behind. By applying the τ-Out rule. Since $P \xrightarrow{exit1\,n} \nu\tilde{m}(P')$ and $P \xrightarrow{exit2\,n} \nu\tilde{m}(P'')$, so we want to check if the structure of P, P', P'' obtained from these transitions agrees with the structure stated in statement 2 in Table 3.8. We have $P = m[out n.S] \mid m[out n.T]$, where $S = P_1 \mid P_2,\ P_3 = m[out n.T]$ and $\tilde{p}$ is the empty set of names. So, $P' = m[out\ n.T]$ and $P'' = m[T]$. We now have $P'' = m[S]$ by statement 4 in Table 3.8, and we also obtain $P'' = m[T]$ from this example. Since the two agents do not match, so there is no reduction relation There is no reduction relation matching the valid transition $n[m[out\ n.S] \mid m[out\ n.T]] \xrightarrow{\tau} n[m[out\ n.T]] \mid m[T]$. Hence, the result of statement 4 in Table 3.8 is not always true.

We finish this section by stating that a new complete operational semantics for CM will be proposed in the next chapter.

3.5 Conclusion

In this chapter we have reviewed the syntax and reduction semantics of Mobile Ambients [11]. We have reused only the mobility part of MA and developed a new and simple operational semantics for CM. Therefore, we have called the calculus as, a Calculus of Mobility (CM). We have proved that the semantics is sound with respect to the standard reduction semantics. Our labelled transition semantics has been inspired by that in [36, 37]. To preserve the standard MA semantics, we have not used

$$\text{(Act)}\ \frac{}{out\ n.S \xrightarrow{out\ n} S} \qquad \text{(Act)}\ \frac{}{out\ n.T \xrightarrow{out\ n} T}$$

$$\text{(Exit1)}\ \frac{}{m[out\ n.S] \xrightarrow{exit1\ n} 0} \qquad \text{(Exit2)}\ \frac{}{m[out\ n.T] \xrightarrow{exit2\ n} m[T]}$$

$$\text{(Par-Exit1)}\ \frac{}{m[out\ n.S] \mid m[out\ n.T] \xrightarrow{exit1\ n} 0 \mid m[out\ n.T]} \qquad \text{(Par1-Exit2)}\ \frac{}{m[out\ n.S] \mid m[out\ n.T] \xrightarrow{exit2\ n} m[T]}$$

$$\text{(τ-Out)}\ \frac{}{n[m[out\ n.S] \mid m[out\ n.T]] \xrightarrow{\tau} n[0 \mid m[out\ n.T]] \mid m[T] \equiv n[m[out\ n.T]] \mid m[T]}$$

$$\text{(Struct)}\ \frac{}{n[m[out\ n.S] \mid m[out\ n.T]] \xrightarrow{\tau} n[m[out\ n.T]] \mid m[T]}$$

Figure 3.10: Inference tree for Example 3.3

the co-capabilities and concretions in our operational semantics as used in [36, 37]. We have proceeded to analyse the completeness of our operational semantics, and discovered that the semantics is not complete for some unusual examples. We have given three examples where the transition rules allowed us to derive τ-transitions for which we have not found any corresponding reductions. These examples will help us to develop a new operational semantics in the next chapter.

Chapter 4

An LTS Based Operational Semantics of a Calculus of Mobility

In this chapter we propose a new complete operational semantics for CM. The operational semantics given in Chapter 3 is not complete for certain unusual cases which are possibly not very useful from a practical point of view. For example, if two ambients with the same name which intend to perform *in* or *out* capabilities. Then the transition rules from previous chapter allow us to derive a τ-transition that does not have the corresponding reduction. Consider a setting

$$m[in\ n.P] \mid m[in\ n.Q] \mid n[R]$$

where two ambients named m intend to enter an ambient n. The transition

$$m[in\ n.P] \mid m[in\ n.Q] \mid n[R] \xrightarrow{\tau} n[m[P] \mid R] \mid m[in\ n.P])$$

could be derived using the operational semantics from Chapter 3. We note that Q has "disappeared" from the target of the transition. The possible reductions of the given term are

$$m[in\ n.P] \mid m[in\ n.Q] \mid n[R] \rightarrow n[m[P] \mid R] \mid m[in\ n.Q]$$

and

$$m[in\ n.P] \mid m[in\ n.Q] \mid n[R] \rightarrow n[m[Q] \mid R] \mid m[in\ n.P]$$

There exists no such reduction matching the right hand term of the above τ-transition. This problem is because the SOS rules for Chapter 3 do not have the

ability to distinguish between the ambients with same name and identical capabilities. To resolve such problems, we now use concretions of the form $\nu\tilde{m}\langle P\rangle Q$ in our new SOS rules, as inspired by [36, 37, 39, 40]. In a transition rule, concretion may split an agent into two distinguished subagents and keep them together in a single premise. Our SOS rules are similar to those in [36, 37] but with few differences, namely we do not use the co-capabilities and passwords in our semantics, hence preserving the standard MA semantics. Completeness ensures that the operational semantics of the calculus is consistent with the standard reduction semantics. We prove that our new operational semantics coincides with the standard reduction semantics.

The following section includes the syntax and new operational structural operational semantics for CM. In Section 4.2 we show that the structural operational semantics of CM coincides with the standard reduction semantics. Section 4.3 concludes the chapter.

4.1 The Syntax and SOS Rules of CM

The syntax of CM processes is the same as in Chapter 3. The SOS rules are presented in Tables 4.2 and 4.3, and use auxiliary actions, namely *enter* n, *move* n and *exit* n which are defined in Table 4.1.

In our new labelled transition semantics $P \xrightarrow{\tau} Q$ represents ambients' mobility by means of their *in* n and *out* n capabilities. In order to model mobility by τ-transitions additional labels and auxiliary terms are used, namely labels λ and concretions K in Table 4.1. So we will need auxiliary transitions $P \xrightarrow{\lambda} O$, where P is a process, λ is a label and O represents an outcome in Table 4.1, which is either a process or *concretion* of the form $(\nu\tilde{m})\langle P\rangle Q$ as introduced by Milner [44] and used by Merro and Hennessy [36, 37]. We adopt the following convention after [37]. If K is the concretion $\nu\tilde{m}\langle P\rangle Q$, then $\nu u K$ stands for $\nu\tilde{m}\langle P\rangle\nu u(Q)$, if $u \notin fn(P)$, otherwise $\nu(u\tilde{m})\langle P\rangle Q$. A similar convention is followed for λ-Par in Table 4.3. We define $K \mid P'$ as the concretion $\nu\tilde{m}\langle P\rangle(Q \mid P')$ where, using α-conversion if necessary, $\tilde{m}$ is selected in such a way that $fn(P') \cap \tilde{m} = \emptyset$.

Transitions $P \xrightarrow{\lambda} O$ are not first class transitions; they are only helpful in SOS rules that define τ-transitions of processes corresponding to the movement by the *in* n and *out* n capabilities.

We consider some examples to show some reductions and, at the same time, explain how auxiliary labels in Table 4.1 and transition rules in Tables 4.2 and 4.3

are used in defining mobility transitions. We assume

$$m[in\ n.P] \mid Q) \mid n[R] \text{ for some } P,\ Q,\ R.$$

The ambient m has the capability to enter the ambient n. By Red In axiom in Table 3.4 we have

$$m[in\ n.P] \mid Q \mid n[R] \longrightarrow n[m[P] \mid R] \mid Q$$

We now derive the τ-transition of $m[in\ n.P] \mid Q \mid n[R]$ by τ-In rule in Table 4.2. For simplicity, we assume that there are no private names in Q and R. We have $in\ n.P \xrightarrow{in\ n} P$. When the migration occurs, we must identify the moving ambient m, and the agent that is left behind. To model these two agents we use concretion $\nu\tilde{m}\langle P\rangle Q$, where P is the agent that moves, while Q is the agent that stays behind, and $\tilde{m}$ is the set of private names shared by P and Q. We introduce a new action $enter\ n$ and have

$$m[in\ n.P] \xrightarrow{enter\ n} \langle m[P]\rangle 0$$

By λ-Par in Table 4.3 we obtain

$$m[in\ n.P] \mid Q \xrightarrow{enter\ n} \langle m[P]\rangle Q$$

Next, to achieve the τ-transition there must exist a sibling ambient n. We define a new action $move\ n$ for n to complete this interaction. By τ-In we get

$$m[in\ n.P] \mid Q \mid n[R] \xrightarrow{\tau} n[m[P] \mid R] \mid Q$$

After the transition the ambient m, becomes a child of n.

Next, we explain emigration capability by considering $m[n[out\ m.P] \mid Q]$, for some P and Q where Q has no private names. The ambient n may emigrate from m by its $out\ m$ capability. By Red Out we have

$$m[n[out\ m.P] \mid Q] \rightarrow n[P] \mid m[Q]$$

We derive the τ-transition of $m[n[out\ m.P] \mid Q]$ by τ-Out. We define a new action $exit\ m$, and by Exit in Table 4.2 we get

$$n[out\ m.P] \xrightarrow{exit\ m} \langle n[P]\rangle 0$$

Ambient Prefixes :	μ	::=	$in\, n$	|	$out\, n$			
Ambient Actions :	λ	::=	$enter\, n$	|	$move\, n$	| $exit\, n$	|	μ
Labels :	ℓ	::=	μ	|	λ	| τ		
Outcomes :	O	::=	P	|	K			
Concretions :	K	::=	$(\nu\tilde{m})\langle P\rangle Q$					

Table 4.1: Prefixes, labels, outcomes and concretions

$$\text{(Act)}\ \frac{}{\mu.P \xrightarrow{\mu} P}$$

$$\text{(Enter)}\ \frac{P \xrightarrow{in\, n} P'}{m[P] \xrightarrow{enter\, n} \langle m[P']\rangle 0} \qquad \text{(Co-Enter)}\ \frac{}{n[P] \xrightarrow{move\, n} \langle P\rangle 0}$$

$$\text{(τ-In)}\ \frac{P \xrightarrow{enter\, n} (\nu\tilde{p})\langle P'\rangle P'' \qquad Q \xrightarrow{move\, n} (\nu\tilde{q})\langle Q'\rangle Q''}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p})(\nu\tilde{q})(n[P' \mid Q'] \mid P'' \mid Q'')}\ (*)$$

$$\text{(Exit)}\ \frac{P \xrightarrow{out\, n} P'}{m[P] \xrightarrow{exit\, n} \langle m[P']\rangle 0} \qquad \text{(τ-Out)}\ \frac{P \xrightarrow{exit\, n} (\nu\tilde{m})\langle P'\rangle P''}{n[P] \xrightarrow{\tau} (\nu\tilde{m})(P' \mid n[P''])}\ (**)$$

$(*)(fn(P') \cup fn(P'')) \cap \tilde{q} = (fn(Q') \cup fn(Q'')) \cap \tilde{p} = \emptyset$

$(**)(fn(P') \cup fn(P'')) \cap \tilde{m} = \emptyset$

Table 4.2: Transition rules for mobility

By λ-Par we get

$$n[out\ m.P] \mid Q \xrightarrow{exit\, m} \langle n[P]\rangle Q,$$

which shows that when this capability is exercised $n[P]$ moves out, while the process Q remains inside m. By τ-Out we have

$$m[n[out\ m.P] \mid Q] \xrightarrow{\tau} n[P] \mid m[Q]$$

After the transition the ambient n, becomes a sibling of m.

Next, we consider examples where agents share private names and show that ambients mobility extends the scope of its restricted names beyond their boundaries. We assume

$$(\nu k)(m[in\ n.0 \mid in\ k.0] \mid k[P]) \mid (n[Q] \mid R),$$

where ambients m and k execute in parallel and they share a private name k. The ambient m has the capabilities to either enter k or n. We consider the ability of m to enter n which is running in parallel sharing no private names with the moving

$$(\lambda\text{-Par})\ \frac{P \xrightarrow{\lambda} O}{P \mid Q \xrightarrow{\lambda} O \mid Q}\ (*) \qquad (\lambda\text{-Res})\ \frac{P \xrightarrow{\lambda} O}{(\nu u)P \xrightarrow{\lambda} (\nu u)O}\ (u \notin fn(\lambda))\ (*)$$

$$(\tau\text{-Amb})\ \frac{P \xrightarrow{\tau} P'}{n[P] \xrightarrow{\tau} n[P']} \qquad (\text{Struct})\ \frac{P \equiv Q \quad Q \xrightarrow{l} Q' \quad Q' \equiv P'}{P \xrightarrow{l} P'}$$

$(**)$The definition of λ is extended to include also a τ

Table 4.3: Transition rules for other operators of CM

ambient. To exercise this capability of agent m, we use $(\nu\tilde{p})$ and $(\nu\tilde{q})$ in τ-In rule. These constructs maintain privacy and sharing of the restricted names in the agents after they move around. We also assume that there are no private names in agents Q and R, and $k \notin fn(Q, R)$.

The application of the τ-In rule specifies how this capability of the agent m is achieved. The resulting transition is:

$$(\nu k)(m[in\ n.0 \mid in\ k.0] \mid k[P]) \mid (n[Q] \mid R) \xrightarrow{\tau} (\nu k)(\nu\tilde{q})(k[P] \mid n[m[in\ k.0] \mid Q] \mid R)$$

In this transition we observe that scope of k is extended to outside m, since $k \notin fn(Q)$ or $fn(R)$. The inference tree that shows the validity of above given transition is given in Figure 4.1.

Similarly, we consider a system

$$n[\nu k(m[in\ k.0 \mid out\ n.0]) \mid P]$$

In this example we restrict the scope of private name k to agent m, and process P executes in parallel with m inside n. We assume that there is no private names in P, and $k \notin fn(P)$. The application of τ-Out gives the resulting transition as follows:

$$n[\nu k(m[in\ k.0 \mid out\ n.0]) \mid P] \xrightarrow{\tau} \nu k(m[in\ k.0] \mid n[P])$$

Here, after exercising the *out* capability, the scope of k is extended to include P since $k \notin fn(P)$, and m which is a child of n, becomes a sibling of n. If $k \in fn(P)$ then the above scope extrusion is not valid. The inference tree for the above given transition is in Figure 4.2.

$$\text{(Act)}\ \frac{}{in\ n.0 \xrightarrow{in\ n} 0}$$

$$\text{(Par)}\ \frac{}{in\ n.0 \mid in\ k.0 \xrightarrow{in\ n} 0 \mid in\ k.0}$$

$$\text{(Enter)}\ \frac{}{m[in\ n.0 \mid in\ k.0] \xrightarrow{enter\ n} \langle m[0 \mid in\ k.0]\rangle 0}$$

$$\text{(λ-Par)}\ \frac{}{m[in\ n.0 \mid in\ k.0] \mid k[P] \xrightarrow{enter\ n} \langle m[0 \mid in\ k.0]\rangle 0 \mid k[P]} \qquad \text{(Co-Enter)}\ \frac{}{n[Q] \xrightarrow{move\ n} \langle Q\rangle 0}$$

$$\text{(Par)}\ \frac{}{n[Q] \mid R \xrightarrow{move\ n} (\nu\tilde{q})(\langle Q\rangle 0 \mid R) \equiv (\nu\tilde{q})(\langle Q\rangle (0 \mid R))}\ (q \notin fn(R))$$

$$\text{(Res-Amb)}\ \frac{}{(\nu k)(m[in\ n.0 \mid in\ k.0] \mid k[P]) \xrightarrow{enter\ n} (\nu k)\langle m[0 \mid in\ k.0]\rangle 0 \mid k[P] \equiv (\nu k)\langle m[0 \mid in\ k.0]\rangle 0 \mid k[P]}\ (k \notin fn(P))$$

$$\text{(τ-In)}\ \frac{}{(\nu k)(m[in\ n.0 \mid in\ k.0] \mid k[P]) \mid (n[Q] \mid R) \xrightarrow{\tau} (\nu k)(\nu\tilde{q})(n[m[0 \mid in\ k.0] \mid Q] \mid k[P] \mid (0 \mid R)) \equiv (\nu k)(\nu\tilde{q})(n[m[in\ k.0] \mid Q] \mid k[P] \mid R)}$$

$$\text{(Struct)}\ \frac{}{(\nu k)(m[in\ n.0 \mid in\ k.0] \mid k[P]) \mid (n[Q] \mid R) \xrightarrow{\tau} (\nu k)(\nu\tilde{q})(n[m[in\ k.0] \mid Q] \mid k[P] \mid R)}$$

Figure 4.1: Inference tree for enter capability

$$\text{(Act)}\ \frac{}{out\ n.0 \xrightarrow{out\ n} 0}$$

$$(\lambda\text{-Par})\ \frac{}{in\ k.0 \mid out\ n.0 \xrightarrow{out\ n} in\ k.0 \mid 0}$$

$$\text{(Exit)}\ \frac{}{m[in\ k.0 \mid out\ n.0] \xrightarrow{exit\ n} \langle m[in\ k.0 \mid 0]\rangle 0}$$

$$\text{(Res-Amb)}\ \frac{}{\nu k(m[in\ k.0 \mid out\ n.0]) \xrightarrow{exit\ n} \nu k(\langle m[in\ k.0 \mid 0]\rangle 0)}$$

$$(\lambda\text{-Par})\ \frac{}{\nu k(m[in\ k.0 \mid out\ n.0]) \mid P \xrightarrow{exit\ n} \nu k(\langle m[in\ k.0 \mid 0]\rangle 0) \mid P \equiv \nu k(\langle m[in\ k.0 \mid 0]\rangle (0 \mid P)))}\ (k \notin fn(P))$$

$$(\tau\text{-Out})\ \frac{}{n[\nu k(m[in\ k.0 \mid out\ n.0]) \mid P] \xrightarrow{\tau} \nu k(m[in\ k.0 \mid 0] \mid n[0 \mid P]) \equiv \nu k(m[in\ k.0] \mid n[P])}$$

$$\text{(Struct)}\ \frac{}{n[\nu k(m[in\ k.0 \mid out\ n.0]) \mid P] \xrightarrow{\tau} \nu k(m[in\ k.0] \mid n[P])}$$

Figure 4.2: Inference tree for exit capability

4.2 Correspondence of Transition Semantics and Reduction Semantics

In this section we show that our transition semantics coincides with the reduction semantics of CM. There are "soundness" and "completeness" parts of this result.

4.2.1 Soundness

Soundness ensures that for every reduction of a term of CM there is a valid τ-transition of the term, and the target of the τ-transition is congruent to the target of the reduction.

Theorem 4.1. $\forall P, R \in CM.\ P \to R \Longrightarrow \exists\, Q \in CM.\ P \xrightarrow{\tau} Q \equiv R.$

Proof. We prove Theorem 4.1 by structural induction where we consider cases of reductions of P depending on the structure of P.

1. Base case: (Constant)
   We show that our statement holds when we choose the simplest term of CM, namely the deadlocked agent 0:

   $$0 \to R \Longrightarrow 0 \xrightarrow{\tau} R$$

   There is no rule defined to show the reduction of 0, so $0 \to R$ is false, and hence, the implication above is true.

2. Induction Hypothesis:
   We assume that our theorem holds for all the sub-processes P' of P, such that if $P' \rightarrow R \Longrightarrow P' \xrightarrow{\tau} R$, for all R.

3. Induction Step:

   (a) $P = (\nu m)P'$ for some P'. In this case we show

   $$(\nu m)P' \rightarrow R \Longrightarrow (\nu m)P' \xrightarrow{\tau} R \tag{4.1}$$

   We assume $(\nu m)P' \rightarrow R$.

   The only reduction rule that can be used to derive a reduction relation of $(\nu m)P'$ is Red Res in Table 3.4. Since the term $(\nu m)P'$ reduces to some process R, by the reduction rule Red Res, so we deduce that the reduction $P' \rightarrow Q$ is also valid for some Q, such that $(\nu m)Q = R$. Since $P' \rightarrow Q$ is valid, so by the inductive hypothesis we get $P' \xrightarrow{\tau} Q$.

   Since $P' \xrightarrow{\tau} Q$ is valid, so by transition rule λ-Res in Table 4.3, we deduce that the transition $(\nu m)P' \xrightarrow{\tau} (\nu m)Q$ is valid. Since $m \notin fn(\tau)(= \emptyset)$.

   Now using the structural congruence rule Struct in Table 4.3, we obtain $(\nu m)P' \xrightarrow{\tau} (\nu m)Q$ and $(\nu m)Q \equiv R$. Hence,we obtain $(\nu m)P' \xrightarrow{\tau} R$ as required.

   (b) $P = n[P']$ for some P' and n.
   In this case we show

   $$n[P'] \rightarrow R \Longrightarrow n[P'] \xrightarrow{\tau} R \tag{4.2}$$

   To prove statement 4.2, we assume $n[P'] \rightarrow R$, for some R. There are two reduction rules Red Amb and Red Out in Table 3.4, that can be used to derive a reduction of $n[P']$. So, to apply each rule separately, we divide this case into two sub-cases.

   i. Red Amb:
      Since using Red Amb, $n[P'] \rightarrow R$ is a valid reduction, so $P' \rightarrow Q$ is also valid for some Q. By this rule the agent R is of the form $n[Q]$, namely $R = n[Q]$. As $P' \rightarrow Q$ is a valid reduction, so by inductive hypothesis we get $P' \xrightarrow{\tau} Q$. Since $P' \xrightarrow{\tau} Q$ is a valid transition so,

$$(\text{Act})\ \frac{}{out\ n.P_1 \xrightarrow{out\ n} P_1}$$

$$(\lambda\text{-Par})\ \frac{}{out\ n.P_1 \mid P_2 \xrightarrow{out\ n} P_1 \mid P_2}$$

$$(\text{Exit})\ \frac{}{m[out\ n.P_1 \mid P_2] \xrightarrow{exit\ n} \langle m[P_1 \mid P_2]\rangle 0}$$

$$(\lambda\text{-Par})\ \frac{}{m[out\ n.P_1 \mid P_2] \mid Q \xrightarrow{exit\ n} \langle m[P_1 \mid P_2]\rangle 0 \mid Q \equiv \langle m[P_1 \mid P_2]\rangle (0 \mid Q)}$$

$$(\tau\text{-Out})\ \frac{}{n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} m[P_1 \mid P_2] \mid n[0 \mid Q] \equiv m[P_1 \mid P_2] \mid n[Q]}$$

Figure 4.3: Inference tree for τ-Out transition

by transition rule τ-Amb in Table 3.4, we deduce that the conclusion of the rule is also valid, that is $n[P'] \xrightarrow{\tau} n[Q]$.
Now by Struct rule, $n[P'] \xrightarrow{\tau} n[Q]$ and $n[Q] \equiv R$. Hence, $n[P'] \xrightarrow{\tau} R$ as required.

ii. Red Out:
By using the rule Red Out, we derive that P' is of the form $m[out\ n.P_1 \mid P_2] \mid Q$ for some P_1, P_2 and Q. Hence, the reduction $n[m[out\ n.P_1 \mid P_2] \mid Q] \to R$ is valid by Red Out, where $R = m[P_1 \mid P_2] \mid n[Q]$.
In principle there could be private names in agents P_1, P_2 and Q. However, using α-conversion if necessary, we can assume without loss of generality (wlog) that there are no private names. We derive the $\xrightarrow{\tau}$ of agent $n[m[out\ n.P_1 \mid P_2] \mid Q]$ by applying the transition rule τ-Out in Table 4.2 (with the empty set of private names). This is supported by inference tree in Figure 4.3. Hence, the resulting transition is

$$n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} m[P_1 \mid P_2] \mid n[Q]$$

Now using Struct in Table 4.3, we obtain $n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} m[P_1 \mid P_2] \mid n[Q]$ and $m[P_1 \mid P_2] \mid n[Q] \equiv R$, as equality is the subset of our congruence ($=\subset\equiv$). Hence, $n[m[out\ n.P_1 \mid P_2] \mid Q] \xrightarrow{\tau} R$ as required.

(c) $P = P' \mid Q$, parallel composition of the processes.

In this case we show

$$P' \mid Q \rightarrow R \Longrightarrow P' \mid Q \xrightarrow{\tau} R \tag{4.3}$$

Assume $P' \mid Q \rightarrow R$ for some R.

There are two reduction rules Red In and Red Par in Table 3.4, that can be used to deduce this reduction. To apply each rule separately, we divide this case into two sub-cases.

i. Red Par:
Since $P' \mid Q \rightarrow R$ is valid by Red Par, we deduce that the reduction $P' \rightarrow S$ is valid, for some S, such that $S \mid Q = R$. Since $P' \rightarrow S$ is valid, so by inductive hypothesis $P' \xrightarrow{\tau} S$ is valid.

Since $P' \xrightarrow{\tau} S$ is valid, by the transition rule λ-Par in Table 4.3, we deduce that $P' \mid Q \xrightarrow{\tau} S \mid Q$ is valid. Since $R \equiv S \mid Q$, we get $P' \mid Q \xrightarrow{\tau} R$ as required.

ii. Red In :
Assume that P' and Q are of the form $m[in\ n.P_1 \mid P_2]$ and $n[Q_1]$ respectively. In principle there could be shared and private names in agents P_1, P_2 and Q_1. However, using α-conversion if necessary, we assume wlog that there are no shared and private names. By parallel composition of the two agents we obtain by reduction rule Red In

$$m[in\ n.P_1 \mid P_2] \mid n[Q_1] \rightarrow R$$

We further deduce that $R = n[m[P_1 \mid P_2] \mid Q_1]$. Now we derive the $\xrightarrow{\tau}$ of agent $m[in\ n.P_1 \mid P_2] \mid n[Q_1]$ by applying the SOS rule τ-In in Table 4.2. This is supported by inference tree in Figure 4.4. Hence, the resulting transition is

$$m[in\ n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q_1]$$

Now using the congruence rule Struct, we obtain $m[in\ n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q_1]$ and $n[m[P_1 \mid P_2] \mid Q_1] \equiv R$. Hence, $m[in\ n.P_1 \mid P_2] \mid n[Q_1] \xrightarrow{\tau} R$ as required.

□

$$\text{(Act)}\ \frac{}{in\ n.P_1 \xrightarrow{in\ n} P_1}$$

$$(\lambda\text{-Par})\ \frac{}{in\ n.P_1 \mid P_2 \xrightarrow{in\ n} P_1 \mid P_2}$$

$$\text{(Enter)}\ \frac{}{m[in\ n.P_1 \mid P_2] \xrightarrow{enter\ n} \langle m[P_1 \mid P_2]\rangle 0} \qquad \text{(Co-Enter)}\ \frac{}{n[Q] \xrightarrow{move\ n} \langle Q\rangle 0}$$

$$(\lambda\text{-Par})\ \frac{}{m[in\ n.P_1 \mid P_2] \mid S \xrightarrow{enter\ n} \langle m[P_1 \mid P_2]\rangle (0 \mid S)} \qquad (\lambda\text{-Par})\ \frac{}{n[Q] \mid T \xrightarrow{move\ n} \langle Q\rangle (0 \mid T)}$$

$$\tau\text{-In}\ \frac{}{m[in\ n.P_1 \mid P_2] \mid S \mid n[Q] \mid T \xrightarrow{\tau} n[m[P_1 \mid P_2] \mid Q] \mid 0 \mid S \mid 0 \mid T \equiv n[m[P_1 \mid P_2] \mid Q]}$$

Figure 4.4: Inference tree for τ-In transition

4.2.2 Completeness

Completeness ensures that for every valid τ-transition of a CM term there is a valid reduction of the term, and the targets of the τ-transitions and the reductions are the same.

Lemma 4.1.

1. $\forall P \in CM.\ P \xrightarrow{\mu} O,$ *where* $\mu \in \{in\ n, out\ n\} \Longrightarrow \exists\ \tilde{p}, P_1, P_2$ *with* $n \notin \tilde{p}$ *such that* $P \equiv \nu\tilde{p}\ (\mu.P_1 \mid P_2)$ *and* $O \equiv \nu\tilde{p}\ (P_1 \mid P_2),$ *where* $\tilde{p}$ *is a set of private ambient names in* P.
2. $\forall P, P', P'' \in CM.\ P \xrightarrow{enter\ n} \nu\tilde{p}\ \langle P'\rangle P'' \Longrightarrow \exists\ P_1, P_2, P_3, k$ *with* $n \notin \tilde{p}$ *such that* $P \equiv \nu\tilde{p}\ (k[in\ n.P_1 \mid P_2] \mid P_3),\ P' \equiv k[P_1 \mid P_2]$ *and* $P'' \equiv P_3,$ *where* $\tilde{p}$ *is a set of private ambient names in* P.
3. $\forall Q,\ Q',\ Q'' \in CM.\ Q \xrightarrow{move\ n} \nu\tilde{q}\ \langle Q'\rangle Q'' \Longrightarrow \exists\ Q_1, Q_2,$ *with* $n \notin \tilde{q}$ *such that* $Q \equiv \nu\tilde{q}\ (n[Q_1] \mid Q_2), Q' \equiv Q_1$ *and* $Q'' \equiv Q_2,$ *where* $\tilde{q}$ *is a set of private ambient names in* Q.
4. $\forall P, P', P'' \in CM.\ P \xrightarrow{exit\ n} \nu\tilde{m}\ \langle P'\rangle P'' \Longrightarrow \exists\ P_1, P_2, P_3,\ k$ *with* $n \notin \tilde{m}$ *such that* $P \equiv \nu\tilde{m}\ (k[out\ n.P_1 \mid P_2] \mid P_3),\ P' \equiv k[P_1 \mid P_2]$ *and* $P'' \equiv P_3,$ *where* $\tilde{m}$ *is a set of private ambient names in* P.

Proof. By transition induction.

1. There are three transition rules, namely Act in Table 4.2, and λ-Res and λ-Par in Table 4.3, that can be used to prove part 1 of Lemma 4.1. So, we consider three cases. Let $\mu = in\ n$.

   (a) Act

   In this case P is of the form $in\ n.R$. Since $in\ n.R \xrightarrow{in\ n} R$ is a valid transition by Act. Therefore, $P \xrightarrow{in\ n} O$ is a valid transition, where $P =$

in $n.R$. Hence, $P \equiv \nu\tilde{p}(\mu.P_1 \mid P_2)$, where $\tilde{p}$ is the empty set of names, and $P_1 = R$ and $P_2 = 0$. Similarly, $O = R$ and hence, $O \equiv \nu\tilde{p}(P_1 \mid P_2)$, where $P_2 = 0$ and $\tilde{p}$ is the empty set of ambient names in P.

(b) λ-Par

In this case we assume that P is of the form $R \mid S$, for some R and S. So, for $\mu = in\ n$, the transition $R \mid S \stackrel{in\,n}{\rightarrow} O \mid S$ is valid by λ-Par, hence, the premise $R \stackrel{in\,n}{\rightarrow} O$ of the rule is also valid.

Since $R \stackrel{in\,n}{\rightarrow} O$ is a valid transition, so by inductive hypothesis $R \equiv \nu\tilde{r}\ (\mu.R_1 \mid R_2)$ and $O \equiv \nu\tilde{r}\ (R_1 \mid R_2)$, for some R_1, R_2 and $\tilde{r}$, where $\tilde{r}$ is the set of private names. By α-conversion, if necessary, we can select $\tilde{r}$ in such a way that $fn(S) \cap \tilde{r} = \emptyset$. So, we get

$$\begin{aligned} R \mid S &\equiv \nu\tilde{r}(\mu.R_1 \mid R_2) \mid S \\ &\equiv \nu\tilde{r}((\mu.R_1 \mid R_2) \mid S) \quad \text{(Struct Res Par)} \\ &\equiv \nu\tilde{r}(\mu.R_1 \mid (R_2 \mid S)) \quad \text{(Struct Par Assoc)} \end{aligned}$$

Similarly, by the rule λ-Par, $O \mid S$ is of the form $\nu\tilde{r}(R_1 \mid R_2) \mid S$. We obtain

$$\begin{aligned} O \mid S &\equiv \nu\tilde{r}(R_1 \mid R_2) \mid S \\ &\equiv \nu\tilde{r}((R_1 \mid R_2) \mid S) \quad \text{(Struct Res Par)} \\ &\equiv \nu\tilde{r}(R_1 \mid (R_2 \mid S)) \quad \text{(Struct Par Assoc),} \end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{p}(\mu.P_1 \mid P_2)$, and $O \equiv \tilde{p}(P_1 \mid P_2)$, where $\tilde{p} = \tilde{r}, P_1 = R_1$ and $P_2 = R_2 \mid S$ as required.

(c) λ-Res

In this case we assume that $P \equiv (\nu u)R$, for some R. We get the transition $(\nu u)R \stackrel{in\,n}{\rightarrow} (\nu u)O$, where O is an outcome and $u \neq n$. As this is a valid transition by rule λ-Res in Table 4.3, hence, the premises $R \stackrel{in\,n}{\rightarrow} O$ of the rule is also valid.

As $R \stackrel{in\,n}{\rightarrow} O$ is a valid transition, so by inductive hypothesis $R \equiv \nu\tilde{p}\ (\mu.R_1 \mid R_2)$ and $O \equiv \nu\tilde{p}\ (R_1 \mid R_2)$, for some R_1, R_2 and $\tilde{p}$, where, $\tilde{p}$ is a set of private names in R . So, we get $\nu u(R) \equiv \nu u(\nu\tilde{p}(\mu.R_1 \mid R_2))$, and $\nu u(O) \equiv \nu u(\nu\tilde{p}(R_1 \mid R_2))$. The new restriction is $\nu\tilde{p'} = \nu(u\tilde{p})$;

and u is added to the set of existing restricted names $\nu\tilde{p}$. Now we get

$$\begin{aligned} \nu u(R) &\equiv \nu u(\nu\tilde{p}(\mu.R_1 \mid R_2)) \\ &\equiv (\nu u)(\nu\tilde{p})(\mu.R_1 \mid R_2) \\ &\equiv \nu(u\tilde{p})(\mu.R_1 \mid R_2) \\ &\equiv \nu\tilde{p'}(\mu.R_1 \mid R_2) \end{aligned}$$

Similarly,

$$\begin{aligned} \nu u(O) &\equiv \nu u(\nu\tilde{p}(R_1 \mid R_2)) \\ &\equiv (\nu u)(\nu\tilde{p})(R_1 \mid R_2) \\ &\equiv \nu(u\tilde{p})(R_1 \mid R_2) \\ &\equiv \nu\tilde{p'}(R_1 \mid R_2) \end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{p'}(\mu.P_1 \mid P_2)$ and $O \equiv \nu\tilde{p'}(P_1 \mid P_2)$, where $P_1 = R_1, P_2 = R_2$ and $\tilde{p'}$ is the set of private ambient names in P.

The proof for the three cases for $\mu = out\ n$ is very similar to the proof for $\mu = in\ n$, so it is omitted.

2. There are three transition rules namely Enter in Table 4.2, and λ-Par and λ-Res in Table 4.3, that can be used to prove this lemma. Depending on the structure of agent P, we consider three cases.

   (a) Enter

   In this case we assume that $P \equiv m[R]$, for some R.

   Since $m[R] \xrightarrow{enter\ n} \langle m[R']\rangle 0$ is valid by Enter in Table 4.2, so the premise of the rule $R \xrightarrow{in\ n} R'$, for some R' is also valid. By part 1 of Lemma 4.1, R has the form $\nu\tilde{r}(in\ n.R_1 \mid R_2)$, for some R_1, R_2 and $\tilde{r}$.

   Consider $m[R]$ and assume wlog that $m \notin \tilde{r}$. We have

$$\begin{aligned} m[\nu\tilde{r}(in\ n.R_1 \mid R_2)] &\equiv (\nu\tilde{r})m[in\ n.R_1 \mid R_2] \qquad \text{if } m \notin \tilde{r} \\ &\equiv (\nu\tilde{r})(m[in\ n.R_1 \mid R_2] \mid 0) \end{aligned}$$

   Since by Enter rule, $m[in\ n.R_1 \mid R_2] \mid 0 \xrightarrow{enter\ n} \langle m[R_1 \mid R_2]\rangle 0$, so by Struct rule we get

$$\begin{aligned} (\nu\tilde{r})(m[in\ n.R_1 \mid R_2] \mid 0) &\xrightarrow{enter\ n} \nu\tilde{r}(\langle m[R_1 \mid R_2]\rangle 0 \mid 0) \\ &\equiv \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle 0 \end{aligned}$$

   Hence, we obtain $P \equiv \nu\tilde{r}(m[in\ n.R_1 \mid R_2] \mid 0),\ P' \equiv m[R_1 \mid R_2],\ P'' \equiv 0$.

(b) λ-Par

In this case we assume that $P \equiv R \mid Q$. So we get the transition of the form $R \mid Q \stackrel{enter\, n}{\longrightarrow} O \mid Q$, for some O.

Since $R \mid Q \stackrel{enter\, n}{\longrightarrow} O \mid Q$ is a valid transition by rule λ-Par in Table 4.3, the premise of the rule, namely $R \stackrel{enter\, n}{\longrightarrow} O$, is also valid.

Since $R \stackrel{enter\, n}{\longrightarrow} O$ is valid, so by inductive hypothesis $R \equiv \nu\tilde{r}(m[in\ n.R_1 \mid R_2] \mid R_3)$, $R' \equiv m[R_1 \mid R_2]$, $R'' \equiv R_3$ and $\tilde{r}$ is a set of private names in R, for some R_1, R_2 and R_3. Here, $O = \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle R_3$. By α-conversion, if necessary, we can select $\tilde{r}$ in such a way that $fn(Q) \cap \tilde{r} = \emptyset$. So, we get

$$\begin{aligned} R \mid Q &\equiv \nu\tilde{r}(m[in\ n.R_1 \mid R_2] \mid R_3) \mid Q \\ &\equiv \nu\tilde{r}(m[in\ n.R_1 \mid R_2] \mid R_3 \mid Q) \end{aligned}$$

Similarly by λ-Par, $O \mid Q$ is defined as concretion of the form $\nu\tilde{r}\langle m[R_1 \mid R_2]\rangle R_3 \mid Q$, and we get

$$\begin{aligned} O \mid Q &\equiv \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle R_3 \mid Q \\ &\equiv \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle (R_3 \mid Q) \end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{r}(m[in\ n.R_1 \mid R_2] \mid R_3 \mid Q)$, $P' \equiv m[R_1 \mid R_2]$, $P'' \equiv R_3 \mid Q$ and $\tilde{r}$ is the set of private names in P as required.

(c) λ-Res

In this case we assume that $P \equiv (\nu u)R$. So we get the transition of the form, $(\nu u)R \stackrel{enter\, n}{\longrightarrow} (\nu u)O$, for some O. Since $(\nu u)R \stackrel{enter\, n}{\longrightarrow} (\nu u)O$ is valid by λ-Res in Table 4.3, the premise of the rule, namely $R \stackrel{enter\, n}{\longrightarrow} O$, is also valid.

Since $R \stackrel{enter\, n}{\longrightarrow} O$ is valid, so by inductive hypothesis $R \equiv \nu\tilde{p}(m[in\ n.R_1 \mid R_2] \mid R_3)$, $R' \equiv m[R_1 \mid R_2]$ and $R'' = R_3$, for some $\tilde{p}, m, R_1, R_2$ and R_3. Now, we get $\nu u(R) \equiv \nu u(\nu\tilde{p}\ (m[in\ n.R_1 \mid R_2] \mid R_3))$. The new restriction is $\nu\tilde{p'} = \nu(u\tilde{p})$; and u is added to the set of existing restricted names $\nu\tilde{p}$. We obtain

$$\begin{aligned} (\nu u)R &\equiv \nu u(\nu\tilde{p}(m[in\ n.R_1 \mid R_2] \mid R_3)) \\ &\equiv (\nu u)(\nu\tilde{p})(m[in\ n.R_1 \mid R_2] \mid R_3) \\ &\equiv \nu(u\tilde{p})(m[in\ n.R_1 \mid R_2] \mid R_3) \\ &\equiv (\nu\tilde{p'})(m[in\ n.R_1 \mid R_2] \mid R_3) \end{aligned}$$

Next, the outcome $O \equiv \nu\tilde{p}\langle m[R_1 \mid R_2]\rangle R_3$, and hence $(\nu u)O \equiv \nu\tilde{p}\langle m[R_1 \mid R_2]\rangle \nu u R_3$ if $u \notin fn(R_1 \mid R_2)$, else we get the concretion of the form

$\nu(u\tilde{p})\langle m[R_1 \mid R_2]\rangle R_3$. In the second case we have

$$\begin{aligned}(\nu u)O &\equiv \nu(u\tilde{p})\langle m[R_1 \mid R_2]\rangle R_3 \\ &\equiv (\nu u)(\nu\tilde{p})\langle m[R_1 \mid R_2]\rangle R_3) \\ &\equiv \nu(u\tilde{p})(\langle m[R_1 \mid R_2]\rangle R_3) \\ &\equiv (\nu\tilde{p'})(\langle m[R_1 \mid R_2]\rangle R_3)\end{aligned}$$

For this case we obtain $P \equiv \nu\tilde{p'}\ (m[in\ n.R_1 \mid R_2] \mid R_3), P' \equiv m[R_1 \mid R_2]$ and $P'' \equiv R_3$, and $\tilde{p'}$ is the set of private ambient names.

In the first case, i-e. if $u \notin fn(R_1 \mid R_2)$, we get

$$\begin{aligned}(\nu u)R &\equiv \nu u(\nu\tilde{p}(m[in\ n.R_1 \mid R_2] \mid R_3)) \\ &\equiv (\nu\tilde{p})(\nu u)(m[in\ n.R_1 \mid R_2] \mid R_3) \\ &\equiv (\nu\tilde{p})(m[in\ n.R_1 \mid R_2] \mid (\nu u)R_3)\end{aligned}$$

Similarly,

$$\begin{aligned}(\nu u)O &\equiv \nu u(\nu\tilde{p}\langle m[R_1 \mid R_2]\rangle R_3) \\ &\equiv \nu\tilde{p}\langle m[R_1 \mid R_2]\rangle(\nu u)R_3, \quad u \notin fn(R_1 \mid R_2)\end{aligned}$$

Overall we obtain $P \equiv \nu\tilde{p}(m[in\ n.R_1 \mid R_2] \mid (\nu u)R_3), P' \equiv m[R_1 \mid R_2]$ and $P'' \equiv (\nu u)R_3$, and $\tilde{p}$ is the set of private ambient names.

3. The proof for the Lemma 4.1 part 3 is very similar to the proof for part 2, so it is omitted.

4. There are three transition rules, namely Exit, λ-Par, λ-Res given in Tables 4.2 and 4.3, that can be used to prove this lemma. Depending on the structure of agent P, we consider three cases.

   (a) Exit

   In this case we assume that $P \equiv m[R]$, for some R, and hence we get the transition of the form $m[R] \stackrel{exit\ n}{\longrightarrow} \langle m[R']\rangle 0$, for some R'. Since $m[R] \stackrel{exit\ n}{\longrightarrow} \langle m[R']\rangle 0$ is valid by Exit in Table 4.2, so the premise of the rule, namely $R \stackrel{out\ n}{\rightarrow} R'$ is also valid. By part 1 of Lemma 4.1, R has the form $\nu\tilde{r}(out\ n.R_1 \mid R_2)$, for some R_1, R_2 and $\tilde{r}$ where $n \notin \tilde{r}$.

   Consider $m[R]$ and assume wlog that $m \notin \tilde{r}$. We have

$$\begin{aligned} m[\nu\tilde{r}(out\ n.R_1 \mid R_2)] &\equiv (\nu\tilde{r})m[out\ n.R_1 \mid R_2] \qquad if\ m \notin \tilde{r} \\ &\equiv (\nu\tilde{r})(m[out\ n.R_1 \mid R_2] \mid 0) \end{aligned}$$

Since by Exit rule, $m[out\ n.R_1 \mid R_2] \mid 0 \xrightarrow{exit\ n} \langle m[R_1 \mid R_2]\rangle 0$, so by Struct rule we get

$$\begin{aligned} (\nu\tilde{r})(m[out\ n.R_1 \mid R_2] \mid 0) &\xrightarrow{exit\ n} \nu\tilde{r}(\langle m[R_1 \mid R_2]\rangle 0 \mid 0) \\ &\equiv \quad \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle 0 \end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{r}(m[out\ n.R_1 \mid R_2] \mid 0), P' \equiv m[R_1 \mid R_2], P'' \equiv 0$ and $\tilde{p} \equiv \tilde{r}$ is the set of private names in P.

(b) λ-Par

In this case we assume that $P \equiv R \mid Q$. So we get the transition of the form $R \mid Q \xrightarrow{exit\ n} O \mid Q$, for some O. Since $R \mid Q \xrightarrow{exit\ n} O \mid Q$ is a valid transition by λ-Par in Table 4.2, the premise of the rule, namely $R \xrightarrow{exit\ n} O$, is also valid.

Since $R \xrightarrow{exit\ n} O$ is valid, so by inductive hypothesis $R \equiv \nu\tilde{r}(m[out\ n.R_1 \mid R_2] \mid R_3)$, $R' \equiv m[R_1 \mid R_2]$, $R'' \equiv R_3$ and $\tilde{r}$ is a set of private names in R, for some R_1, R_2 and R_3. Here, $O = \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle R_3$. By α-conversion, if necessary, we can select $\tilde{r}$ in such a way that $fn(Q) \cap \tilde{r} = \emptyset$. So, we get

$$\begin{aligned} R \mid Q &\equiv \nu\tilde{r}(m[out\ n.R_1 \mid R_2] \mid R_3) \mid Q \\ &\equiv \nu\tilde{r}(m[out\ n.R_1 \mid R_2] \mid R_3 \mid Q) \end{aligned}$$

Similarly by λ-Par, $O \mid Q$ is defined as concretion of the form $\nu\tilde{r}\langle m[R_1 \mid R_2]\rangle R_3 \mid Q$, and we get

$$\begin{aligned} O \mid Q &\equiv \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle R_3 \mid Q \\ &\equiv \nu\tilde{r}\langle m[R_1 \mid R_2]\rangle (R_3 \mid Q) \end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{r}(m[out\ n.R_1 \mid R_2] \mid R_3 \mid Q)$, $P' \equiv m[R_1 \mid R_2]$, $P'' \equiv R_3 \mid Q$ and $\tilde{r}$ is the set of private names in P as required.

(c) λ-Res

In this case we assume that $P \equiv (\nu u)R$. So we get the transition of the form, $(\nu u)R \xrightarrow{exit\ n} (\nu u)O$, for some O. Since $(\nu u)R \xrightarrow{exit\ n} (\nu u)O$ is valid by λ-Res in Table 4.3, the premise of the rule, namely $R \xrightarrow{exit\ n} O$, is also valid.

Since $R \xrightarrow{exit\, n} O$ is valid, so by inductive hypothesis $R \equiv \nu\tilde{p}(m[out\ n.R_1 \mid R_2] \mid R_3)$, $R' \equiv m[R_1 \mid R_2]$ and $R'' = R_3$, for some $\tilde{p}, m, R_1, R_2$ and R_3. Now, we get $\nu u(R) \equiv \nu u(\nu\tilde{p}\ (m[out\ n.R_1 \mid R_2] \mid R_3))$. The new restriction is $\nu\tilde{p'} = \nu(u\tilde{p})$; and u is added to the set of existing restricted names $\nu\tilde{p}$. We obtain

$$\begin{aligned}(\nu u)R &\equiv \nu u(\nu\tilde{p}(m[out\ n.R_1 \mid R_2] \mid R_3)) \\ &\equiv (\nu u)(\nu\tilde{p})(m[out\ n.R_1 \mid R_2] \mid R_3) \\ &\equiv \nu(u\tilde{p})(m[out\ n.R_1 \mid R_2] \mid R_3) \\ &\equiv (\nu\tilde{p'})(m[out\ n.R_1 \mid R_2] \mid R_3)\end{aligned}$$

Next, the outcome $O \equiv \nu\tilde{p}\langle m[R_1 \mid R_2]\rangle R_3$, and hence $(\nu u)O \equiv \nu\tilde{p}\langle m[R_1 \mid R_2]\rangle \nu u R_3$ if $u \notin fn(R_1 \mid R_2)$, else we get the concretion of the form $\nu(u\tilde{p})\langle m[R_1 \mid R_2]\rangle R_3$. In the second case we have

$$\begin{aligned}(\nu u)O &\equiv (\nu u)(\nu\tilde{p}\langle m[R_1 \mid R_2]\rangle R_3) \\ &\equiv (\nu u)(\nu\tilde{p})\langle R_1 \mid R_2\rangle R_3) \\ &\equiv \nu(u\tilde{p})(\langle m[R_1 \mid R_2]\rangle R_3) \\ &\equiv (\nu\tilde{p'})\langle m[R_1 \mid R_2]\rangle R_3\end{aligned}$$

For this case we obtain $P \equiv \nu\tilde{p'}\ (m[out\ n.R_1 \mid R_2] \mid R_3), P' \equiv m[R_1 \mid R_2]$ and $P'' \equiv R_3$, and $\tilde{p'}$ is the set of private ambient names in P.

In the first case, i-e. if $u \notin fn(R_1 \mid R_2)$, we get

$$\begin{aligned}(\nu u)R &\equiv \nu u(\nu\tilde{p}(m[out\ n.R_1 \mid R_2] \mid R_3)) \\ &\equiv (\nu\tilde{p})(\nu u)(m[out\ n.R_1 \mid R_2] \mid R_3) \\ &\equiv (\nu\tilde{p})(m[out\ n.R_1 \mid R_2] \mid (\nu u)R_3)\end{aligned}$$

Similarly,

$$\begin{aligned}(\nu u)O &\equiv \nu u(\nu\tilde{p}\langle m[R_1 \mid R_2]\rangle R_3) \\ &\equiv \nu\tilde{p}\langle m[R_1 \mid R_2]\rangle(\nu u)R_3, \quad u \notin fn(R_1 \mid R_2)\end{aligned}$$

Overall we obtain $P \equiv \nu\tilde{p}(m[out\ n.R_1 \mid R_2] \mid (\nu u)R_3), P' \equiv m[R_1 \mid R_2]$ and $P'' \equiv (\nu u)R_3$, and $\tilde{p}$ is the set of private ambient names.

□

This completes the proof of Lemma 4.1.

Finally, we are ready to prove the completeness result.

Theorem 4.2. $\forall S, R \in CM.\ S \xrightarrow{\tau} R \Longrightarrow S \rightarrow R.$

Proof. By transition induction where we consider cases of transitions of S depending on the structure of the term.

1. Base case: (Constant)
   We show that our statement holds when we choose the simplest term of CM, namely the deadlocked agent 0:

   $$0 \xrightarrow{\tau} R \Longrightarrow 0 \rightarrow R$$

   There is no rule defined to show the transition of 0, so the τ-transition for 0 is not possible. Thus, the transition $0 \xrightarrow{\tau} R$ is false, hence, the implication above is true.

2. Induction Hypothesis:
   In this step we assume that our Theorem holds for all the sub-processes P of S, such that if $P \xrightarrow{\tau} R$ then $P \rightarrow R$, for all R.

3. Induction Step:

   (a) $S = C.P$ for some P, where the prefix C is an ambient capability, namely *in* n and *out* n with an ambient name n.

   We consider the ambient's entering capability, and show that

   $$in\ n.P \xrightarrow{\tau} R \Longrightarrow in\ n.P \rightarrow R$$

   The only transition for $in\ n.P$ is $in\ n.P \xrightarrow{in\ n} P$ by applying the rule Act in Table 4.2. Since there is no other rule defined that could be applied to derive the transition, the τ-transition for $in\ n.P$ is not possible. Thus, the transition $in\ n.P \xrightarrow{\tau} R$ is not valid and, hence, the implication above is true.

   The proof for *out* n capability is very similar to the proof for *in* n capability, so its is omitted.

   (b) $S = n[P]$, for some P and n.
   In this case we show that

   $$n[P] \xrightarrow{\tau} R \Longrightarrow n[P] \rightarrow R \tag{4.4}$$

We assume $n[P] \xrightarrow{\tau} R$ for some R.

There are two transition rules τ-Out and τ-Amb in Tables 4.2 and 4.3, that can be used to derive a τ-transition of $n[P]$. So, to apply each rule separately, we divide this case into two sub-cases.

i. τ-Out

Since $n[P] \xrightarrow{\tau} R$ is valid by the transition rule τ-Out in Table 4.2, so we deduce that the premise of the rule, namely $P \xrightarrow{exit\, n} \nu\tilde{m}\langle P'\rangle P''$, for some P' and P'', is also valid. Here, $\tilde{m}$ is the set of private ambient names in process P, and $((fn(P') \cup fn(P'')) \cap \{\tilde{m}\}) = \emptyset$. Since by τ-Out, the process R is of the form $(\nu\tilde{m})(n[P''] \mid P')$, namely $R = (\nu\tilde{m})(n[P'] \mid P'')$. Hence, we obtain

$$n[P] \xrightarrow{\tau} (\nu\tilde{m})(n[P''] \mid P')$$

Now using part 4 of Lemma 4.1, we get $P \equiv \nu\tilde{m}\ (k[out\ n.P_1 \mid P_2] \mid P_3)$, $P' \equiv k[P_1 \mid P_2]$ and $P'' \equiv P_3$, for some P_1, P_2, P_3 and k, where $n \notin \tilde{m}$. Hence, we deduce that,

$$\begin{array}{rcll} n[P] & \equiv & n[\nu\tilde{m}\ (k[out\ n.P_1 \mid P_2] \mid P_3)] & \\ & \equiv & \nu\tilde{m}\ (n[k[out\ n.P_1 \mid P_2] \mid P_3]) & \text{(Struct Res Amb)} \\ & & \text{(where it is assumed wlog that } n \notin \tilde{m}) & \end{array}$$

By Red Out $n[k[out\ n.P_1 \mid P_2] \mid P_3] \rightarrow n[P_3] \mid k[P_1 \mid P_2]$,
so $\nu\tilde{m}\ (n[k[out\ n.P_1 \mid P_2] \mid P_3])$ rewrites as follows:

$$\begin{array}{rcll} & \rightarrow & \nu\tilde{m}\ (n[P_3] \mid k[P_1 \mid P_2]) & \text{(Red Res)} \\ & \equiv & \nu\tilde{m}\ (n[P''] \mid k[P_1 \mid P_2]) & \text{(Struct Amb)} \\ & \equiv & \nu\tilde{m}\ (n[P''] \mid P') & \text{(Struct Amb)} \end{array}$$

Now using rule Red $\equiv$ in Table 3.4, we obtain $n[P] \rightarrow \nu\tilde{m}\ (n[P''] \mid P')$ and $\nu\tilde{m}\ (n[P''] \mid P') \equiv R$, as equality is the subset of our congruence $(= \subset \equiv)$. Hence, the conclusion of the rule is valid, that is: $n[P] \rightarrow R$ as required.

ii. τ-Amb

Since $n[P] \xrightarrow{\tau} R$, for some process R is valid by the transition rule τ-Amb in Table 4.3, we deduce that the premise $P \xrightarrow{\tau} P'$ is valid for some P'. By τ-Amb the agent $R = n[P']$. Since $P \xrightarrow{\tau} P'$ is valid, so by the inductive hypothesis we get $P \rightarrow P'$. Since the reduction

$P \rightarrow P'$ is valid, so by reduction rule Red Amb in Table 3.4, we get $n[P] \rightarrow n[P']$. Since $R \equiv n[P']$ we obtain $n[P] \rightarrow R$ as required.

(c) $S = (\nu m)P$, an ambient name m private in P.
In this case we show that

$$(\nu m)P \xrightarrow{\tau} R \Longrightarrow (\nu m)P \rightarrow R \tag{4.5}$$

We assume $(\nu m)P \xrightarrow{\tau} R$.

The only transition rule that can be used to derive a transition of $(\nu m)P$ is λ-Res in Table 4.3. Since $(\nu m)P \xrightarrow{\tau} R$, for some process R is valid by the transition rule λ-Res, we deduce that the premise $P \xrightarrow{\tau} P'$ is valid for some P'. By λ-Res the agent R is of the form $(\nu m)P'$, that is $R = (\nu m)P'$. Since $P \xrightarrow{\tau} P'$ is valid, so by the inductive hypothesis we get $P \rightarrow P'$. Since the reduction $P \rightarrow P'$ is valid, so by reduction rule Red Res in Table 3.4, we get $\nu m P \rightarrow \nu m P'$. Since $R \equiv \nu m P'$ we obtain $\nu m P \rightarrow R$ as required.

(d) $S = P \mid Q$, parallel composition of the processes.
In this case we show

$$P \mid Q \xrightarrow{\tau} R \Longrightarrow P \mid Q \rightarrow R \tag{4.6}$$

We assume $P \mid Q \xrightarrow{\tau} R$ for some R.

There are only two transition rules λ-Par and τ-In in Tables 4.3 and 4.2, that can be used to derive a τ-transition of $P \mid Q$. So, to apply each rule separately, we divide this case into two sub-cases.

i. λ-Par
Since $P \mid Q \xrightarrow{\tau} R$ is valid by λ-Par, we deduce that the premise of the rule, namely $P \xrightarrow{\tau} P'$ is valid for some P'. By λ-Par, the agent R is of the form $P' \mid Q$, namely $R = P' \mid Q$.
Since $P \xrightarrow{\tau} P'$ is valid, so by inductive hypothesis we obtain $P \rightarrow P'$.
Since the reduction $P \rightarrow P'$ is valid, so by reduction rule Red Par in Table 3.4, we get $P \mid Q \rightarrow P' \mid Q$, where Q is given in the beginning of this case. Since $R = P' \mid Q$ we get $P \mid Q \rightarrow R$ as required.

ii. τ-In
Since $P \mid Q \xrightarrow{\tau} R$ is valid by τ-In rule, we deduce that the premises of the rule are also valid. These premises are $P \xrightarrow{enter\ n} \nu\tilde{p}\langle P'\rangle P''$, for

some P' and P'', and $Q \stackrel{move\ n}{\rightarrow} \nu\tilde{q}\langle Q'\rangle Q''$, for some Q' and Q''. Here $\tilde{p}$ and $\tilde{q}$ are the sets of private ambient names in P and Q respectively, and $(fn(P') \cup fn(P'')) \cap \{\tilde{q}\} = (fn(Q') \cup fn(Q'')) \cap \{\tilde{p}\} = \emptyset$. By τ-In the agent R is of the form $(\nu\tilde{p})(\nu\tilde{q})(n[P' \mid Q'] \mid P'' \mid Q'')$, namely $R = (\nu\tilde{p})(\nu\tilde{q})(n[P' \mid Q'] \mid P'' \mid Q'')$. Hence, the resulting transition is

$$P \mid Q \stackrel{\tau}{\rightarrow} (\nu\tilde{p})(\nu\tilde{q})(n[P' \mid Q'] \mid P'' \mid Q'')$$

Now, by using Lemma 4.1 part 2, we obtain

$$P \equiv \nu\tilde{p}\ (m[in\ n.P_1 \mid P_3] \mid P_2),\ P' \equiv m[P_1 \mid P_3] \text{ and } P'' \equiv P_2$$

We further obtain by part 3 of Lemma 4.1

$Q \equiv \nu\tilde{q}\ (n[Q_1 \mid Q_3] \mid Q_2),\ Q' \equiv Q_1 \mid Q_3$ and $Q'' \equiv Q_2$ Hence, we deduce that

$$P \mid Q \ \equiv\ \nu\tilde{p}\ (m[in\ n.P_1 \mid P_3] \mid P_2) \mid \nu\tilde{q}\ (n[Q_1 \mid Q_3] \mid Q_2)$$

Since members of $\tilde{q}$ are not free names in $\nu\tilde{p}\ (m[in\ n.P_1 \mid P_3] \mid P_2)$, and members of $\tilde{p}$ are not free names in $(n[Q_1 \mid Q_3] \mid Q_2)$, we obtain

$$\begin{aligned}
P \mid Q \ &\equiv\ \nu\tilde{q}(\nu\tilde{p}\ (m[in\ n.P_1 \mid P_3] \mid P_2) \mid (n[Q_1] \mid Q_2)) && \text{(Struct Res Par)}\\
&\equiv\ \nu\tilde{q}((n[Q_1] \mid Q_2) \mid \nu\tilde{p}\ (m[in\ n.P_1 \mid P_3] \mid P_2)) && \text{(Struct Par Comm)}\\
&\equiv\ \nu\tilde{q}(\nu\tilde{p}(n[Q_1] \mid Q_2 \mid m[in\ n.P_1 \mid P_3] \mid P_2)) && \text{(Struct Res Par)}\\
&\equiv\ \nu\tilde{q}(\nu\tilde{p}(n[Q_1] \mid m[in\ n.P_1 \mid P_3] \mid Q_2 \mid P_2)) && \text{(Struct Res Par)}\\
&\equiv\ \nu\tilde{q}(\nu\tilde{p}(n[Q_1] \mid m[in\ n.P_1 \mid P_3] \mid P_2 \mid Q_2)) && \text{(Struct Res Par)}\\
&\equiv\ \nu\tilde{q}(\nu\tilde{p}(m[in\ n.P_1 \mid P_3] \mid n[Q_1] \mid P_2 \mid Q_2)) && \text{(Struct Res Par)}\\
&\rightarrow\ \nu\tilde{p}\ \nu\tilde{q}\ (n[m[P_1 \mid P_3] \mid Q_1] \mid P_2 \mid Q_2) && \text{(Red In)}\\
&\equiv\ \nu\tilde{p}\ \nu\tilde{q}\ (n[P' \mid Q'] \mid P'' \mid Q'')
\end{aligned}$$

Now using the structural congruence rule Red$\equiv$ in Table 3.4, we obtain

$$P \mid Q \rightarrow \nu\tilde{p}\ \nu\tilde{q}\ (n[P' \mid Q'] \mid P'' \mid Q'')$$

and $R \equiv \nu\tilde{p}\ \nu\tilde{q}\ (n[P' \mid Q'] \mid P'' \mid Q'')$, as equality is the subset of our congruence $(= \subset \equiv)$. Hence, $P \mid Q \rightarrow R$ as required.

□

4.3 Conclusion

In this chapter we have presented a new transition semantics for the Calculus of Mobility defined in the previous chapter. The operational semantics given in Chapter 3 is not complete for certain unusual cases where ambients with the same name intend to perform *in* or *out* capabilities. SOS rules developed in previous chapter do not have the ability to distinguish between the ambients with same name performing the identical capabilities. Such problems have been resolved in this chapter by using concretions $\nu\tilde{m}\langle P\rangle Q$ in our SOS rules. We then have showed that the transition semantics of CM is sound and complete with respect to the standard reduction semantics. In the next chapter we will add a new form of global communication to CM.

Chapter 5

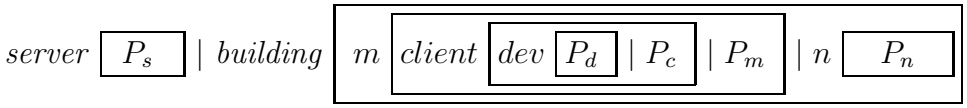
The Calculus of Mobility and Communication

In this chapter we introduce the Calculus of Mobility and Communication (CMC) for the modelling of mobile agents that may communicate globally in the setting of ubiquitous and mobile computing. The mobility feature of the calculus is modelled using *in* and *out* capabilities of Cardelli and Gordon's MA [11, 10]. We introduce a new form of global communication in CMC inspired by Milner's CCS [43]. The usefulness of the calculus is illustrated by two case studies and a number of small examples. The first case study illustrates the usefulness of the calculus by calculating a path between two locations, and allowing the mobile ambient to navigate from one location to another inside a building. The second case study enables services to follow ambients in an intelligent hospital setting. A new notion of behavioural equivalence is introduced by defining two forms of barbs; one for ambients and another for ambients' capabilities. We prove that the congruence relations of the two forms of barbs agree.

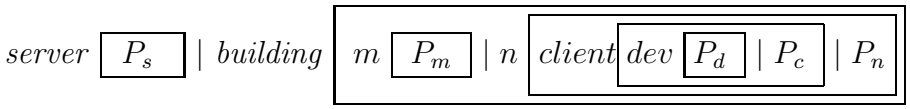
The inspiration for this work comes from several mobile ambient and process calculi based on MA that have proved useful in the modelling of mobility, communication and concurrent systems. The calculus of Mobile Ambients and its variants do not support a direct interaction of an agent with a subagent inside another agent. Communication can only happen between the two adjacent agents, namely communication between parent and child or between siblings.

In ubiquitous computing settings it is beneficial to model interactions among agents that communicate globally. Communication in such settings could be global, which means that agents may interact with subagents inside other agents. For example, consider an agent *server* that instructs a mobile ambient *client* to move from location m to n inside a *building*. We assume that *client* receives information from

server via a mobile device *dev*. Such a setting is represented as follows:



Then after receiving a message from *server*, *client* physically moves from m to n holding *dev*. The new setting is



Mobility, locations and global communication are the main features in such examples. Therefore, in order to model locations and mobility, MA is a suitable calculus. MA helps in modelling both mobile physical mobility of devices as well as logical mobility where code moves between devices. Ambients' *open* capability in combination with *in* and *out* capabilities models local communication between agents. In CMC we introduce a new form of direct and global communication so we drop the *open* capability. Recently, a number of variants of MA have been introduced. In particular Boxed Ambients, BA for short, [6] inherits mobility primitives, namely *in* and *out* capabilities from MA and drops *open* capability to avoid certain risks. BA proposes a new communication mechanism where the additional communication primitives complement the existing constructs of MA in an effective manner. Several variants of BA have been introduced in order to improve the existing calculus. For example, Safe Boxed Ambients [41] uses ambients co-capabilities that help in controlling ambients access across the boundaries. Channel Ambients [53] is an extension of BA where ambients are allowed to move in and out over channels. Our definition of ambient is different from the ambients calculi discussed above. We extend the syntax of CM by introducing a new set of ports/channels A to ambient structure which shall be useful for the new form of global communication among agents. The ambients calculi discussed above do not support a direct interaction of an agent with a subagent inside another agent. Communication can only happen between the two adjacent agents, namely communication between parent and child or between siblings. We introduce a direct method of communication in CMC, as in Milner's CCS. To achieve this we extend the syntax of MA by introducing a new set of port names A to ambient structure. We define ambients $m_A[P]$, where m is the name of the ambient, A is the set of ports that m is allowed to communicate on, and P is an executing agent.

In past few years, several operational semantics have been developed for Mobile Ambients and its variants as, for example, in [36, 37, 40, 38]. The authors

Names :	$m_A, n_B, k_C ... \in \mathcal{N}$						
Actions :	$\alpha, \beta, ... \in Act = \mathcal{A} \cup \overline{\mathcal{A}} \cup \{\tau\}$						
Variables :	$x, y, ... \in \mathcal{X}$						
Processes :	$P, Q ::=$	D		$C.P$	$a(z).P$		$\overline{a}(x).P$
		$m_A[P]$		$P + Q$	$P \mid Q$		$(\nu m_A)P$
		$(\nu l)P$		$P[f]$			
Capabilities :	$C ::=$	x		μ	ϵ		$C.C'$

Table 5.1: Syntax of CMC

in [36, 37] introduce a label transition system based operational semantics, and a labelled bisimulation equivalence which is proved to coincide with reduction barbed congruence. We also develop a new notion of behavioural equivalence for CMC, and formulate the equivalence in terms of α-transitions and observation predicate, inspired by [11, 36, 37].

The chapter is organised as follows. We introduce CMC in Section 5.1. A labelled transition semantics for CMC is given in Section 5.2. Section 5.3 presents two case studies that illustrate the usefulness of CMC. In Section 5.4 we develop behavioural semantics for CMC and Section 5.5 contains conclusions.

5.1 The Syntax of CMC

The syntax of CMC is given in Tables 5.1 and 5.2. In CMC we use the syntax of both CM as discussed in Chapters 3 and 4, and the syntax of Milner's CCS [43].

We assume that $\mathcal{A}$ is an infinite set of port names, which is ranged over by a, b, c, and the set of co-names, denoted by $\overline{\mathcal{A}}$ is ranged over by $\overline{a}, \overline{b}, \overline{c}$. We set $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$ and we let A, B, C range over it. An infinite set Act comprises all possible actions that an agent can perform and α, β range over it. Act also includes τ, which is a single completed action of composite agents. So $Act = \mathcal{L} \cup \{\tau\}$, and the typical subsets of α are A, B. The set of agent constants $\mathcal{K}$ is ranged over by D and E, and the deadlocked agent 0 is a member of $\mathcal{K}$. In our syntax the variable z in $a(z).P$ and in $\overline{a}(z).P$ can be replaced by a value from a set $\mathcal{V}$, which may contain the capabilities as defined in Table 5.1.

We further assume an infinite set of ambient names $\mathcal{N}$ that is ranged over by m_A, n_B and k_C, where $A, B, C \subseteq \mathcal{A} \cup \overline{\mathcal{A}}$. Ambients are terms $m_A[P]$, where m is the name of the ambient, A is the set of ports that ambient m is allowed to communicate

Ambient Prefixes :	$\mu ::=$	$in\ n_B$	$\mid$	$out\ n_B$		
Action Prefixes :	$\alpha ::=$	$a(z)$	$\mid$	$\overline{b}(z)$	$\mid$	τ
Ambient Action :	$\lambda ::=$	$enter\ n_B$	$\mid$	$move\ n_B$		
	$\mid$	$exit\ n_B$	$\mid$	μ		
Labels :	$\ell ::=$	μ	$\mid$	α		
	$\mid$	λ	$\mid$	τ		
Outcomes :	$O ::=$	P	$\mid$	K		
Concretions :	$K ::=$	$(\nu\tilde{m})\langle P\rangle Q$				

Table 5.2: Prefixes, labels, concretions and outcomes

on, and P is an executing agent. When ambients allow communication on all visible ports then we shall write $m[P]$ instead of $m_A[P]$. Other ambient constructs that are inherited from MA are $(\nu m_A)P$, $C.P$ and $C.C'$. An ambient restriction $(\nu m_A)P$ executes process P with a private ambient named m_A. In $C.P$, the process P cannot start execution until the prefix capability C is performed. The capability μ in Tables 5.1 and 5.2 allows ambients to perform certain actions, namely $in\ n_B$ and $out\ n_B$, for some B, whereas $C.C'$ represents a sequence of capabilities (path) when input variable represents one or more of these capabilities. The empty path is represented by ϵ.

We further borrow the constructs for agent constants, action prefixing, parallel composition, summation and action restriction from Milner's CCS or the π-calculus [43, 44]. The agent constant D has a unique equation of the form $D \stackrel{def}{=} P$ where P is an agent that may contain agent constants. The agent constants can also be defined in terms of each other. $\overline{a}(x)$ and $a(z).P$ sends or receives a message on port $\overline{a}$ and a respectively, and then execute P. The received message can be any value $v \in V$, and is bound to the variable z in P. Parallel composition is given in terms of a binary operator, $P \mid Q$, and summation is given by the choice operator $P + Q$ that allows either process P or process Q to execute. In $(\nu l)P$ the port labels l or $\overline{l}$ are restricted in P, where $l \in \mathcal{L}$. In a relabelling $P[f]$, P is a process with the relabelling function f applied to its action labels. Finally, we have the set of terms $T(\Sigma, V)$, where V is the set of process variables, and $T(\Sigma)$, the set of closed terms (agents or processes) ranged over by P, Q.

Free names (revisions and additions)

The addition of communication mechanism changes the ambient definition to $m_A[P]$, where we tag ambient m with a set of ports A. We also add new communication primitives, and therefore revise the definition of free names as given in Table 5.3.

$fn(0)$	$\stackrel{def}{=}$	$\emptyset$
$fn(C.P)$	$\stackrel{def}{=}$	$fn(C) \cup fn(P)$
$fn(a(z).P)$	$\stackrel{def}{=}$	$fn(P) \cup \{a\}$
$fn(\overline{b}(z).P)$	$\stackrel{def}{=}$	$fn(P) \cup \{b\}$
$fn(m_A[P])$	$\stackrel{def}{=}$	$\{m\} \cup A \cup fn(P)$
$fn(P \mid Q)$	$\stackrel{def}{=}$	$fn(P) \cup fn(Q)$
$fn((\nu l(P)))$	$\stackrel{def}{=}$	$fn(P) - fn\{l\} \qquad (l \in \mathcal{L})$
$fn(\nu m_A(P))$	$\stackrel{def}{=}$	$fn(P) - (\{m\} \cup A)$
$fn(P[f])$	$\stackrel{def}{=}$	$f(fn(P))$
$fn(x)$	$\stackrel{def}{=}$	$\emptyset$
$fn(in\ n_B)$	$\stackrel{def}{=}$	$\{n\} \cup B$
$fn(out\ n_B)$	$\stackrel{def}{=}$	$\{n\} \cup B$
$fn(\epsilon)$	$\stackrel{def}{=}$	$\emptyset$
$fn(C.C')$	$\stackrel{def}{=}$	$fn(C) \cup fn(C')$

Table 5.3: Free names

5.1.1 Reduction Semantics of CMC

The reduction semantics is formalised by two concepts: the structural congruence relation, $\equiv$, and the reduction relation $\rightarrow$. We follow the definitions in [37].

Definition 5.1. A relation $\mathcal{R}$ over processes in a process calculus is contextual if it is preserved by all the operators in the process calculus.

Definition 5.2. A relation $\mathcal{R}$ over processes in a process calculus is partially contextual, or p-contextual, w.r.t a set of operators Op if it is preserved by all the operators in the set Op.

The relation $\equiv$, in Definition 3.2, is contextual for CM since it satisfies axioms 4, 5, 7, 8 in Table 5.4 for the four of its operators. However, $\equiv$ is not contextual for CMC since it doesn't satisfy the axiom $P \equiv Q$ infer $P + R \equiv Q + R$. So, we need to define when only some operators of a process calculus preserve a relation R.

Definition 5.3. Structural Congruence, $\equiv$, over CMC processes is the least p-contextual equivalence relation w.r.t the set of operators $Op_1 = \{\nu, |, [f], n_A[\,], C., \alpha.\}$

that satisfies the axioms:

$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$P \mid 0 \equiv P$	(Struct Zero Par)
$P + Q \equiv Q + P$	(Struct Sum Comm)
$(P + Q) + R \equiv P + (Q + R)$	(Struct Sum Assoc)
$P + 0 \equiv P$	(Struct Zero Identity)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \quad if\ n \notin fn(P)$	(Struct Res Par)
$(\nu n_B)(m_A[P]) \equiv m_A[(\nu n_B)P] \quad if\ n \neq m$	(Struct Res Amb)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Struct Res Res)
$(\nu n)0 \equiv 0$	(Struct Zero Res)
$A \equiv P \qquad if\ A \stackrel{def}{=} P$	(Struct Const)
$\epsilon.P \equiv P$	(Epsilon)

By this definition we get the axioms and the rules for $\equiv$ in Table 5.4.

Definition 5.4. Reduction relation, $\rightarrow$, over CMC processes is the least p-contextual relation w.r.t the set of operators $Op_2 = \{\nu, \mid, n_A[\ ]\}$ that satisfies the rule and axioms in Table 5.5.

$m_A[in\ n_B.P \mid Q] \mid n_B[R] \rightarrow n_B[m_A[P \mid Q] \mid R]$	(Red In)
$n_B[m_A[out\ n_B.P \mid Q] \mid R] \rightarrow m_A[P \mid Q] \mid n_B[R]$	(Red Out)
$P \equiv Q,\ Q \rightarrow Q',\ Q' \equiv P' \Rightarrow P \rightarrow P'$	(Red $\equiv$)

Table 5.5: Reduction rules

By this definition we get the axioms and rules for $\rightarrow$, same as in Table 3.4.

5.2 Transition Semantics for CMC

The labelled transition system for CMC is given as follows: The set of CMC processes is the set of states, the set of labels α as in Table 5.2 is the set of transition labels, and the transition relations $\stackrel{\alpha}{\rightarrow}$ are defined by Plotkin's SOS [55] rules in Tables 5.7 and 5.8. These rules are the same as in Chapter 4. In our semantics $P \stackrel{\tau}{\rightarrow} Q$ represents not only binary communication of processes as in CCS but also mobility of ambients by means of their *in* n_B and *out* n_B capabilities. In order to model

$P \equiv P$	(Struct Refl)	(1)
$P \equiv Q \Rightarrow Q \equiv P$	(Struct Symm)	(2)
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	(Struct Trans)	(3)
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Struct Res)	(4)
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	(Struct Par)	(5)
$P \equiv Q \Rightarrow P[f] \equiv Q[f]$	(Struct Rel)	(6)
$P \equiv Q \Rightarrow n_A[P] \equiv n_A[Q]$	(Struct Amb)	(7)
$P \equiv Q \Rightarrow C.P \equiv C.Q$	(Struct Capability)	(8)
$P \equiv Q \Rightarrow \alpha.P \equiv \alpha.Q$	(Struct Action)	(9)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)	(10)
$P + Q \equiv Q + P$	(Struct Sum Comm)	(11)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)	(12)
$(P + Q) + R \equiv P + (Q + R)$	(Struct Sum Assoc)	(13)
$P + 0 \equiv P$	(Struct Zero Identity)	(14)
$P \mid 0 \equiv P$	(Struct Zero Par)	(15)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Struct Res Res)	(16)
$(\nu n)0 \equiv 0$	(Struct Zero Res)	(17)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$ if $n \notin fn(P)$	(Struct Res Par)	(18)
$(\nu n_B)(m_A[P]) \equiv m_A[(\nu n_B)P]$ if $n \neq m$	(Struct Res Amb)	(19)
$A \equiv P \quad if\ A \stackrel{def}{=} P$	(Struct Const def)	(20)
$\epsilon.P \equiv P$	(Epsilon)	(21)

Table 5.4: Structural Congruence

mobility by τ-transitions additional labels and auxiliary terms are used, namely labels λ and concretions K as discussed in Section 4.1.

Communication in CMC is defined as in CCS, so in addition to the SOS rules in Tables 5.7 and 5.8, we have the SOS rules for the communication part of CMC in Table 5.6, as discussed in Section 2.1.1.

$$\text{(Input)}\ \frac{}{a(z).P \xrightarrow{a(v)} P\{v/z\}}\ (v \in V) \qquad \text{(Output)}\ \frac{}{\overline{a}(x).P \xrightarrow{\overline{a}(x)} P}$$

$$\text{(Res-Act)}\ \frac{P \xrightarrow{\alpha} P'}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'}\ (a \notin fn(\alpha)) \qquad \text{(Sum)}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\text{(Par-Com)}\ \frac{P \xrightarrow{a(x)} P' \qquad Q \xrightarrow{\overline{a}(x)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \text{(Par-Act)}\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

$$\text{(Rel)}\ \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]} \qquad \text{(Const)}\ \frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'}\ (A \stackrel{def}{=} P)$$

$$\text{(Struct)}\ \frac{P \equiv Q \quad Q \xrightarrow{l} Q' \quad Q' \equiv P'}{P \xrightarrow{l} P'}$$

$$\text{(Global-Com)}\ \frac{P \xrightarrow{\alpha} P'}{m_A[P] \xrightarrow{\alpha} m_A[P']}\ (if(\alpha = a(x)\ or\ \alpha = \overline{a}(x))\ then\ a \in A)$$

Table 5.6: Transition rules for communication

The SOS rules in Table 5.6 are similar to those in [43], except that we have an additional Global-Com rule that allows ambients to communicate globally only on ports $a \in A$. Recall that when ambients allow communication on all visible channels then we write $m[P]$ instead of $m_A[P]$.

$$(\lambda\text{-Par})\ \frac{P \xrightarrow{\lambda} O}{P \mid Q \xrightarrow{\lambda} O \mid Q}\ ^{(*)} \qquad (\lambda\text{-Res})\ \frac{P \xrightarrow{\lambda} O}{(\nu u)P \xrightarrow{\lambda} (\nu u)O}\ (u \notin fn(\lambda))\ ^{(*)}$$

$$(\tau\text{-Amb})\ \frac{P \xrightarrow{\tau} P'}{n_A[P] \xrightarrow{\tau} n_A[P']} \qquad \text{(Struct)}\ \frac{P \equiv Q \quad Q \xrightarrow{\ell} Q' \quad Q' \equiv P'}{P \xrightarrow{\ell} P'}$$

The definition of λ is extended to include also a τ.

Table 5.8: Transition rules for other operators of CMC

5.3 Applications of CMC

In this section we present two case studies that illustrate the usefulness of CMC. In the first case study, a system calculates a path between two locations which is later

$$\text{(Act)}\ \frac{}{\mu.P \xrightarrow{\mu} P}$$

$$\text{(Enter)}\ \frac{P \xrightarrow{in\, n_B} P'}{m_A[P] \xrightarrow{enter\, n_B} \langle m_A[P']\rangle 0} \qquad \text{(Co-Enter)}\ \frac{}{n_B[P] \xrightarrow{move\, n_B} \langle P\rangle 0}$$

$$\text{(τ-In)}\ \frac{P \xrightarrow{enter\, n_B} (\nu\tilde{p})\langle P'\rangle P'' \qquad Q \xrightarrow{move\, n_B} (\nu\tilde{q})\langle Q'\rangle Q''}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p})(\nu\tilde{q})(n_B[P' \mid Q'] \mid P'' \mid Q'')}\ (*)$$

$$\text{(Exit)}\ \frac{P \xrightarrow{out\, n_B} P'}{m_A[P] \xrightarrow{exit\, n_B} \langle m_A[P']\rangle 0} \qquad \text{(τ-Out)}\ \frac{P \xrightarrow{exit\, n_B} (\nu\tilde{m})\langle P'\rangle P''}{n_B[P] \xrightarrow{\tau} (\nu\tilde{m})(P' \mid n_B[P''])}\ (**)$$

$(*)(\ fn(P') \cup fn(P'')) \cap \tilde{q} = (fn(Q') \cup fn(Q'')) \cap \tilde{p} = \emptyset$.
$(**)(\ fn(P') \cup fn(P'')) \cap \tilde{m} = \emptyset$.

Table 5.7: Transition rules for mobility

sent to a mobile agent to relocate from a source location to some target location. The second case study shows that services follow a mobile ambient, namely we consider an intelligent hospital setting where a doctor moves around a building and receives services on the appropriate screens located in the building.

5.3.1 Calculating Path Between Two Locations

We assume a system *sys* that takes the source and target locations, and calculates a path between the two locations. Next, the system outputs the path as a message to the moving ambient. The path is a sequence of capabilities which allow the ambient to move from the source location to the destination location. We assume a setting

$$\nu\ abc\ (sys[\overline{a}(n).b(x,y).\overline{c}(path(T,x,y))] \mid k[m_1[m_A[a(u).\overline{b}(m_1,u).c(z).z.0]] \mid n_1[n[0]]])$$

The graphical representation of the above expression is given in Figure 5.1.

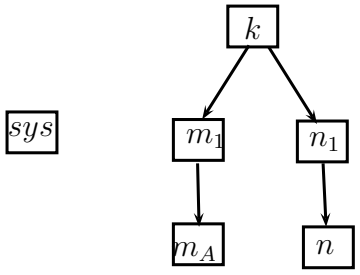


Figure 5.1: Path: source tree

In this figure, ambient k is a building with three rooms m_1, n_1 and n, where n is inside n_1, and $a, b, c \in A$. The ambient m_A for some A, is a moving agent. We assume an independent system *sys* that informs m_A to move from its current location m_1 to n. The above given expression shows the sequence of actions between the system *sys* and the moving ambient m_A. The interaction steps between the agents are:

- $\overline{a}(n)$: The system *sys* sends the target location n on a.
- $a(u)$: The moving agent m_A is ready to receive a value on port a, in this case it receives the target location n.
- $\overline{b}(m_1, u)$: m_A sends the source and target locations on b.
- $b(x, y)$: The system *sys* is ready to receive the two values, in this case m_1 and n as source and target locations respectively.
- $path(T, x, y)$ is a function that calculates a path between the source node x and the target node y in a given tree T.
- $\overline{c}(path(T, x, y))$: Using $path(T, x, y)$ the system calculates a path between the source and the target values received and delivers it to m_A on c.
- $c(z).z.0$: The agent m_A receives a value on port c and binds it to variable z. In this particular case the value received is the sequence of capabilities representing the path between the source and the target ambients.

We have defined several functions (as given in Appendix A) that are used to calculate a path between two locations. The general expression to calculate a path from a source location s to a target location t in a tree T is as follows:

$path(T, s, t) \stackrel{def}{=}$

Sequence(Moves(Join(Path(s, T), Path(t, T), Index(Path(s, T), Path(t, T)))))

In this example the path from m_1 to n is *out* m_1.*in* n_1.*in* n. The sequence of transitions that completes the communication between the two agents is:

$\nu abc(sys[\overline{a}(n).b(x, y).\overline{c}(path(T, x, y))] \mid k[m_1[m_A[a(u).\overline{b}(m_1, u).c(y).y.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc(sys[b(x, y).\overline{c}(path(T, x, y))] \mid k[m_1[m_A[\overline{b}(m_1, n).c(y).y.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc(sys[\overline{c}(path(T, m_1, n))] \mid k[m_1[m_A[c(y).y.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc(sys[0] \mid k[m_1[m_A[out\ m_1.in\ n_1.in\ n.0]] \mid n_1[n[0]]]).$

The last expression contains a sequence of ambient capabilities. The ambient m_A moves from m_1 to n by $out\ m_1.in\ n_1.in\ n$ as shown by the following transitions:

$\nu abc(sys[0] \mid k[m_1[m_A[out\ m_1.in\ n_1.in\ n.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc(sys[0] \mid k[m_1[0] \mid m_A[in\ n_1.in\ n.0] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc(sys[0] \mid k[m_1[0] \mid n_1[m_A[in\ n.0] \mid n[0]]]) \xrightarrow{\tau}$

$\nu abc(sys[0] \mid k[m_1[0] \mid n_1[n[m_A[0] \mid 0]]]).$

The final setting is given in Figure 5.2.

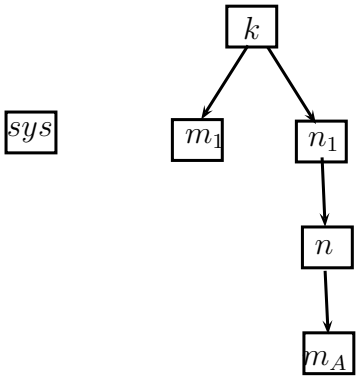


Figure 5.2: Path: target tree

5.3.2 Services Follow Doctor

This section presents a scenario where services follow a doctor. We consider a hospital setting, where doctor moves around the building and deals with patients. While dealing with patients, he may need to use information displayed on the screens that are fixed around the building. The doctor can only read the information on an appropriate screen. We assume that an independent server communicates globally with the doctor and with the screens around the building. The server supplies the services to a screen provided that the doctor is in the same room as the screen.

In this scenario an ambient k represents the building. The ambient k contains ambients dr_K and w_L which represent the doctor's room and the ward respectively. K and L are sets of communication ports, where $b, c_1 \in K$ and $b, c_2 \in L$. This means that the ambient dr_K can communicate at least on ports a and c_1 and the ambient w_L can communicate at least on ports a and c_2. Furthermore, there are two fixed screens scr_{A_1} and scr_{A_2} in dr_K and w_L respectively. A_1 and A_2 are the sets of communication ports, where $c_1 \in A_1$ and $c_2 \in A_2$, but $c_1 \notin A_2$ and $c_2 \notin A_1$. Finally, the doctor is represented as an ambient d_B for some B with $a, b \in B$.

Initially, the ambient d_B is in the doctor's room dr_K and is using services on the

screen scr_{A_1}. He then moves to the ward w_L and continues using the previously left services on the screen scr_{A_2} in w_L. The graphical representation of our setting is given in Figure 5.3. The ambients are represented by boxes, whereas dashed lines represent the communication channels.

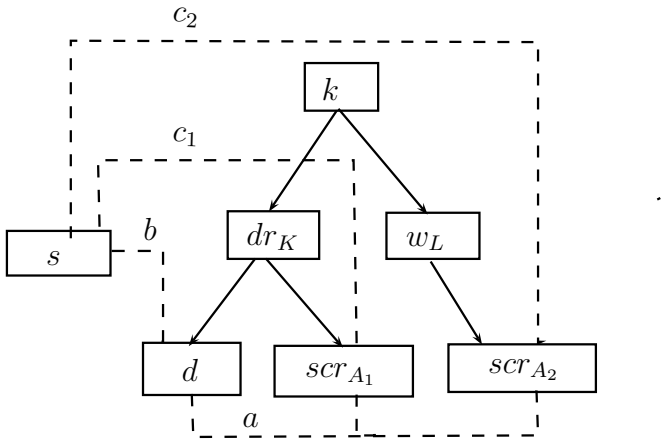


Figure 5.3: Hospital setting: services follow doctor

Next, we show communication among the four agents, namely the doctor, the two screens, and the server. The agents are defined as follows:

Agents *Server* and S are as follows, where l is a finite sequence of values $v_1, v_2, ..., v_k$ for some k:

$$\begin{aligned} Server(v:l) &\stackrel{def}{=} b(x).\ \mathit{if}\ (x = dr_K\ \mathit{then}\ \overline{c_1}(v).Server(l) \\ &\qquad \mathit{else\ if}\ x = w_L\ \mathit{then}\ \overline{c_2}(v).Server(l)\ \mathit{else}\ Server(v:l)) \\ Server(\epsilon) &\stackrel{def}{=} 0 \qquad\qquad S \stackrel{def}{=} s[Server(l)] \end{aligned}$$

Agents $Screen_m$ and Scr_m for $m \in \{1, 2\}$, are defined as follows:

$$Screen_m \stackrel{def}{=} c_m(x).\overline{a}(x).Screen_m \qquad Scr_m \stackrel{def}{=} scr_{A_m}[Screen_m]$$

The agent Scr_{A_m} receives an input x from the server on c_m and outputs x on a. Since $a \in B$, the agent *Doc*, defined below, is able to view x via port a.

Finally, we define agents *Doctor* and *Doc* as follows:

$$
\begin{array}{lll}
Doctor(p,l) & \stackrel{def}{=} & \overline{b}(p).a(x_1).Doctor(p, x_1 : l) \\
Doctor(p, x_1 : l) & \stackrel{def}{=} & \overline{b}(p).a(x_2).Doctor(p, x_2 : x_1 : l) \\
 & & + \; out\, p.\overline{b}(k).in\, r.\overline{b}(r).a(x_2).Doctor(r, x_2 : x_1 : l) \\
 & & \text{if } (p, r \in \{dr_K, w_L\} \text{ and } r \neq p) \\
Doc & \stackrel{def}{=} & d_B[Doctor(dr_K, \epsilon)]
\end{array}
$$

We use p to represent the initial location of *Doc*, here $p = dr_K$. He receives the first piece of information via port a in dr_K. Now there are two possibilities, either to receive the second piece of information or move out of dr_K and view the remaining information on scr_{A_2} in w_L. When *Doc* leaves p by performing *out* p capability, his new location becomes k. He now may enter r by *in* r, and send his location to *Server*. In this particular situation, $r = w_L$ since $r \neq p$ and $p = dr_K$.

The hospital setting is represented by the parallel composition of the server and the building, which contains doctor's room, ward, the doctor and two screens as follows:

$$S \mid k[dr_K[Doc \mid Scr_1] \mid w_L[Scr_2]\ ]$$

For simplicity we assume that the server S sends two values, namely $l = v_2 : v_1 : \epsilon$ for some v_1, v_2. Initially *Doc* is in dr_K and S wants to send the values v_1 and v_2 to *Doc* via either Scr_1 or Scr_2. There are two possible sequences of execution of this setting.

$$
\begin{array}{ll}
(i) & \stackrel{\tau_{b(dr)}}{\longrightarrow} \stackrel{\tau_{c_1(v_1)}}{\longrightarrow} \stackrel{\tau_{a(v_1)}}{\longrightarrow} \stackrel{\tau_{b(dr)}}{\longrightarrow} \stackrel{\tau_{c_1(v_2)}}{\longrightarrow} \stackrel{\tau_{a(v_2)}}{\longrightarrow} \\
(ii) & \stackrel{\tau_{b(dr)}}{\longrightarrow} \stackrel{\tau_{c_1(v_1)}}{\longrightarrow} \stackrel{\tau_{a(v_1)}}{\longrightarrow} \stackrel{\tau_{out}}{\longrightarrow} \stackrel{\tau_{b(k)}}{\longrightarrow} \stackrel{\tau_{in}}{\longrightarrow} \stackrel{\tau_{b(w)}}{\longrightarrow} \stackrel{\tau_{c_2(v_2)}}{\longrightarrow} \stackrel{\tau_{a(v_2)}}{\longrightarrow}
\end{array}
$$

In the first sequence, *Doc* sends its location dr_K to S on port b, the server in response sends v_1 to Scr_1 on port c_1, and then *Doc* views v_1 via port a. He then waits for v_2 via the same port after sending his current location to S. These interactions are indicated by appropriate labels that annotate the τs of this sequence.

In the second case *Doc* sends its location dr_K to S on port b, the server in response sends v_1 to Scr_1 on port c_1, after viewing v_1 via port a *Doc* leaves the dr_K and enters the ward by its *out* dr_K and *in* w_L capabilities. It sends its current location to S on port b after executing every move capability. The server in response sends v_2 to the Scr_2 on port c_2, and then the screen displays v_2 to *Doc* on port a.

The possible interaction steps among the above given agents are as follows:

$\overline{b}(p)$: The agent *Doc* sends p as its current location to the server s via b. In this

particular case, $p = dr_K$.

$b(x)$: Server S is ready to receive a value at port b.

$\overline{c_1}(v).Server_n(l)$: The server S outputs a value v to a screen. Since we assumed $p = dr_K$, so S sends the value to Scr_1 via c_1, whereas $c_1(x_1)$ shows that Scr_1 is ready to receive the value on c_1.

$\overline{a}(x_1)$: Now Scr_1 displays the value at port a, and $a(x_1)$ shows that Doc is ready to view it at the same port.

Once Doc receives v, there are two possible sequences of execution of this setting. The choice operator "+" allows the agent Doc to either read the remaining information on the same screen or to leave dr_K by its capability $out\ dr_K$. As we have assumed earlier that Doc moves to the ward w_L and reads the remaining information on the screen there, so the interaction steps are:

$out\ p.\overline{b}(k)$: The ambient d_B leaves the doctor room dr_K by $out\ p$, where $p = dr_K$, and sends its new location k to S.

$in\ r.\overline{b}(w)$: Now d_B enters the ward w_L by its capability $in\ r$, where $r = w_L$ since $r \neq p$ and $p = dr_K$, and sends its new location w_L to S.

$\overline{c_2}(v).Server_n(l)$: Once the server receives w_L, it sends a value at c_2, whereas, $c_2(x_2).\overline{a}(x_2)$ shows that Scr_2 is ready to receive the value from S at c_2, and display via a.

$a(x_2)$: Finally, Doc reads the information via a.

The sequence (ii) of τ-transitions is as follows:

$\nu abc_1c_2(s[b(x).\ if\ (x = dr_K\ then\ \overline{c_1}(v_1).Server(v_2 : \epsilon)\ else\ if\ x = w_L\ then\ \overline{c_2}(v_2).$
$Server(\epsilon)\ else\ Server(v_1 : v_2 : \epsilon))] \mid k[dr_K[d_B[\overline{b}(dr_K).a(x_1).(\overline{b}(dr_K).a(x_2).$
$Doctor(x_2 : x_1 : \epsilon)\ + out\ dr_K.\overline{b}(k).in\ w_L.\overline{b}(w_L).a(x_3).Doctor(x_3 : x_1 : \epsilon))] \mid$
$scr_{A_1}[c_1(x_1).\overline{a}(x_1).Screen_1]\ ] \mid w_L[scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau}$

$\nu abc_1c_2(s[\ if\ (dr_K = dr_K\ then\ \overline{c_1}(v_1).Server(v_2 : \epsilon)\ else\ if\ dr_K = w_L\ then\ \overline{c_2}(v_2).$
$Server(\epsilon)\ else\ Server(v_1 : v_2 : \epsilon))] \mid k[dr_K[d_B[a(x_1).(\overline{b}(dr_K).a(x_2).$
$Doctor(x_2 : x_1 : \epsilon)\ + out\ dr_K.\overline{b}(k).in\ w_L.\overline{b}(w_L).a(x_3).Doctor(x_3 : x_1 : \epsilon))] \mid$
$scr_{A_1}[c_1(x_1).\overline{a}(x_1).Screen_1]\ ] \mid w_L[scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau} \equiv$

$\nu abc_1c_2(s[b(x).\ if\ (x = dr_K\ then\ \overline{c_1}(v_2).Server(\epsilon)\ else\ if\ x = w_L\ then\ \overline{c_2}(v_2).$
$Server(\epsilon)\ else\ Server(v_2 : \epsilon))] \mid k[dr_K[d_B[a(x_1).(\overline{b}(dr_K).a(x_2).Doctor(x_2 : x_1 : \epsilon)$
$+ out\ dr_K.\overline{b}(k).in\ w_L.\overline{b}(w_L).a(x_3).Doctor(x_3 : x_1 : \epsilon))] \mid scr_{A_1}[\overline{a}(v_1).Screen_1]\ ] \mid$
$w_L[scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau}$

$\nu abc_1c_2(s[b(x).\ if\ (x = dr_K\ then\ \overline{c_1}(v_2).Server(\epsilon)\ else\ if\ x = w_L\ then\ \overline{c_2}(v_2).$
$Server(\epsilon)\ else\ Server(v_2 : \epsilon))] \mid k[dr_K[d_B[(\overline{b}(dr_K).a(x_2).Doctor(x_2 : v_1 : \epsilon)$
$+ out\ dr_K.\overline{b}(k).in\ w_L.\overline{b}(w_L).a(x_3).Doctor(x_3 : v_1 : \epsilon))] \mid scr_{A_1}[Screen_1]\ ] \mid$
$w_L[scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ])$

Here, *Doc* has received v_1, he then next leaves dr_K by *out* dr_K and enters w_L by *in* w_L and receives v_2 via Scr_2 there. We show all possible transitions in Figure 5.4 and define agents' states in Table 5.9 in terms of CCS expressions. We show the resulting transitions where *Doc* moves towards w_L and continues reading the

remaining information as follows:

$\nu abc_1c_2(s[b(x).\ if\ (x = dr_K\ then\ \overline{c_1}(v_2).Server(\epsilon)\ else\ if\ x = w_L\ then\ \overline{c_2}(v_2).$
$Server(\epsilon)\ else\ Server(v_2 : \epsilon))] \mid k[dr_K[d_B[(\overline{b}(dr_K).a(x_2).Doctor(x_2 : v_1 : \epsilon)$
$+\ out\ dr_K.\overline{b}(k).in\ w_L.\overline{b}(w_L).a(x_3).Doctor(x_3 : v_1 : \epsilon))] \mid scr_{A_1}[Screen_1]\ ] \mid$
$w_L[scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau}$

$\nu abc_1c_2(s[b(x).\ if\ (x = dr_K\ then\ \overline{c_1}(v_2).Server(\epsilon)\ else\ if\ x = w_L\ then\ \overline{c_2}(v_2).$
$Server(\epsilon)\ else\ Server(v_2 : \epsilon))] \mid k[dr_K[scr_{A_1}[Screen_1]\ ] \mid d_B[\overline{b}(k).in\ w_L.$
$\overline{b}(w_L).a(x_3).Doctor(x_3 : v_1 : \epsilon))] \mid w_L[scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau}$

$\nu abc_1c_2(s[\ if\ (k = dr_K\ then\ \overline{c_1}(v_2).Server(\epsilon)\ else\ if\ k = w_L\ then\ \overline{c_2}(v_2).Server(\epsilon)$
$else\ Server(v_2 : \epsilon))] \mid k[dr_K[scr_{A_1}[Screen_1]\ ] \mid d_B[in\ w_L.\overline{b}(w_L).a(x_3).$
$Doctor(x_3 : v_1 : \epsilon))] \mid w_L[scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau} \equiv$

$\nu abc_1c_2(s[b(x).\ if\ (x = dr_K\ then\ \overline{c_1}(v_2).Server(\epsilon)\ else\ if\ x = w_L\ then\ \overline{c_2}(v_2).$
$Server(\epsilon)\ else\ Server(v_2 : \epsilon))] \mid k[dr_K[scr_{A_1}[Screen_1]\ ] \mid w_L[d_B[\overline{b}(w_L).a(x_3).$
$Doctor(x_3 : v_1 : \epsilon))] \mid scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau}$

$\nu abc_1c_2(s[\ if\ (w_L = dr_K\ then\ \overline{c_1}(v_2).Server(\epsilon)\ else\ if\ w_L = w_L\ then\ \overline{c_2}(v_2).Server(\epsilon)$
$else\ Server(v_2 : \epsilon))] \mid k[dr_K[scr_{A_1}[Screen_1]\ ] \mid w_L[d_B[a(x_3).Doctor(x_3 : v_1 : \epsilon))]$
$\mid scr_{A_2}[c_2(x_2).\overline{a}(x_2).Screen_2]\ ]\ ]) \xrightarrow{\tau}$

$\nu abc_1c_2(s[Server(\epsilon)] \mid k[dr_K[scr_{A_1}[Screen_1]\ ] \mid w_L[d_B[a(x_3).Doctor(x_3 : v_1 : \epsilon))]$
$\mid scr_{A_2}[\overline{a}(v_2).Screen_2]\ ]\ ]) \xrightarrow{\tau}$

$\nu abc_1c_2(s[Server(\epsilon)] \mid k[dr_K[scr_{A_1}[Screen_1]\ ] \mid w_L[d_B[Doctor(v_2 : v_1 : \epsilon))]$
$\mid scr_{A_2}[Screen_2]\ ]\ ]).$

The CCS expressions representing various states of the four agents, the doctor, the server and the two screens, are defined in Table 5.9.

$$
\begin{array}{lcl}
Doctor(\epsilon) & \stackrel{def}{=} & \overline{b}(dr).Doctor1(\epsilon) \\
Doctor1(\epsilon) & \stackrel{def}{=} & a(v_1).Doctor2(v_1 : \epsilon) \\
Doctor2(v_1 : \epsilon) & \stackrel{def}{=} & \overline{b}(dr).Doctor8(v_1 : \epsilon) + out\ dr.Doctor3(v_1 : \epsilon) \\
Doctor8(v_1 : \epsilon) & \stackrel{def}{=} & a(v_2).Doctor7'(v_2 : v_1 : \epsilon) \\
Doctor3(v_1 : \epsilon) & \stackrel{def}{=} & \overline{b}(k).Doctor4(v_1 : \epsilon) \\
Doctor4(v_1 : \epsilon) & \stackrel{def}{=} & in\ w.Doctor5(v_1 : \epsilon) \\
Doctor5(v_1 : \epsilon) & \stackrel{def}{=} & \overline{b}(w).Doctor6(v_1 : \epsilon) \\
Doctor6(v_1 : \epsilon) & \stackrel{def}{=} & a(v_2).Doctor7(v_2 : v_1 : \epsilon)
\end{array}
$$

$$
\begin{array}{lcl}
Screen_1 & \stackrel{def}{=} & c_1(x).Screen'_1 \\
Screen'_1 & \stackrel{def}{=} & \overline{a}(x).Screen_1 \\
Screen_2 & \stackrel{def}{=} & c_2(x).Screen'_2 \\
Screen'_2 & \stackrel{def}{=} & \overline{a}(x).Screen_2
\end{array}
$$

$$
\begin{array}{lcl}
S(v_1 : v_2 : \epsilon) & \stackrel{def}{=} & b(dr).S1(v_1 : v_2 : \epsilon) \\
S1(v_1 : v_2 : \epsilon) & \stackrel{def}{=} & \overline{c_1}(v_1).S2(v_2 : \epsilon) \\
S2(v_2 : \epsilon) & \stackrel{def}{=} & b(dr).S3(v_2 : \epsilon) + b(k).S3'(v_2 : \epsilon) \\
S3(v_2 : \epsilon) & \stackrel{def}{=} & \overline{c_1}(v_2).S4(\epsilon) \\
S3'(v_2 : \epsilon) & \stackrel{def}{=} & b(w).S3''(v_2 : \epsilon) \\
S3''(v_2 : \epsilon) & \stackrel{def}{=} & \overline{c_2}(v_2).S4'(\epsilon)
\end{array}
$$

Table 5.9: CCS expressions for doctor, server and screens

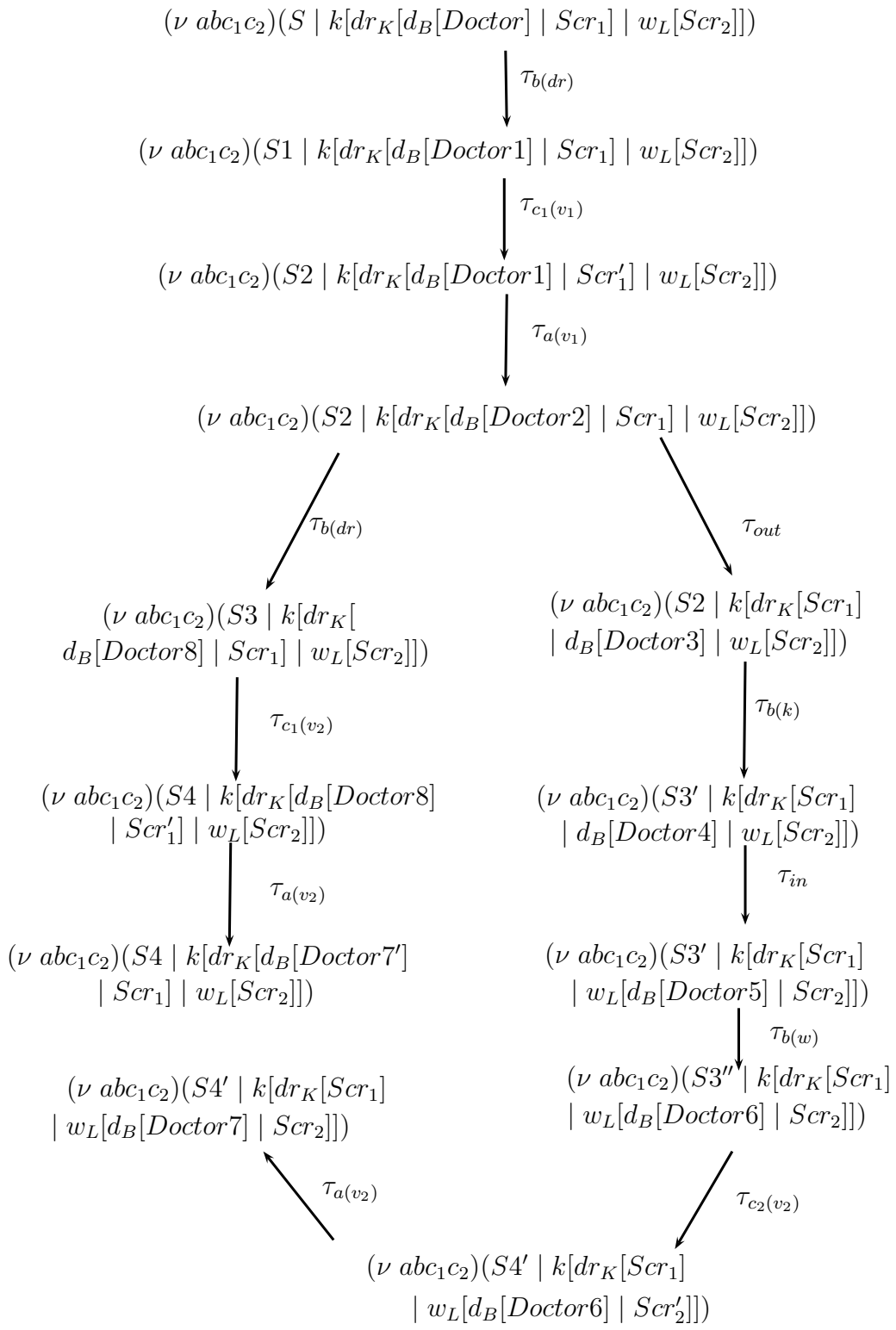


Figure 5.4: Services follow doctor: transition diagram

This case study clearly illustrates the expressiveness of the calculus in the given problem domain, where the primitives for mobility and communication are quite relevant. Agents' mobility and global communication features are modelled in a scenario where services follow mobile ambients, and server supplies services globally

to appropriate device provided that the receiving ambient is at the same location as the device.

Initially, the agent *Doc* globally communicates with S and then S interacts globally with Scr_1 at location dr_K. The agent *Doc* receives one part of the information via a in the doctor room dr_K on screen scr_{A_1}. Here, we intuitively move *Doc* from dr_K to the ward w_L to show the continuation of the services from the point it was left previously. Now S sends the remaining part of information to *Doc* on Scr_2 which is inside w_L. The transition system representing the above scenario is given in Figure 5.4.

5.4 Behavioural Semantics

In this section we develop an appropriate notion of behavioural equivalence for CMC. All processes and context mentioned in this section are from our calculus CMC. We formulate the equivalence in terms of α-transitions ($\xrightarrow{\alpha}$), for $\alpha \in a(z), \overline{b}(z), in\ m_A, out\ m_A, \tau$, as usual, for all a, b, m, A, and observation predicate as in [11, 37]. In MA observation predicate is used to detect the presence of top-level ambient. We write $P \downarrow_{n_A}$ to denote the presence of ambient n_A at the top level, in the other words process P may interact with the environment via n_A. We write $P \Downarrow_{n_A}$, if after some number of τ-transitions, the process P exhibits n_A at the top level. The two predicates are defined as follows:

$$P \downarrow_{n_A} \stackrel{def}{=} P \equiv \nu\tilde{m}(n_A[P_1] \mid P_2), \text{ where } n_A \notin \tilde{m} \text{ for some} P_1, P_2$$

$$P \Downarrow_{n_A} \stackrel{def}{=} P \stackrel{\hat{\tau}}{\Rightarrow} Q \text{ and } Q \downarrow_{n_A} \text{ for some } Q$$

A relation R over process P, Q is barb preserving if it is preserved by observation predicates, namely if P may interact with environment via ambient n_A then Q may also interact via the ambient n_A after a number of τ-transitions. Observation predicates of the two process P, Q are invariant under any contexts $\mathcal{C}[\cdot]$.

Definition 5.5. (Barb Preserving)
A relation R over processes is said to be *barb preserving* if $P\ R\ Q$ and $P \downarrow_{n_A}$ implies $Q \Downarrow_{n_A}$.

Definition 5.6. (Context)
A *context* $\mathcal{C}[\cdot]$ is a process with zero or more holes $[\cdot]$. A hole $[\cdot]$ in a context $\mathcal{C}$ is replaced by at most one occurrence of a process. A context $\mathcal{C}[\cdot]$ with a hole $[\cdot]$ replaced by a process P is denoted by $\mathcal{C}[P]$.

Definition 5.7. (Contextual Equivalence)
Processes P, Q are *contextual equivalent*, denoted by $P \simeq Q$, if for all contexts $\mathcal{C}[\cdot]$ and ambient names n_A, $\mathcal{C}[P] \downarrow_{n_A}$ implies $\mathcal{C}[Q] \Downarrow_{n_A}$.

Since we are considering weak equivalence we provide the notion of weak actions as follows. We write $\alpha \in Act$ (recall that $\tau \in Act$). We write $\Rightarrow$ for the reflexive and transitive closure of $\xrightarrow{\tau}$, where $\xrightarrow{\tau}$ specifies exactly the τ-transition. $\stackrel{\tau}{\Rightarrow}$ specifies at least a τ transition. $\hat{\alpha}$ is a sequence obtained by deleting all occurrences of τ actions, note that $\hat{\tau} = \epsilon$. Furthermore, $\stackrel{\hat{\tau}}{\Rightarrow}$ is $\stackrel{\epsilon}{\Rightarrow}$, an empty sequence of τ-transitions, and $\stackrel{\hat{\alpha}}{\Rightarrow}$ is $\stackrel{\alpha}{\Rightarrow}$, for $\alpha \neq \tau$.

We define two forms of barbs; one at ambient level whereas another for ambients capabilities. They give rise to (a) barbed bisimulation and congruence, and (b) capability barbed bisimulation and congruence. We then show that the respective congruence relations imply each other. Two processes are barbed congruent if when they are placed into any context then the context processes are barbed bisimilar.

Definition 5.8. (Barbed Bisimulation and Congruence)
A relation S is a *barbed bisimulation*, if it is symmetric and if $(P, Q) \in S$ then for all $\alpha \in \{a(z), \overline{b}(z), in\ m_A, out\ m_A\}$,

- if $P \xrightarrow{\alpha} P'$ then $Q \stackrel{\widehat{\alpha}}{\Rightarrow} Q'$ and $(P', Q') \in S$;

- if $P \downarrow_{n_A}$ then $Q \Downarrow_{n_A}$.

Processes P and Q are barbed bisimilar, $P \approx Q$, if $(P, Q) \in S$ for some barbed bisimulation S. P and Q are barbed congruent, $P \cong Q$, if for all contexts $\mathcal{C}[\cdot]$, $\mathcal{C}[P] \approx \mathcal{C}[Q]$.

We now show a barbed bisimulation relation between CMC processes. Consider, for example two agents

$$P \stackrel{def}{=} m_A[n_B[out\ m_A.0 \mid in\ k_C.0]] \mid k_C[0]$$
$$Q \stackrel{def}{=} m_A[n_B[out\ m_A.in\ k_C.0] \mid 0] \mid k_C[0]$$

We show that the agents are equivalent according to barbed bisimulation. We will construct a barbed bisimulation S by checking Definition 5.8. There is only one possible sequence of transitions for each agent:

$m_A[n_B[out\ m_A.0 \mid in\ k_C.0]] \mid k_C[0] \xrightarrow{\tau} m_A[0] \mid n_B[in\ k_C.0] \mid k_C[0] \xrightarrow{\tau}$
$m_A[0] \mid k_C[n_B[0] \mid 0]$.

$m_A[n_B[out\ m_A.in\ k_C.0] \mid 0] \mid k_C[0] \xrightarrow{\tau} m_A[0] \mid n_B[in\ k_C.0] \mid k_C[0] \xrightarrow{\tau}$

$m_A[0] \mid k_C[n_B[0] \mid 0]$.

The first τ-transitions of both P and Q arise for the capability *out* m_A. This is the only transition that P can start with, whereas Q can start with the same transition. This is followed by the τ-transitions that that arise for the capability *in* k_C. Since the sequence of transitions of P and Q matches each other, therefore it is verified that the two agents are equivalent. We obtain S as follows:

$S = \{(m_A[n_B[out\ m_A.0 \mid in\ k_C.0]] \mid k_C[0], m_A[n_B[out\ m_A.in\ k_C.0] \mid 0] \mid k_C[0]), (m_A[0] \mid n_B[in\ k_C.0] \mid k_C[0], m_A[0] \mid n_B[in\ k_C.0] \mid k_C[0]), (m_A[0] \mid k_C[n_B[0] \mid 0], m_A[0] \mid k_C[n_B[0] \mid 0])\}$

which shows that the two agents P and Q are barbed bisimilar. At various stages of checking barbed bisimulation, we observe that the barbs also match.

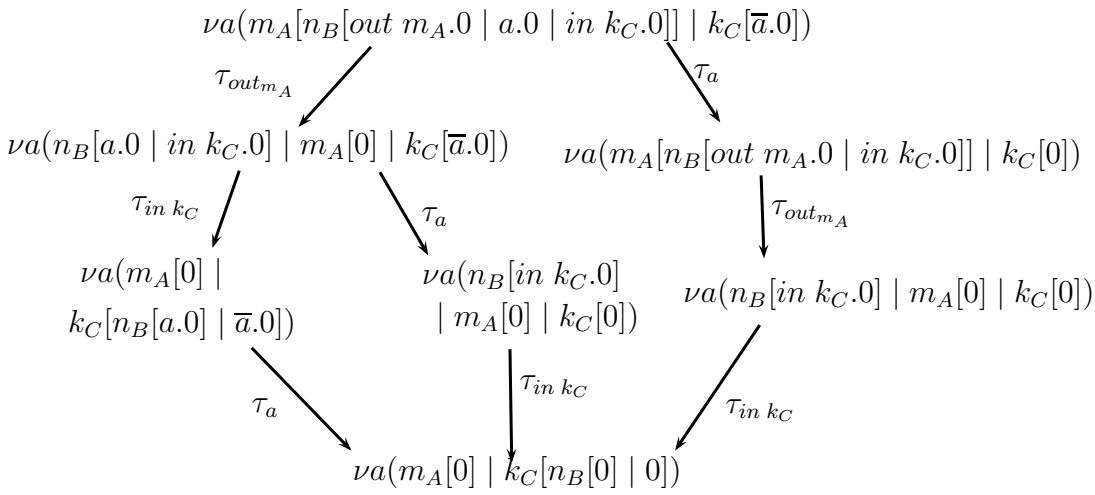
We now show the equivalence between two mobile agents that may communicate globally. Consider, agents P and Q defined below.

$$P \stackrel{def}{=} \nu a(m_A[n_B[out\ m_A.0 \mid a.0 \mid in\ k_C.0]] \mid k_C[\overline{a}.0]), \text{ where } a \in A, B, C.$$

$$Q \stackrel{def}{=} \nu a(m_A[n_B[out\ m_A.in\ k_C.0] \mid a.0] \mid k_C[\overline{a}.0]), \text{ for } a \in A, B, C.$$

To help the reader, we also present transition graphs for P and Q in Figure 5.5. The transition graphs show that whatever transitions the agent P performs the corresponding transitions of agent Q match them, and correspondingly if the agents are swapped. Similarly, at various stages of checking, the barbs also match.

$P \stackrel{def}{=} \nu a(m_A[n_B[out\ m_A.0 \mid a.0 \mid in\ k_C.0]] \mid k_C[\overline{a}.0]),$ where $a \in A, B, C$



$Q \stackrel{def}{=} \nu a(m_A[n_B[out\ m_A.in\ k_C.0] \mid a.0] \mid k_C[\overline{a}.0]),$ where $a \in A, B, C.$

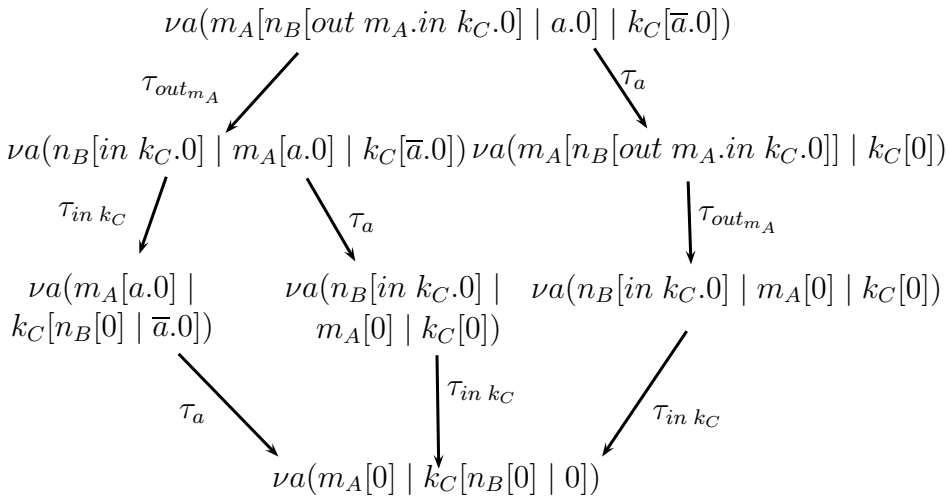


Figure 5.5: Transition graphs

Furthermore, we consider two agents

$a(m_A[n_B[out\ m_A.0 \mid a.0 \mid in\ k_C.0]] \mid k_C[\overline{a}.0]),$ for $a \in B, C$ and $a \notin A,$ and

$\nu a(m_A[n_B[out\ m_A.in\ k_C.0] \mid a.0] \mid k_C[\overline{a}.0]),$ for $a \in B, C$ and $a \notin A$

and show that they are not equivalent. The first possible τ-transitions of both P and Q are

$$a(m_A[n_B[out\ m_A.0 \mid a.0 \mid in\ k_C.0]] \mid k_C[\overline{a}.0]) \stackrel{\tau}{\rightarrow} \nu a(n_B[a.0 \mid in\ k_C.0] \mid m_A[0] \mid k_C[\overline{a}.0]),$$

$$\nu a(m_A[n_B[out\ m_A.in\ k_C.0] \mid a.0] \mid k_C[\overline{a}.0]) \stackrel{\tau}{\rightarrow} \nu a(n_B[in\ k_C.0] \mid m_A[a.0] \mid k_C[\overline{a}.0])$$

where τ-actions arise for the capability *out* m_A.
Then next, the resulting term $\nu a(n_B[a.0 \mid in\ k_C.0] \mid m_A[0] \mid k_C[\overline{a}.0])$ of the first agent may execute as follows.

$$\nu a(n_B[a.0 \mid in\ k_C.0] \mid m_A[0] \mid k_C[\overline{a}.0]) \xrightarrow{\tau_a} \nu a(n_B[in\ k_C.0] \mid m_A[0] \mid k_C[0])$$

where τ_a action corresponds to communication between n_B and k_C via a. The resulting term $\nu a(n_B[in\ k_C.0] \mid m_A[a.0] \mid k_C[\overline{a}.0])$ of the second agent can not match this transition, since $a \notin A$ which does not allow m_A of the second agent to communicate via a. This is shown as below:

$$\nu a(n_B[in\ k_C.0] \mid m_A[a.0] \mid k_C[\overline{a}.0]) \xrightarrow{\tau_a} .$$

Definition 5.9. We write $P \downarrow_\beta$ if $P \xrightarrow{\beta} P'$ for some P', where $\beta \in \{in\ n_A, out\ n_A, enter\ n_A, move\ n_A, exit\ n_A\}$. We write $P \Downarrow_\beta$ if $P \xrightarrow{\tau^*} P' \xrightarrow{\beta} P''$ for some P' and P''.

We now define β-barb bisimulation, where barb congruence between two process remains invariant when they are placed into any context.

Definition 5.10. (Capability Barbed Bisimulation)
Let $L = \{in\ n_A, out\ n_A, enter\ n_A, move\ n_A, exit\ n_A\}$, and let $\beta \in L$. A relation R is a *β-barbed bisimulation*, if R is symmetric and if $(P, Q) \in R$ then for all $\alpha \in \{a(z), \overline{b}(z), in\ n_A, out\ n_A\}$:

- if $P \xrightarrow{\alpha} P'$ then $Q \stackrel{\hat{\alpha}}{\Rightarrow} Q'$ and $(P', Q') \in R$;

- if $P \downarrow_\beta$ then $Q \Downarrow_\beta$.

P and Q are β-barbed bisimilar, $P \approx_\beta Q$, if $(P, Q) \in R$ for some β-barbed bisimulation R. P and Q are barbed congruent, $P \cong_\beta Q$, if for all contexts $\mathcal{C}[\cdot]$, $\mathcal{C}[P] \approx_\beta \mathcal{C}[Q]$.

A well-known result that comes from [37] is given in Lemma 5.1.

Lemma 5.1. *If $P \cong_\beta Q$ then*

1. *$P \Downarrow_{n_A}$ iff $Q \Downarrow_{n_A}$*
2. *$P \stackrel{\hat{\tau}}{\Rightarrow} P'$ implies that there is Q such that $Q \stackrel{\hat{\tau}}{\Rightarrow} Q'$ and $P' \cong_\beta Q'$.*

In preparation for the main result given in Theorem 5.1, Lemmas 5.2 and 5.3 are required.

Lemma 5.2. *For m_A and k_C fresh in an agent R, $R \Downarrow_{move\ n_B}$ iff $\mathcal{C}_1[R] \Downarrow_{m_A}$.*

Proof. We prove the left to right implication first:

$$R \Downarrow_{move\ n_B} \text{ implies } \mathcal{C}_1[R] \Downarrow_{m_A}$$

By Definition 5.9, $R \Downarrow_{move\ n_B}$ iff $R \xrightarrow{\tau^*} R' \xrightarrow{move\ n_B} R''$ for some R', R''. Since $R \Downarrow_{move\ n_B}$ is valid, we obtain $R \xrightarrow{\tau*} R' \xrightarrow{move\ n_B} R''$.

We now consider $R' \xrightarrow{move\ n_B} R''$. By part 3 of Lemma 4.1, if $R' \xrightarrow{move\ n_B} \nu\tilde{r}\langle Q'\rangle Q''$ then $R' \equiv \nu\tilde{r}(n_B[R_1] \mid R_2)$ and $R'' \equiv \nu\tilde{r}\langle Q'\rangle Q''$, where $Q' \equiv R_1$ and $Q'' \equiv R_2$. We now have,

$$\begin{aligned}\mathcal{C}_1[R'] \;&\equiv \mathcal{C}_1[\nu\tilde{r}(n_B[R_1] \mid R_2)] \equiv \\ &\quad \nu m_A(\nu\tilde{r}(n_B[R_1] \mid R_2)) \mid \nu a(k_C[in\ n_B.out\ n_B.\overline{a}.0] \mid a.m_A[P])\end{aligned}$$

Since by $(*)$ in τ-In, the members of $\tilde{r}$ are not free names in $\nu a(k_C[in\ n_B.out\ n_B.\overline{a}.0] \mid a.m_A[P])$, and $a \notin fn(\nu m_A(\nu\tilde{r}(n_B[R_1] \mid R_2)))$, the process $\mathcal{C}_1[\nu\tilde{r}(n_B[R_1] \mid R_2)]$ executes as follows

$$\begin{aligned}
&\xrightarrow{\tau} \quad \nu a\nu\tilde{r}(\nu m_A(n_B[k_C[out\ n_B.\overline{a}.0] \mid R_1] \mid R_2) \mid a.m_A[P]), \\
&\qquad (k_C \neq m_A \text{ and } k_C \notin \tilde{r}) \text{ and } (a \notin fn(R_2) \text{ and } \tilde{r} \cap fn(P) = \emptyset) && (\tau\text{-In}) \\
&\xrightarrow{\tau} \quad \nu a\nu\tilde{r}(\nu m_A(n_B[R_1] \mid R_2 \mid k_C[\overline{a}.0]) \mid a.m_A[P]) && (\tau\text{-Out}) \\
&\xrightarrow{\tau} \quad \nu a\nu\tilde{r}(\nu m_A(n_B[R_1] \mid R_2 \mid k_C[0]) \mid m_A[P]) && (\text{Global-Com})
\end{aligned}$$

We need to show $\mathcal{C}_1[R] \Downarrow_{m_A}$ which by our predicate definition, means $\mathcal{C}_1[R] \xrightarrow{\tau^*} \mathcal{C}_1[R'] \downarrow_{m_A}$, and $\mathcal{C}_1[R'] \downarrow_{m_A}$ means $\mathcal{C}_1[R'] \equiv \nu\tilde{m}(m_A[P_1] \mid P_2)$ for some $P_1, P_2, \tilde{m}$. When $P_2 \equiv \nu m_A(n_B[R_1] \mid R_2 \mid k_C[0])$, $m_A[P_1] \equiv m_A[P]$ and $\tilde{m} \equiv \nu a\nu\tilde{r}$, then we obtain $\mathcal{C}_1[R'] \equiv \nu a\nu\tilde{r}(\nu m_A(n_B[R_1] \mid R_2 \mid k_C[0]) \mid m_A[P])$, which implies $\mathcal{C}_1[R'] \downarrow_{m_A}$. Since $R \xrightarrow{\tau^*} R'$ we obtain $\mathcal{C}_1[R] \xrightarrow{\tau^*} \mathcal{C}_1[R']$. Since $\mathcal{C}_1[R] \xrightarrow{\tau^*} \mathcal{C}_1[R']$ and $\mathcal{C}_1[R'] \downarrow_{m_A}$, we obtain $\mathcal{C}_1[R] \Downarrow_{m_A}$ as required.

We now show the right to left implication of Lemma 5.2, namely

$$\mathcal{C}_1[R] \Downarrow_{m_A} \text{ implies } R \Downarrow_{move\ n_B}$$

Since $\mathcal{C}_1[R] \Downarrow_{m_A}$ means $\mathcal{C}_1[R] \xrightarrow{\tau^*} \mathcal{C}_1[R'] \downarrow_{m_A}$ for some R', we have

$$\mathcal{C}_1[R] \equiv \nu m_A(R) \mid \nu a(k_C[in\ n_B.out\ n_B.\overline{a}.0] \mid a.m_A[P])$$

Here, $\mathcal{C}_1[R]$ may interact with the environment via the ambient m_A only if, after some τ-transitions, m_A exists at the top level. To bring m_A at the top level the

process R must contain n_B, so $R \xrightarrow{\tau*} R' \downarrow_{n_B}$ and we obtain

$$\nu m_A(R') \mid \nu a(k_C[in\ n_B.out\ n_B.\overline{a}.0] \mid a.m_A[P])$$

Since we define predicate $(R' \downarrow_{n_B})$ as $R' \downarrow_{n_B} \stackrel{def}{=} R' \equiv \nu\tilde{q}(n_B[Q_1] \mid Q_2)$ for some Q_1, Q_2, and $n_B \notin \tilde{q}$, we obtain

$\nu m_A(R') \mid \nu a(k_C[in\ n_B.out\ n_B.\overline{a}.0] \mid a.m_A[P]) \xrightarrow{\tau}\xrightarrow{\tau}$

$\nu m_A(R') \mid \nu a(k_C[\overline{a}.0] \mid a.m_A[P]) \xrightarrow{\tau}$

$\nu m_A(R') \mid \nu a(k_C[0] \mid m_A[P])$.

Since after a number of τ-transitions we have m_A at the top level of context $\mathcal{C}_1$, so $\mathcal{C}_1[R']$ may interact with environment via m_A and we obtain $\mathcal{C}_1[R'] \downarrow_{m_A}$.

Since $\mathcal{C}_1[R] \xrightarrow{\tau^*} \mathcal{C}_1[R']$ and $\mathcal{C}_1[R'] \downarrow_{m_A}$, we obtain $\mathcal{C}_1[R] \Downarrow_{m_A}$.

Since we have $R' \equiv \nu\tilde{q}(n_B[Q_1] \mid Q_2)$, we show $\xrightarrow{move\ n_B}$ as follows:

$$\dfrac{\dfrac{\dfrac{\text{Co-Enter}\ \dfrac{}{n_B[Q_1] \xrightarrow{move\ n_B} \langle n_B[Q_1]\rangle 0}}{\lambda\text{-Par}\ \dfrac{}{n_B[Q_1] \mid Q_2 \xrightarrow{move\ n_B} \langle n_B[Q_1]\rangle(0 \mid Q_2)}}}{\lambda\text{-Res}\ \dfrac{}{\nu\tilde{q}(n_B[Q_1] \mid Q_2) \xrightarrow{move\ n_B} \nu\tilde{q}\langle n_B[Q_1]\rangle(0 \mid Q_2) \equiv \nu\tilde{q}\langle n_B[Q_1]\rangle Q_2}\ fn(Q) \notin \tilde{q}}}{\text{Struct}\ \dfrac{}{\nu\tilde{q}(n_B[Q_1] \mid Q_2) \xrightarrow{move\ n_B} \nu\tilde{q}\langle n_B[Q_1]\rangle Q_2}}$$

□

Lemma 5.3. *For k_C and n_B fresh in an agent R, $R \Downarrow_{m_A}$ iff $\mathcal{C}_1[R] \Downarrow_{move\ n_B}$.*

Proof. Since the proof is very similar to the proof of Lemma 5.2 it is omitted. □

We now prove that two congruence relations, namely barbed bisimulation congruence and capability barbed bisimulation congruence for $\beta = move\ n_B$, imply each other. Our definition of barbed bisimulation and congruence remains unchanged for β-barb.

Theorem 5.1. *Let $P, Q \in CMC$. Then, $P \cong Q$ iff $P \cong_{move\ n_B} Q$ for all n_B.*

Proof. The only difference between the two forms of the barbs is the level at which each barb is defined, namely the definition at ambient level and the definition at the capability level. We show that the two forms of barbs imply each other.

Firstly, we show that $P \cong Q$ implies $P \cong_{move\ n_B} Q$ for all P, Q and n_B.

Assume that $P \cong Q$ and $P \Downarrow_{move\ n_B}$, and we will show $Q \Downarrow_{move\ n_B}$.

We define a context $\mathcal{C}_1[\cdot]$ as follows:

$\mathcal{C}_1[\cdot] \stackrel{def}{=} \nu m_A([\cdot]) \mid \nu a(k_C[in\ n_B.out\ n_B.\overline{a}.0] \mid a.m_A[P])$, with $a \notin B$ and $a \in C$

This context allows the ambient m_A to interact with the environment after the communication on a has happened. Also, k_C may execute its *in* n_B and *out* n_B capabilities only if n_B exists in parallel with k_C. Therefore, any process replacing the context hole $[\cdot]$ must contain n_B. Then k_C, after executing its capabilities, may communicate on a, which enables the context $\mathcal{C}_1\ [\cdot]$ to interact with the environment via m_A.

Global communication is very useful in the definition of context $\mathcal{C}_1[\cdot]$. It acts as a guard and the context may interact with the environment via corresponding guarded ambient if the guard is satisfied.

Since $P \Downarrow_{move\ n_B}$ we get, by Lemma 5.2, $\mathcal{C}_1[P] \Downarrow_{m_A}$. Since $P \cong Q$, we obtain $\mathcal{C}_1[P] \cong \mathcal{C}_1[Q]$, for context $\mathcal{C}_1[\cdot]$. Then since $\mathcal{C}_1[P] \cong \mathcal{C}_1[Q]$, $\mathcal{C}_1[P] \Downarrow_{m_A}$ gives us $\mathcal{C}_1[Q] \Downarrow_{m_A}$. Finally, by Lemma 5.2, $\mathcal{C}_1[Q] \Downarrow_{m_A}$ implies $Q \Downarrow_{move\ n_B}$ as required.

Next, we show the right to left implication, namely

$$P \cong_{move\ n_B} Q \Rightarrow P \cong Q \text{ for all } P, Q.$$

Assume that $P \cong_{move\ n_B} Q$ and $P \downarrow_{m_A}$, and we will show $Q \Downarrow_{m_A}$.

We define the context $\mathcal{C}_2[\cdot]$ as follows:
$\mathcal{C}_2[\cdot] \stackrel{def}{=} \nu n_B([\cdot]) \mid \nu a(k_C[in\ m_A.out\ m_A.\overline{a}.0] \mid a.n_B[P])$, with $a \notin A$ and $a \in C$.

Since $P \Downarrow_{m_A}$ Lemma 5.3 gives us $\mathcal{C}_2[P] \Downarrow_{move\ n_B}$. Since $P \cong_{move\ n_B} Q$, we obtain $\mathcal{C}_2[P] \cong_{move\ n_B} \mathcal{C}_2[Q]$ for context $\mathcal{C}_2[\cdot]$. Next, since $\mathcal{C}_2[P] \cong_{move\ n_B} \mathcal{C}_2[Q]$, $\mathcal{C}_2[P] \Downarrow_{move\ n_B}$ gives us $\mathcal{C}_2[Q] \Downarrow_{move\ n_B}$. Hence, by Lemma 5.3, $\mathcal{C}_2[Q] \Downarrow_{move\ n_B}$ implies $Q \Downarrow_{m_A}$ as required. □

Conjecture 5.1. We conjecture that Theorem 5.1 will hold for the other capabilities, namely *enter* n_B and *exit* n_B of CMC.

The notion of behavioural equivalence and the proof method for establishing the equivalence is inspired by that in [36, 37]. The authors in [36, 37], use co-actions and passwords that help them in proving their results, whereas the use of global communication is fundamental in proving the above results.

5.5 Conclusion

We have presented CMC for the modelling of mobility and communication in the setting of ubiquitous computing. The notion of ambients mobility has been modelled in CMC by the *in* n_B and *out* n_B capabilities [11]. A new form of global communication has been introduced in CMC which is similar to that in Milner's CCS. Ambient's name has been tagged with the set of ports which are functioning as a

restriction on global communication, specified at the level of ambients. A labelled transition system semantics has been developed for CMC, where $P \xrightarrow{\tau} Q$ represents not only a binary communication of processes as in CCS but also the ambients' mobility steps by means of their *in* n_B and *out* n_B capabilities. This has been achieved by additional labels and specialised transitions from processes to the so-called outcomes which are either processes or concretions. We have illustrated the usefulness of the calculus by presenting *path between two locations* and *intelligent hospital* case studies. New forms of behavioural equivalences for CMC has been introduced. We have defined barbed bisimulation and congruence, and capability barbed bisimulation and congruence and have showed that the respective congruence relations of the two forms of barbs coincide. In the following two chapters we extend our calculus by adding passive mobility features and context-awareness mechanism.

Chapter 6

Operational Semantics for Push and Pull Ambient Calculus

In this chapter we extend CMC by introducing passive mobility feature to it. We add additional mobility primitives, namely the *push* and *pull* capabilities of Push and Pull Ambient Calculus (PAC), by Phillips and Vigliotti [54, 77]. The basic idea of PAC relies on ambients' $push_m\ n$ and $pull_m\ n$ capabilities instead of the *in* and *out* capabilities of Cardelli and Gordon's Mobile Ambients. We define a new transition operational semantics, and the first such operational semantics to the best of our knowledge, for the *push* and *pull* capabilities. We prove that the new operational semantics coincides with the standard reduction semantics. The usefulness of the extended calculus is illustrated by modelling example scenarios.

In the ubiquitous computing setting it is beneficial to consider both active and passive mobile structures. Active mobile structures are those that could move on their own, while passive mobile structures may only move around when active structures carry them. In this chapter we aim at modelling both active and passive mobile structures.

We now show the usefulness of passive mobility primitives with the help of an example. Consider a setting, where $a \notin A$ for some A,

$$user_A[a(x).P'_u \mid P''_u] \mid device[\overline{a}(v).P_d]$$

ambient *device* wants to send a value to ambient $user_A$ via port a. Ambient $user_A$ is unable to receive the value on a since $a \notin A$. In this case $user_A$ can communicate with *device* if *device* is inside the scope of $user_A$. The behaviour of passive mobile structures is easily expressed in PAC. By PAC syntax an agent can easily push a non-useful child out from its scope. Similarly, an agent can pull any other sibling

inside whenever needed. We use $push(m_A)\ n_B$ and $pull(m_A)\ n_B$ capabilities, where $A, B \subseteq \mathcal{L}$, to move passive ambients around. The capability $push(m_A)\ n_B$ allows an ambient m_A to push an ambient n_B out of its boundary, whereas $pull(m_A)\ n_B$ allows m_A to pull in n_B into its scope. Therefore, to make communication possible we modify our setting as follows:

$$user_A[a(x).P'_u \mid pull(user_A)\ device.P''_u] \mid device[\overline{a}(v).P_d]$$

Now, the capability $pull(user_A)\ device$ enables the user to pull the device into its scope and a communication may take place.

This chapter is organised as follows: We introduce the extended calculus in Section 6.1, where we add new mobility primitives to model passive mobile structures. In Section 6.2 reduction semantics for *push* and *pull* is given, which is followed by their operational semantics. We then show usefulness of the calculus with small examples in Section 6.3. In Section 6.4 we show the correspondence between the two semantics and Section 6.5 contains conclusions.

6.1 The Syntax of CMC_P

In Chapter 5 the operational semantics of CMC has been defined in terms of ambients entering or exiting other ambients, whereas we now aim to use the capabilities of PAC to formulate the behaviour of passive mobile structures. We extend CMC with the *push* and *pull* capabilities to obtain the calculus CMC_P.

The syntax of CMC_P processes is similar to that in Table 5.1, except that we extend the definition of μ in Table 6.1 to also include $push(m_A)\ n_B$ and $pull(m_A)\ n_B$ for all A and B. Furthermore, the definition of λ in Table 6.1 is extended to also include ambient auxiliary actions $pushed(m_A)\ n_B$, $pulled(m_A)\ n_B$ and $move(m_A)\ n_B$ for all m, n and A and B (note that $move(m_A)\ n_B$ actions are different from $move\ n_B$ actions of CMC). The capabilities $pull(m_A)\ n_B$ and $push(m_A)\ n_B$ are used to move passive ambients around, $push(m_A)\ n_B$ allows an ambient m_A to push an ambient n_B out of its boundary, whereas $pull(m_A)\ n_B$ allows m_A to pull in n_B into its scope. Since most of the syntax is identical to the syntax in Chapter 5, we omit the detailed explanation.

Ambient Prefixes :	$\mu ::=$	$in\ n_B$	$\mid$	$out\ n_B$		
	$\mid$	$push(m_A)\ n_B$	$\mid$	$pull(m_A)\ n_B$		
Action Prefixes :	$\alpha ::=$	$a(z)$	$\mid$	$\overline{b}(z)$	$\mid$	τ
Ambient Action :	$\lambda ::=$	$enter\ n_B$	$\mid$	$exit\ n_B$	$\mid$	$move\ n_B$
	$\mid$	$pulled(m_A)\ n_B$	$\mid$	$pushed(m_A)\ n_B$	$\mid$	$move(m_A)\ n_B$
	$\mid$	μ				
Labels :	$\ell ::=$	μ	$\mid$	α		
	$\mid$	λ	$\mid$	τ		
Outcomes :	$O ::=$	P	$\mid$	K		
Concretions :	$K ::=$	$(\nu\tilde{m})\langle P\rangle Q$				

Table 6.1: Prefixes, labels, concretions and outcomes

6.2 Reduction Semantics of $\mathrm{CMC_P}$

The reduction semantics of $\mathrm{CMC_P}$ is given in terms of structural congruence, $\equiv$, and the reduction relation, $\rightarrow$. The additional reduction rules for the *push* and *pull* capabilities are given in Table 6.2.

$$m_A[push(m_A)\ n_B.P \mid Q \mid n_B[R]] \rightarrow m_A[P \mid Q] \mid n_B[R] \quad \text{(Red Push)}$$

$$m_A[pull(m_A)\ n_B.P \mid Q] \mid n_B[R] \rightarrow m_A[P \mid Q \mid n_B[R]] \quad \text{(Red Pull)}$$

Table 6.2: Reduction axioms for *push* and *pull*

Structural congruence, $\equiv$, for $\mathrm{CMC_P}$ processes is as in Definition 3.2 where capabilities C include additionally $push(m_A)\ n_B$ and $pull(m_A)\ n_B$ for all A and B. The reduction relation, $\rightarrow$, for $\mathrm{CMC_P}$ processes is as in Definition 3.3 except that it satisfies additionally the reductions in Table 6.2.

The reductions in Table 6.2 do not allow any ambient to enter or exit other ambient, rather being pulled inside or being pushed away by another ambient, as is illustrated by an example inspired by [54].

Example 6.1. We consider an ambient *client* that controls its mobility and can freely enter or leave an ambient named *server* by exercising its *in server* and *out server* capabilities, namely

$$client[in\ server.P] \mid server[Q], \text{ reduces to } server[client[P] \mid Q] \text{ and}$$

$$server[client[out\ server P] \mid Q],\ \text{reduces to}\ client[P] \mid server[Q].$$

Here, *server* cannot avoid *client* to enter even if it knows that *client* is an harmful agent. We rewrite the same scenario using *push* and *pull* capabilities as follows:

$$client[P] \mid server[pull(server)\ client.Q] \rightarrow server[client[P] \mid Q] \text{ and}$$

$$server[push(server)\ client.Q \mid client[P]] \rightarrow client[P] \mid server[Q]$$

Now, *server* controls *clients*'s mobility across its boundary and can pull it in whenever needed and push away the unwanted *client* by its $pull(server)$ *client* and $push(server)$ *client* capabilities.

6.3 Transition Semantics for Push and Pull

The SOS rules for the *push* and *pull* capabilities are given in Tables 6.3, 6.4, and we also use SOS rules in Table 4.3. As before, we use concretions $\nu\tilde{m}\langle P\rangle Q$ in our rules. In order to illustrate reductions and transitions associated with the *pull* capability, consider the agent

$$m_A[pull(m_A)\ n_B.P_1] \mid n_B[P_2]$$

for some P_1 and P_2, where P_2 has no private names. The ambient m_A runs in parallel with the ambient n_B; m_A pulls ambient n_B in by its $pull(m_A)\ n_B$ capability. By Red Pull in Table 6.2 we obtain

$$m_A[pull(m_A)\ n_B.P_1] \mid n_B[P_2] \longrightarrow m_A[P_1 \mid n_B[P_2]]$$

Next, we derive the τ-transition of $m_A[pull(m_A) n_B.P_1] \mid n_B[P_2]$ that corresponds to the reduction above by using τ-Pull rule in Table 6.3.
We have $pull(m_A)\ n_B.P_1 \xrightarrow{pull(m_A)\ n_B} P_1$. When the actual movement happens, we must identify the agent that is pulled in, and the agents that remain behind. To model this, we use concretions of the form $\nu\tilde{n}\langle P\rangle Q$, where P represents the agent which is pulled in, while Q stays behind and $\tilde{n}$ is the set of private names shared by P and Q. We introduce a new action $pulled(m_A)\ n_B$ and by the rule Pull we obtain

$$m_A[pull(m_A)\ n_B.P_1] \xrightarrow{pulled(m_A)\ n_B} \langle P_1\rangle 0$$

This shows that when $pull(m_A)\ n_B$ is exercised the agent P_1 remains inside the pulling ambient m_A. Then next to achieve the τ-transition there must exist a sibling ambient n_B. We define a new action $move(m_A)\ n_B$ for n_B to complete this

$$\text{(Act-Pull)}\ \frac{}{pull(m_A)\ n_B.P \xrightarrow{pull(m_A)\ n_B} P}$$

$$\text{(Pull)}\ \frac{P \xrightarrow{pull(m_A)\ n_B} P'}{m_A[P] \xrightarrow{pulled(m_A)\ n_B} \langle P' \rangle 0}$$

$$\text{(Move)}\ \frac{}{n_B[P] \xrightarrow{move(m_A)\ n_B} \langle n_B[P] \rangle 0}$$

$$\text{(Res-Pull)}\ \frac{P \xrightarrow{pull(m_A)\ n_B} P'}{(\nu u)P \xrightarrow{pull(m_A)\ n_B} (\nu u)P'}\ (u \neq n_B)$$

$$\text{(Par-Pull)}\ \frac{P \xrightarrow{pull(m_A)\ n_B} P'}{P \mid Q \xrightarrow{pull(m_A)\ n_B} P' \mid Q}$$

$$(\tau\text{-Pull})\ \frac{P \xrightarrow{pulled(m_A)\ n_B} (\nu \tilde{p})\langle P' \rangle P'' \qquad Q \xrightarrow{move(m_A)\ n_B} (\nu \tilde{q})\langle Q' \rangle Q''}{P \mid Q \xrightarrow{\tau} \nu \tilde{p} \nu \tilde{q}(m_A[P' \mid Q'] \mid P'' \mid Q'')}\ (*)$$

$(*)(fn(P') \cup fn(P'')) \cap \tilde{q} = (fn(Q') \cup fn(Q'')) \cap \tilde{p} = \emptyset$

Table 6.3: SOS rules for pull

interaction, by the rule Move we obtain

$$n_B[P_2] \xrightarrow{move(m_A)\ n_B} \langle n_B[P_2] \rangle 0$$

Now τ-Pull gives us

$$m_A[pull(m_A)\ n_B.P_1] \mid n_B[P_2] \xrightarrow{\tau} m_A[P_1 \mid n_B[P_2]]$$

The τ-transition shows that n_B, the sibling of m_A, becomes after the transition a child of m_A.

Next, we illustrate the use of τ-Push in Table 6.4 by considering the agent

$$m_A[push(m_A)\ n_B.P \mid Q \mid n_B[R]]$$

for some P, Q and R, where m_A has the capability to push out its child ambient n_B. For simplicity, we assume that Q and R have no private names. By the axiom Red Push in Table 6.2 we obtain

$$m_A[push(m_A)\ n_B.P \mid Q \mid n_B[R]] \rightarrow m_A[P \mid Q] \mid n_B[R]$$

$$\text{(Act-Push)}\ \frac{}{push(m_A)\ n_B.P \xrightarrow{push(m_A)\ n_B} P}$$

$$\text{(Pushed)}\ \frac{}{n_B[P] \xrightarrow{pushed(m_A)\ n_B} \langle n_B[P]\rangle 0}$$

$$\text{(Res-Push)}\ \frac{P \xrightarrow{push(m_A)\ n_B} P'}{(\nu u)P \xrightarrow{push(m_A)\ n_B} (\nu u)P'}\ (u \neq n_B)$$

$$\text{(Par-Push)}\ \frac{P \xrightarrow{push(m_A)\ n_B} P'}{P \mid Q \xrightarrow{push(m_A)\ n_B} P' \mid Q}$$

$$(\tau\text{-Push})\ \frac{P \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{p}\langle P'\rangle P''' \qquad P''' \xrightarrow{push(m_A)\ n_B} P''}{m_A[P] \xrightarrow{\tau} \nu\tilde{p}(m_A[P''] \mid P')}\ (**)$$

$$(**)(fn(P') \cup fn(P''')) \cap \tilde{m} = \emptyset$$

Table 6.4: SOS rules for push

The application of τ-Push for the agent gives the following transition

$$m_A[push(m_A)\ n_B.P \mid Q \mid n_B[R]] \xrightarrow{\tau} m_A[P \mid Q] \mid n_B[R]$$

The τ-Push rule uses the notion of *lookahead* as, for example, in [68, 69]. In order to derive a τ-transition of $m_A[P]$ we need to ensure that P contains an n_B ambient. This is achieved by $P \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{p}\langle P'\rangle P'''$ where P' is this n_B ambient. The remaining ambient P''' must then be able to perform the pushing: $P''' \xrightarrow{push(m_A)\ n_B} P''$. Hence P''' is used both on the right-hand side and on the left-hand side of the premises in τ-Push, so τ-Push has a lookahead.

6.3.1 Applications of Push and Pull Capabilities

We consider scenarios where both active and passive mobile structures interact with each other and communicate globally.

Example 6.2. We consider a setting where a mobile device *Personal Digital Assistant (PDA)* sends a message to its user. The user cannot view the message unless he picks up the device. The system is

$$user_A[pull(user_A)\ pda.P_u \mid a(x).P'_u] \mid pda[\overline{a}(v).P_{pda}]$$

where the ambient $user_A$, with $a \notin A$, runs in parallel with the PDA which is repre-

sented by the ambient *pda*. The ambient *pda* sends a value v on port a. Inside the ambient $user_A$, a communication on a and the *pull* capability are concurrent. The $user_A$ may pull *pda* by its $pull(user_A)$ *pda* capability, but it cannot receive input on a since $a \notin A$. The τ-transition for the *pull* capability is as follows:

$$\nu a(pda[\overline{a}(v).P_{pda}] \mid user_A[pull(user_A)\ pda.P_u \mid a(x).P'_u]) \xrightarrow{\tau}$$

$$\nu a(user_A[P_u \mid a(x).P'_u \mid pda[\overline{a}(v).P_{pda})]]$$

It makes the *pda*, which is initially the sibling of $user_A$, a child of $user_A$. Now communication on a can take place:

$$\nu a(user_A[P_u \mid a(x).P'_u \mid pda[\overline{a}(v).P_{pda})]] \xrightarrow{\tau} \nu a(user_A[P_u \mid P'_u\{v/x\} \mid pda[P_{pda})]]$$

Example 6.3. We consider a setting where a server *serv* communicates globally with an active mobile agent m_A, for some A, via a device dev_B for some B. Here, dev_B is a passive mobile agent. The server *serv* sends services to the appropriate device provided that an authorised user is holding it, in this case ambient m_A. Also, only m_A may turn on dev_B to activate services. *serv* instructs the mobile agent to move from its current location to room n, and the agent m_A views the instructions from server on the device dev_B. The graphical representation of the above given setting is given in Figure 6.1.

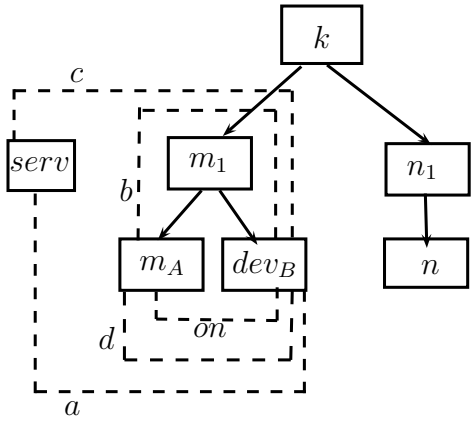


Figure 6.1: Global communication, active and passive mobility

We define formally the component processes of our setting as follows:

$$\begin{array}{ll} MobileAmb & \stackrel{def}{=} pull(m_A)\ dev_B.\overline{on}.a(u).\overline{d}(m_1,u).a(z).z.0 \\ MA & \stackrel{def}{=} m_A[MobileAmb] \end{array}$$

$$\begin{array}{ll} Device & \stackrel{def}{=} on.c_1(x).\overline{a}(x).d(y,x).\overline{b}(y,x).c_1(z).\overline{a}(z).0 \\ D & \stackrel{def}{=} dev_B[Device] \end{array}$$

$$\begin{array}{ll} Server & \stackrel{def}{=} \overline{c_1}(n).b(y,x).\overline{c_1}(path(T,y,x)).0 \\ S & \stackrel{def}{=} serv[Server] \end{array}$$

The system is the parallel composition of the component processes with appropriate channels of communication restricted:

$$\begin{array}{rl} \nu(on\ abcd)(S \mid MA \mid D) & \equiv \nu(on\ abcd)(serv[\overline{c_1}(n).b(y,x).\overline{c_1}(path(T,y,x)).0] \\ & \mid m_A[pull(m_A)\ dev_B.\overline{on}.a(u).\overline{d}(m_1,u).a(z).z.0] \\ & \mid dev_B[on.c_1(x).\overline{a}(x).d(y,x).\overline{b}(y,x).c_1(z).\overline{a}(z).0]) \end{array}$$

The interaction steps among the three agents are :

- $pull(m_A)\ dev_B$: The active mobile agent m_A initiates the interaction by picking up the device dev_B by its *pull* capability and switching the device on via port *on*. The device is now ready to be interacted by the main server *serv*.
- $\overline{c_1}(n), c_1(x)$: The server *serv* sends the target location n on c_1 to dev_B, whereas $c_1(x)$ allows dev_B to receive the value n.
- $\overline{a}(x)$, $a(u)$: The device then sends the target location n to m_A via a.
- $\overline{d}(m_1,u), d(y,x)$: After receiving a value on a, m_A sends its source(m_1) and target locations on d to dev_B.
- $\overline{b}(y,x), b(y,x)$: The device then sends the two values received from m_A to the server. The server is ready to receive the two values, in this case m_1 and n as source and target locations respectively.
- $\overline{c}(path(T,y,x))$: The server sends the path on port c to the device, which is calculated by using the function $path(T,y,x)$.
- $\overline{a}(z), a(z).z.0$: The device displays the path to the ambient m_A on port a, and ambient m_A binds it to z. In this case the value received is the sequence of

capabilities representing the path between the source and the target ambients. In this particular case the path sent by *server* is $out\ m_1.in\ n_1.in\ n.0$, which followed by m_A to move from m_1 to n.

The only possible sequence of execution of $\nu(on\ abcd)(S \mid MA \mid D)$ is

$$\xrightarrow{\tau_{pull}} \xrightarrow{\tau_{on}} \xrightarrow{\tau_{c_1}} \xrightarrow{\tau_a} \xrightarrow{\tau_d} \xrightarrow{\tau_b} \xrightarrow{\tau_{c_1}} \xrightarrow{\tau_a} \xrightarrow{\tau_{out\ m_1}} \xrightarrow{\tau_{in\ n_1}} \xrightarrow{\tau_{in\ n}} S', \text{ for some } S'.$$

We list the individual transition of this execution below:

$\nu(on\ abc)(serv[\overline{c_1}(n).b(y,x).\overline{c_1}(path(T,y,x)).0] \mid k[m_1[m_A[pull(m_A)\ dev_B.\overline{on}.a(u).$
$\overline{d}(m_1,u).a(z).z.0]] \mid n_1[n[0]]] \mid dev_B[on.c_1(x).\overline{a}(x).d(y,x).\overline{b}(y,x).c_1(z).\overline{a}(z).0]) \xrightarrow{\tau}$

$\nu(on\ abc)(serv[\overline{c_1}(n).b(y,x).\overline{c_1}(path(T,y,x)).0] \mid k[m_1[m_A[\overline{on}.a(u).\overline{d}(m_1,u).a(z).z.0$
$\mid dev_B[on.c_1(x).\overline{a}(x).d(y,x).\overline{b}(y,x).c_1(z).\overline{a}(z).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu(on\ abc)(serv[\overline{c_1}(n).b(y,x).\overline{c_1}(path(T,y,x)).0] \mid k[m_1[m_A[a(u).\overline{d}(m_1,u).a(z).z.0 \mid$
$dev_B[c_1(x).\overline{a}(x).d(y,x).\overline{b}(y,x).c_1(z).\overline{a}(z).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu(on\ abc)(serv[b(y,x).\overline{c_1}(path(T,y,x)).0] \mid k[m_1[m_A[a(u).\overline{d}(m_1,u).a(z).z.0 \mid$
$dev_B[\overline{a}(n).d(y,x).\overline{b}(y,x).c_1(z).\overline{a}(z).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu(on\ abc)(serv[b(y,x).\overline{c_1}(path(T,y,x)).0] \mid k[m_1[m_A[\overline{d}(m_1,n).a(z).z.0 \mid$
$dev_B[d(y,x).\overline{b}(y,x).c_1(z).\overline{a}(z).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu(on\ abc)(serv[b(y,x).\overline{c_1}(path(T,y,x)).0] \mid k[m_1[m_A[a(z).z.0 \mid$
$dev_B[\overline{b}(m_1,n).c_1(z).\overline{a}(z).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu(on\ abc)(serv[\overline{c_1}(path(T,m_1,n)).0] \mid k[m_1[m_A[a(z).z.0 \mid dev_B[c_1(z).\overline{a}(z).0]]]$
$\mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu(on\ abc)(serv[0] \mid k[m_1[m_A[a(z).z.0 \mid dev_B[\overline{a}(out\ m_1.in\ n_1.in\ n).0]]] \mid n_1[n[0]]]) \xrightarrow{\tau}$
$\nu(on\ abc)(serv[0] \mid k[m_1[m_A[out\ m_1.in\ n_1.in\ n.0 \mid dev_B[0]]] \mid n_1[n[0]]]).$

In this example we show that *serv* starts communicating m_A via dev_B when m_A switches on dev_B. Following the sequence of capabilities shared by *serv*, m_A hold-

ing dev_B moves from m_1 to n. Thus the system evolves to

$$serv[0] \mid k[m_1[0] \mid n_1[n[m_A[dev_B[0]]]]],$$

which is represented graphically in Figure 6.2 below.

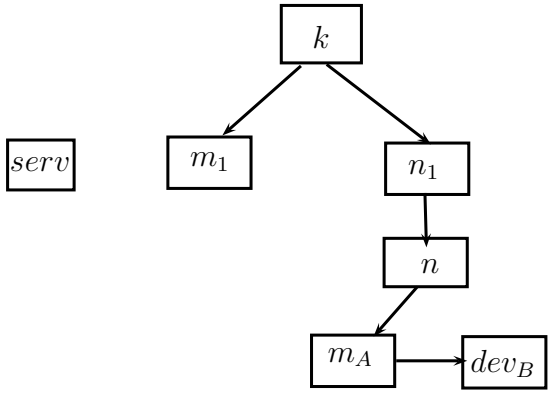


Figure 6.2: Resulting system: global communication, active and passive mobility

6.4 Correspondence of Semantics

In this section we show the correspondence of the reduction semantics and the transition semantics for CMC_P. Let T' be a sub-calculus of CMC_P that consists of all operators of CMC_P apart from the prefixing with actions (including τ) operators, the choice operator and the relabelling operator. There are "soundness" and "completeness" parts of this correspondence.

6.4.1 Soundness

Soundness ensures that for every reduction of a T' term there is a valid τ-transition of the term, and the target of the τ-transition is congruent to the target of the reduction. We easily have the soundness part of this correspondence between the two semantics:

Theorem 6.1. $\forall P,\ R \in T'.\ P \to R \Longrightarrow \exists\, Q \in T'.\ P \xrightarrow{\tau} Q \equiv R.$

Proof. By structural induction where we consider cases of reductions of P depending on the structure of P.

1. Base case: (Constant)
   We show that our statement holds when we choose the simplest term of T', namely the deadlocked agent 0:

$$0 \rightarrow R \Longrightarrow 0 \xrightarrow{\tau} R$$

There is no rule defined to show the reduction of 0, so the reduction $0 \rightarrow R$ is false, and hence, the implication above is true.

2. Induction Hypothesis:

   We assume that our statement holds for all the sub-processes P' of P, namely if $P' \rightarrow R$ then $P' \xrightarrow{\tau} R$, for all R.

3. Induction Step:

   (a) $P = (\nu m)P'$, for some P'.

   In this case we show that

$$(\nu m)P' \rightarrow R \Longrightarrow (\nu m)P' \xrightarrow{\tau} R \tag{6.1}$$

   We assume, $(\nu m)P' \rightarrow R$.

   The reduction of $(\nu m)P'$ can be derived by using the Red Res given in Table 3.4.

   Since the term $(\nu m)P'$ reduces to some process R, by the reduction rule Red Res, so we deduce that the reduction $P' \rightarrow Q$ is also valid for some Q, such that $(\nu m)Q = R$. Since $P' \rightarrow Q$ is proved valid, so by the inductive hypothesis we obtain $P' \xrightarrow{\tau} Q$.

   Since $P' \xrightarrow{\tau} Q$ is valid, so by the rule λ-Res in Table 4.3, $(\nu m)P' \xrightarrow{\tau} (\nu m)Q$ is a valid transition.

   Next, by Struct in Table 4.3, $(\nu m)P \xrightarrow{\tau} (\nu m)Q$ and $(\nu m)Q \equiv R$. Hence, we obtain $(\nu m)P \xrightarrow{\tau} R$ as required.

   (b) $P = n_A[P']$, for some P' and n_A.

   In this case we show that

$$n_A[P'] \rightarrow R \Longrightarrow n_A[P'] \xrightarrow{\tau} R \tag{6.2}$$

   To prove statement 6.2, we assume $n_A[P'] \rightarrow R,$ for some R.

   There are three reduction rules, namely Red Amb and Red Out in Table 3.4, and Red Push in Table 6.2, that can be used to derive a reduction transition of $n_A[P']$. Since the proofs for Red Amb and Red Out are identical to the proofs of the corresponding cases given in Subsection 4.2.1, they are omitted.

i. Red Push:

By using the rule Red Push, we deduce that P' is of the form $push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q]$ for some P_1, P_2 and Q. Hence, the reduction $n_A[push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q]] \rightarrow R$ is valid by Red Push, where R is of the form $n_A[P_1 \mid P_2] \mid m_B[Q]$, that is $R = n_A[P_1 \mid P_2] \mid m_B[Q]$.

In principle there could be shared and private names in agents P_1, P_2 and Q. However, using α-conversion if necessary, we can assume without loss of generality that there are no shared and private names among P_1, P_2 and Q. We derive the τ transition of agent $n_A[push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q]]$ by applying τ-Push in Table 6.4. This is supported by inference tree in Figure 6.3. Hence, the resulting transition is as follows:

$$n_A[push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q]] \xrightarrow{\tau} n_A[P_1 \mid P_2] \mid m_B[Q]$$

Now by Struct, since $n_A[push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q]] \xrightarrow{\tau} n_A[P_1 \mid P_2] \mid m_B[Q]$ and $R \equiv n_A[P_1 \mid P_2] \mid m_B[Q]$. Hence, we obtain $n_A[push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q]] \xrightarrow{\tau} R$ as required.

(c) $P = P' \mid Q$, parallel composition of P' and Q.

In this case we show that

$$P' \mid Q \rightarrow R \Longrightarrow P' \mid Q \xrightarrow{\tau} R \tag{6.3}$$

Assume $P' \mid Q \rightarrow R$ for some R.

There are three reduction rules, namely Red Par and Red In in Table 3.4, and Red Pull in Table 6.2, that can be used to derive a reduction of $P' \mid Q$. The proofs for Red In and Red Par are identical to the proof of the corresponding cases given in Subsection 4.2.1, they are omitted.

i. Red Pull

We assume that P' and Q are of the form $m_A[pull(m_A)\ n_B.P_1 \mid P_2]$ and $n_B[Q_1]$ respectively for some P_1, P_2 and Q_1. By the reduction rule Red Pull, we obtain

$$m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid n_B[Q_1] \rightarrow R$$

We deduce that the agent R is of the form $m_A[P_1 \mid P_2 \mid n_B[Q_1]]$,

namely $R = m_A[P_1 \mid P_2 \mid n_B[Q_1]]$. In principle there could be shared and private names in agents P_1, P_2 and Q_1. However, using α-conversion if necessary, we assume wlog that there are no shared and private names in P_1, P_2 and Q_1.
Now we derive the τ transition of $m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid n_B[Q_1]$ by applying τ-Pull in Table 6.3. This is supported by inference tree in Figure 6.4. Hence, the resulting transition is as follows:

$$m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid n_B[Q_1] \xrightarrow{\tau} m_A[P_1 \mid P_2 \mid n_B[Q_1]]$$

Now by Struct,
$m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid n_B[Q_1] \xrightarrow{\tau} m_A[P_1 \mid P_2 \mid n_B[Q_1]]$ and $R \equiv m_A[P_1 \mid P_2 \mid n_B[Q_1]]$. Hence, we obtain $m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid n_B[Q_1] \xrightarrow{\tau} R$ as required.

□

$$\text{(Act)}\ \frac{}{push(n_A)\ m_B.P_1 \xrightarrow{push(n_A)\ m_B} P_1} \qquad \text{(Pushed)}\ \frac{}{m_B[Q] \xrightarrow{pushed(n_A)\ m_B} \langle m_B[Q]\rangle 0}$$

$$\text{(Par-Push)}\ \frac{}{push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q] \xrightarrow{push(n_A)\ m_B} P_1 \mid P_2 \mid m_B[Q]} \qquad \text{(λ-Par)}\ \frac{}{P_1 \mid P_2 \mid m_B[Q] \xrightarrow{pushed(n_A)\ m_B} \langle m_B[Q]\rangle 0 \mid P_1 \mid P_2}$$

$$\tau\text{-Push}\ \frac{}{n_A[push(n_A)\ m_B.P_1 \mid P_2 \mid m_B[Q]] \xrightarrow{\tau} n_A[P_1 \mid P_2 \mid 0] \mid m_B[Q] \equiv n_A[P_1 \mid P_2] \mid m_B[Q]}$$

Figure 6.3: Inference tree for τ-Push transition

$$\text{(Act-Pull)}\ \frac{}{pull(n_A)\ m_B.P_1 \xrightarrow{pull(n_A)\ m_B} P_1}$$

$$\text{(Par-Pull)}\ \frac{}{pull(n_A)\ m_B.P_1 \mid P_2 \xrightarrow{pull(n_A)\ m_B} P_1 \mid P_2}$$

$$\text{(Pull)}\ \frac{}{n_A[pull(n_A)\ m_B.P_1 \mid P_2] \xrightarrow{pulled(n_A)\ m_B} \langle P_1 \mid P_2\rangle 0} \qquad \text{(Move)}\ \frac{}{m_B[Q] \xrightarrow{move\ n_B} \langle m_B[Q]\rangle 0}$$

$$\tau\text{-Pull}\ \frac{}{n_A[pull(n_A)\ m_B.P_1 \mid P_2] \mid m_B[Q] \xrightarrow{\tau} n_A[P_1 \mid P_2 \mid m_B[Q]]}$$

Figure 6.4: Inference tree for τ-Pull transition

6.4.2 Completeness

Completeness ensures that for every valid τ-transition of a T' term there is a valid reduction of the term, and the targets of the τ-transitions and the reductions are the same.

Lemma 6.1.

1. $\forall P.P \xrightarrow{\pi} O$, *where* $\pi \in \{pull(m_A)\ n_B,\ push(m_A)\ n_B\} \Longrightarrow \exists\ \tilde{p}, P_1, P_2,$ *with* $m_A, n_B \notin \tilde{p}$ *such that* $P \equiv \nu\tilde{p}\ (\pi.P_1 \mid P_2)$ *and* $O \equiv \nu\tilde{p}\ (P_1 \mid P_2)$, *where* $\tilde{p}$ *is a set of ambient names private in* P.

2. $\forall P, P', P''.P \xrightarrow{pulled(m_A)\ n_B} (\nu\tilde{p})\langle P'\rangle P'' \Longrightarrow \exists\ P_1, P_2, P_3,$ *with* $m_A, n_B \notin \tilde{p}$ *such that* $P \equiv \nu\tilde{p}\ (m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid P_3)$, $P' \equiv P_1 \mid P_2$ *and* $P'' \equiv P_3$, *where* $\tilde{p}$ *is a set of ambient names private in* P.

3. $\forall Q, Q', Q''.Q \xrightarrow{move(m_A)\ n_B} \nu(\tilde{q})\ \langle Q'\rangle Q'' \Longrightarrow \exists\ Q_1, Q_2$ *with* $m_A, n_B \notin \tilde{q}$ *such that* $Q \equiv \nu\tilde{q}\ (n_B[Q_1] \mid Q_2), Q' \equiv n_B[Q_1]$ *and* $Q'' \equiv Q_2$, *where* $\tilde{q}$ *is a set of ambient names private in* Q.

4. $\forall P, P', P'', P'''.\ P \xrightarrow{pushed(m_A)\ n_B} (\nu\tilde{p})\ \langle P'\rangle P'''$ *and* $P''' \xrightarrow{push(m_A)\ n_B} P'' \Longrightarrow \exists\ P_1, P_2, P_3,$ *with* $m_A, n_B \notin \tilde{p}$, *such that* $P \equiv \nu\tilde{p}\ (push(m_A)\ n_B.P_1 \mid P_2 \mid n_B[P_3])$, $P' \equiv n_B[P_3], P''' \equiv push(m_A)\ n_B.P_1 \mid P_2$ *and* $P'' \equiv P_1 \mid P_2$, *where* $\tilde{p}$ *is a set of ambient names private in* P.

Proof. By transition induction.

The proofs of Parts 1, 2 and 3 of Lemma 6.1 are very similar to the proofs of the corresponding parts of Lemma 4.1. For example, *enter* n_B action part 2 of Lemma 4.1 states that if $P \xrightarrow{enter\ n_B} \nu\tilde{p}\ \langle P'\rangle P''$ then P has the form $\nu\tilde{p}\ (k_A[in\ n_B.P_1 \mid P_2] \mid P_3)$ for some P_1, P_2, P_3, k_A with $n_B \notin \tilde{p}$. Similarly, for $pulled(m_A)\ n_B$ action, part 2 of Lemma 6.1 states that if $P \xrightarrow{pulled(m_A)\ n_B} (\nu\tilde{p})\langle P'\rangle P''$ then P has the form $\nu\tilde{p}\ (m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid P_3)$ for some $P_1, P_2, P_3,$ with $m_A, n_B \notin \tilde{p}$. The only difference between the two statements is that in the first case the capability is exercised by the moving ambient, whereas in the second case the target ambient exercises the pull capability to move an ambient in. The difference is clearly stated by the SOS rules Enter and Pulled in Tables 4.2 and 6.4 for the corresponding actions.

Due to the close similarities between Lemma 6.1 and Lemma 4.1, the proofs for the first three parts of Lemma 6.1 are omitted.

The proof for the part 4 is as follows:

Assume $P \xrightarrow{pushed(m_A)\ n_B} (\nu\tilde{p})\ \langle P'\rangle P'''$ and $P''' \xrightarrow{push(m_A)\ n_B} P''$ for some P, P', P'', P'''. There are three cases $(a)-(c)$ below for $P \xrightarrow{pushed(m_A)\ n_B} (\nu\tilde{p})\ \langle P'\rangle P'''$ depending on the structure of P. Since the transitions of this part of the lemma use lookahead each case consists of a number of nested sub-cases:

(a) Pushed ($P \equiv k_C[R]$)

(b) λ-Par ($P \equiv R \mid Q$)

(c) λ-Res ($P \equiv \nu u_E\ (R)$)

(a) Pushed

In this case we assume $P \equiv k_c[R]$ for some k_C and R. Hence we get the transition $k_C[R] \xrightarrow{pushed(m_A)\ n_B} \langle k_C[R]\rangle 0$ if $k_C = n_B$. Since there is no SOS rule for transitions of 0, so $0 \xrightarrow{push(m_A)\ n_B} R''$ is false for all R''. Hence $k_C[R] \xrightarrow{pushed(m_A)\ n_B} \langle k_C[R]\rangle 0$ and $0 \xrightarrow{push(m_A)\ n_B} R''$ is false, so the implication of part 4 is true.

(b) λ-Par

In this case we assume that $P \equiv R \mid Q$. So we get the transition of the form $R \mid Q \xrightarrow{pushed(m_A)\ n_B} O \mid Q$ for some O. Since $R \mid Q \xrightarrow{pushed(m_A)\ n_B} O \mid Q$ is a valid transition by λ-Par in Table 4.3, the premise of the rule, namely $R \xrightarrow{pushed(m_A)\ n_B} O$, is also valid where $O \equiv \nu\tilde{r}\langle R'\rangle R'''$ for some $\tilde{r}, R', R'''$ and $m_A, n_B \notin \tilde{r}$. So we have

$$R \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{r}\langle R'\rangle R'''$$

Now by α-conversion, if necessary, $\tilde{r}$ is selected in such a way that $fn(Q) \cap \tilde{r} = \emptyset$, so we now have

$$R \mid Q \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{r}\langle R'\rangle (R''' \mid Q)$$

Next we consider three sub-cases for $R''' \xrightarrow{push(m_A)\ n_B} R''$ depending on the structure of R''' in $R''' \mid Q$.

i. $R''' \equiv push(m_A)\ n_B.S$, for some S, m_A, n_B.

ii. $R''' \equiv S_1 \mid S_2$, for some S_1, S_2

iii. $R'''\nu s(S)$, for some s and S

i. In this case R''' has the form $push(m_A)\ n_B.S$.
The transition $push(m_A)\ n_B.S \xrightarrow{push(m_A)\ n_B} S$ is valid by Act-Push, where $R''' \equiv push(m_A)\ n_B.S$ and $R'' \equiv S$.

Since $R \overset{pushed(m_A)\ n_B}{\longrightarrow} \nu\tilde{r}\langle R'\rangle R'''$ and $R''' \overset{push(m_A)\ n_B}{\longrightarrow} S$, so by inductive hypothesis $R \equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3])$, $R' \equiv n_B[R_3]$, $R''' \equiv push(m_A)\ n_B.R_1 \mid R_2$, $R'' \equiv R_1 \mid R_2$ and $\tilde{r}$ is a set of private names in R, for some R_1, R_2 and R_3. By α-conversion, if necessary, we can select $\tilde{r}$ in such a way that $fn(Q) \cap \tilde{r} = \emptyset$. So, we get

$$\begin{aligned} R \mid Q &\equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3]) \mid Q \\ &\equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3] \mid Q) \end{aligned}$$

Similarly,

$$\begin{aligned} O \mid Q &\equiv \nu\tilde{r}\langle n_B[R_3]\rangle(push(m_A)\ n_B.R_1 \mid R_2) \mid Q \\ &\equiv \nu\tilde{r}\langle n_B[R_3]\rangle(push(m_A)\ n_B.R_1 \mid R_2 \mid Q) \end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid n_B[R_3] \mid R_2 \mid Q)$, $P' \equiv n_B[R_3]$, $P''' \equiv push(m_A)\ n_B.R_1 \mid R_2 \mid Q$, $P'' \equiv R_1 \mid R_2 \mid Q$ and $\tilde{r}$ is the set of private names in P as required.

ii. In this case R''' has the form $S_1 \mid S_2$, namely $R''' \equiv S_1 \mid S_2$. By Par-Push $S_1 \mid S_2 \overset{push(m_A)\ n_B}{\longrightarrow} S_1' \mid S_2$ for some S_1' is valid. The premise $S_1 \overset{push(m_A)\ n_B}{\longrightarrow} S_1'$ is also valid.
Since $R \overset{pushed(m_A)\ n_B}{\longrightarrow} \nu\tilde{r}\langle R'\rangle R'''$ and $R''' \overset{push(m_A)\ n_B}{\longrightarrow} S_1' \mid S_2$ are valid, by inductive hypothesis there exist R_1, R_2, R_3 such that,
$R \equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3])$,
$R' \equiv n_B[R_3]$, $R''' \equiv push(m_A)\ n_B.R_1 \mid R_2$ and $R'' \equiv R_1 \mid R_2$. So, $S_1 \equiv push(m_A)\ n_B.R_1$, $S_2 \equiv R_2$ and $R' \equiv n_B[R_3]$.
Now by α-conversion, if necessary, $\tilde{r}$ is selected in such a way that $fn(Q) \cap \tilde{r} = \emptyset$, and we get

$$\begin{aligned} R \mid Q &\equiv \nu\tilde{r}(S_1 \mid S_2 \mid n_B[R_3]) \mid Q \\ &\equiv \nu\tilde{r}(S_1 \mid S_2 \mid n_B[R_3] \mid Q) \end{aligned}$$

Similarly,

$$\begin{aligned} O \mid Q &\equiv \nu\tilde{r}\langle n_B[R_3]\rangle(S_1 \mid S_2) \mid Q \\ &\equiv \nu\tilde{r}\langle n_B[R_3]\rangle(S_1 \mid S_2 \mid Q) \end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{r}(S_1 \mid S_2 \mid n_B[R_3] \mid Q)$, $P' \equiv n_B[R_3]$, $P''' \equiv S_1 \mid S_2 \mid Q$, $P'' \equiv S_1' \mid S_2 \mid Q$ and $\tilde{r}$ is the set of private names in P as required.

iii. In this case R''' has the form $\nu s(S)$ for some name s private in S. The transition $\nu s(S) \overset{push(m_A)\ n_B}{\longrightarrow} \nu s(S')$, where $s \neq n_B$, is valid by Res-Push. The premise of the rule, $S \overset{push(m_A)\ n_B}{\longrightarrow} S'$, is also valid. Since $R \overset{pushed(m_A)\ n_B}{\longrightarrow} \nu\tilde{r}\langle R'\rangle R'''$ and $R''' \overset{push(m_A)\ n_B}{\longrightarrow} \nu s(S')$ are valid, where $\nu s(S') \equiv R''$, by inductive hypothesis there exist R_1, R_2, R_3 such that

$R \equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3]), R' \equiv n_B[R_3]$,
$R''' \equiv push(m_A)\ n_B.R_1 \mid R_2$ and $R'' \equiv R_1 \mid R_2$. We now have, $\nu s(S) \equiv push(m_A)\ n_B.R_1$ and $\nu s(S') \equiv R_1$, $R_2 \equiv 0$ since we have assumed $R''' \equiv \nu s(S)$ at the beginning of this case, and $R' \equiv n_B[R_3]$. Now,

$$\begin{aligned} R \mid Q &\equiv \nu\tilde{r}(\nu s(S) \mid R_2 \mid n_B[R_3]) \mid Q \\ &\equiv \nu\tilde{r}(\nu s(S \mid R_2 \mid n_B[R_3])) \mid Q \quad s \notin fn(R_3) \\ &\equiv \nu(s,\tilde{r})(S \mid R_2 \mid n_B[R_3] \mid Q) \quad fn(Q) \cap (s,\tilde{r}) = \emptyset \end{aligned}$$

Similarly,

$$\begin{aligned} O \mid Q &\equiv \nu\tilde{r}\langle n_B[R_3]\rangle(\nu s(S) \mid R_2) \mid Q \\ &\equiv \nu\tilde{r}\langle n_B[R_3]\rangle \nu s(S \mid R_2 \mid Q) \\ &\quad (s \notin fn(R_3) \text{ and } fn(Q) \cap (s,\tilde{r}) = \emptyset) \\ &\equiv \nu(s,\tilde{r})\langle n_B[R_3]\rangle(S \mid R_2 \mid Q) \qquad s \in fn(R_3) \end{aligned}$$

Hence we obtain, $P \equiv \nu(s,\tilde{r})(S \mid R_2 \mid n_B[R_3] \mid Q)$, where $R_1 \equiv S, R_2 \equiv 0$ and R_3 is in $n_B[R_3]$, $P' \equiv n_B[R_3]P''' \equiv \nu s(S)$ and $P'' \equiv \nu s(S')$.

(c) λ-Res

In this case we assume that $P \equiv \nu u(R)$, where name u is private in R. So we get the transition of the form $\nu u(R) \overset{pushed(m_A)\ n_B}{\longrightarrow} \nu u(O)$, for some O. Since $\nu u(R) \overset{pushed(m_A)\ n_B}{\longrightarrow} \nu u(O)$ for $u \neq n_B$, is a valid transition by λ-Res in Table 4.3, the premise of the rule, namely $R \overset{pushed(m_A)\ n_B}{\longrightarrow} O$ is also valid, where $O \equiv \nu\tilde{r}\langle R'\rangle R'''$ for some $\tilde{r}, R', R'''$ and $m_A, n_B \notin \tilde{r}$. So we have

$$R \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{r}\langle R'\rangle R'''$$

Next we consider three sub-cases for $R''' \xrightarrow{push(m_A)\ n_B} R''$ for some R'' depending on the structure of R'''.

i. $R''' \equiv push(m_A)\ n_B.S$, for some S

ii. $R''' \equiv S_1 \mid S_2$, for some S_1, S_2

iii. $R'''\nu s(S)$, for ambient name s private in S

i. In this case R''' has the form $push(m_A)\ n_B.S$. By Act-Push in Table 6.4, $push(m_A)\ n_B.S \xrightarrow{push(m_A)\ n_B} S$ is valid, where $R''' \equiv push(m_A)\ n_B.S$ and $R'' \equiv S$.
Since $R \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{r}\langle R'\rangle R'''$ and $R''' \xrightarrow{push(m_A)\ n_B} S$, so by inductive hypothesis $R \equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3])$, $R' \equiv n_B[R_3]$, $R''' \equiv push(m_A)\ n_B.R_1 \mid R_2$, $R'' \equiv R_1 \mid R_2$ and $\tilde{r}$ is a set of private names in R, for some R_1, R_2 and R_3. So $S \equiv R_1, R_2 \equiv 0$, since we assumed $R''' \equiv push(m_A)\ n_B.S$ at the beginning of this case. So we have

$$\begin{array}{rcll} \nu u(R) & \equiv & \nu u(\nu\tilde{r}(push(m_A)\ n_B.S \mid 0 \mid n_B[R_3])) & \\ & \equiv & \nu u(\nu\tilde{r}(push(m_A)\ n_B.S \mid n_B[R_3])) & \text{(Struct Zero Par)} \\ & \equiv & \nu(u,\tilde{r})(push(m_A)\ n_B.S \mid n_B[R_3]) & u \notin fn(R_3) \end{array}$$

Similarly,

$$\begin{array}{rl} O \mid Q & \equiv \nu u(\nu\tilde{r}\langle n_B[R_3]\rangle push(m_A)\ n_B.S \mid R_2) \\ & \equiv \nu\tilde{r}\langle n_B[R_3]\rangle \nu u(push(m_A)\ n_B.R_1 \mid R_2) \quad \textit{if } u \notin fn(S) \\ & \equiv \nu(u,\tilde{r})\langle n_B[R_3]\rangle push(m_A)\ n_B.R_1 \mid R_2) \quad \textit{if } u \in fn(S) \end{array}$$

Hence, we obtain $P \equiv \nu(u,\tilde{r})(push(m_A)\ n_B.S \mid R_2 \mid n_B[R_3])$, $P' \equiv n_B[R_3]$, $P''' \equiv push(m_A)\ n_B.S$ and $P'' \equiv S$, where $S \equiv R_1$ and $R_2 \equiv 0$, and $\nu(u,\tilde{r})$ the private names in P as required.

ii. In this case R''' has the form $S_1 \mid S_2$, namely $R''' \equiv S_1 \mid S_2$ for some S_1, S_2. By Par-Push $S_1 \mid S_2 \xrightarrow{push(m_A)\ n_B} S_1' \mid S_2$ for some S_1' is valid. Since $R \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{r}\langle R'\rangle R'''$ and $R''' \xrightarrow{push(m_A)\ n_B} S_1' \mid S_2$ are valid, by inductive hypothesis there exists R_1, R_2, R_3 such that
$R \equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3]), R' \equiv n_B[R_3]$,
$R''' \equiv push(m_A)\ n_B.R_1 \mid R_2$ and $R'' \equiv R_1 \mid R_2$. So we now have $S_1 \equiv push(m_A)\ n_B.R_1$. $S_2 \equiv R_2$ and $R' \equiv n_B[R_3]$. We now have

$$\begin{array}{rcl} \nu u(R) & \equiv & \nu u(\nu\tilde{r}(S_1 \mid S_2 \mid n_B[R_3])) \\ & \equiv & \nu\tilde{r}(S_1 \mid S_2 \mid n_B[R_3] \mid Q) \quad fn(Q) \cap \tilde{r} = \emptyset, \end{array}$$

where new restriction is $\nu\tilde{r}' = \nu(u,\tilde{r})$, so we get

$$\nu u(R) \equiv \nu\tilde{r}'(S_1 \mid S_2 \mid n_B[R_3])$$

Similarly,

$$\begin{aligned}\nu u(O) &\equiv \nu u(\nu \tilde{r}\langle n_B[R_3]\rangle S_1 \mid S_2)\\ &\equiv \nu(u, \nu\tilde{r})\langle n_B[R_3]\rangle(S_1 \mid S_2)\\ &\equiv \nu\tilde{r}\langle n_B[R_3]\rangle(\nu u)(S_1 \mid S_2)\\ &\quad (u \notin fn(R_3) \text{ and } fn(S_2) \cap (u, \tilde{r}) = \emptyset)\end{aligned}$$

Hence, we obtain $P \equiv \nu\tilde{r}(S_1 \mid S_2 \mid n_B[R_3] \mid Q)$, $P' \equiv n_B[R_3]$, $P''' \equiv S_1 \mid S_2 \mid Q$, $P'' \equiv S_1' \mid S_2 \mid Q$ and $\tilde{r}$ is the set of private names in P as required.

iii. In this case R''' has the form $\nu s(S)$, where name s is private in S. The transition $\nu s(S) \overset{push(m_A)\ n_B}{\longrightarrow} \nu s(S')$, where $s \neq n_B$ is valid by Res-Push.
Since $R \overset{pushed(m_A)\ n_B}{\longrightarrow} \nu\tilde{r}\langle R'\rangle R'''$ and $R''' \overset{push(m_A)\ n_B}{\longrightarrow} \nu s(S')$ are valid for $s \notin \tilde{r}$, where $\nu s(S') \equiv R''$, by inductive hypothesis
$R \equiv \nu\tilde{r}(push(m_A)\ n_B.R_1 \mid R_2 \mid n_B[R_3]), R' \equiv n_B[R_3]$,
$R''' \equiv push(m_A)\ n_B.R_1 \mid R_2$ and $R'' \equiv R_1 \mid R_2$ for some R_1, R_2, R_3.
So we have $\nu s(S) \equiv push(m_A)\ n_B.R_1$ and $\nu s(S') \equiv R_1, R_2 \equiv 0$ and $R' \equiv n_B[R_3]$.
Now,

$$\begin{aligned}\nu u(R) &\equiv \nu u(\nu\tilde{r}(\nu s(S) \mid R_2 \mid n_B[R_3]))\\ &\equiv \nu u(\nu\tilde{r}(\nu s(S \mid R_2 \mid n_B[R_3]))) \quad s \neq n_B\\ &\equiv \nu(u, s, \tilde{r})(S \mid R_2 \mid n_B[R_3]),\end{aligned}$$

here, new restriction $\nu\tilde{r}'' = \nu(u, s, \tilde{r})$ and we obtain

$$\nu u(R) \equiv \nu\tilde{r}''(S \mid R_2 \mid n_B[R_3])$$

Similarly,

$$\begin{aligned}\nu u(O) &\equiv \nu u(\nu\tilde{r}\langle n_B[R_3]\rangle(\nu s(S) \mid R_2))\\ &\equiv \nu u(\nu\tilde{r}\langle n_B[R_3]\rangle \nu s(S \mid R_2))\\ &\quad (s \notin fn(R_3) \text{ and } fn(R_2) \cap (s, \tilde{r}) = \emptyset)\\ &\equiv \nu(u, \tilde{r})\langle n_B[R_3]\rangle \nu s(S \mid R_2)\\ &\equiv \nu(u, s, \tilde{r})\langle n_B[R_3]\rangle(S \mid R_2) \qquad s \in fn(R_3)\\ &\equiv \nu(\tilde{r}'')\langle n_B[R_3]\rangle(S \mid R_2)\end{aligned}$$

Hence we obtain, $P \equiv \nu(\tilde{r}'')(S \mid R_2 \mid n_B[R_3])$, where $R_1 \equiv S, R_2 \equiv 0$

and R_3 is in $n_B[R_3]$, $P' \equiv n_B[R_3], P''' \equiv \nu s(S)$ and $P'' \equiv \nu s(S')$.

□

Theorem 6.2. $\forall S, R \in T'.\ S \xrightarrow{\tau} R \Longrightarrow S \to R$.

The proof of Theorem 6.2 relies on several auxiliary statements given in Lemma 6.1. For example, for τ-Push rule, we require that
if $S \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{p}\langle P'\rangle P'''$ then

(a) P' has the form $n_B[Q]$ for some Q, and

(b) if $P''' \xrightarrow{push(m_A)\ n_B} P''$ then P''' has a sub-term $push(m_A)n_B.Q'$ for some Q' possibly nested within a parallel and restriction context.

Proof. By transition induction where we consider the most significant cases of transitions of T' terms. We have omitted the proofs of cases which are identical to the proofs of the corresponding cases of Theorem 4.2.

1. $S = C.P$, where the prefix C is an ambient capability, namely $in\ n_B$, $out\ n_B$, $pull(m_A)\ n_B$ and $push(m_A)\ n_B$.

   We consider the ambient's pull capability, and show that

   $$pull(m_A)\ n_B.P \xrightarrow{\tau} R \Longrightarrow pull(m_A)\ n_B.P \longrightarrow R$$

   The only transition for $pull(m_A)\ n_B.P$ is $pull(m_A)\ n_B.P \xrightarrow{pull(m_A)\ n_B} P$ by applying the rule Act-Pull in Table 6.3. Since there is no other rule defined that could be applied to derive the transition, the τ-transition for $pull(m_A)\ n_B.P$ is not possible. Thus, the transition $pull(m_A)\ n_B.P \xrightarrow{\tau} R$ is not valid, and hence, the implication above is true.

   The proofs for $in\ n_B$, $out\ n_B$, and $push(m_A)\ n_B$ capabilities are very similar to the proof for $pull(m_A)\ n_B$, they are omitted.

2. $S = m_A[P]$, an ambient with name m_A and body P.
   In this case we show that

   $$m_A[P] \xrightarrow{\tau} R \Longrightarrow m_A[P] \longrightarrow R \tag{6.4}$$

   We assume $m_A[P] \xrightarrow{\tau} R$, for some R.

   There are three transition rules τ-Out, τ-Amb and τ-Push in Tables 4.2, 4.3, and 6.4 respectively, that can be used to derive a τ-transition of $m_A[P]$. Since

the proofs for τ-Out and τ-Amb are identical to the proofs of the corresponding cases of Theorem 4.2, they are omitted.

(a) τ-Push

Since $m_A[P] \xrightarrow{\tau} R$ is valid by τ-Push in Table 6.4, so we deduce that the premises of the rule, namely $P \xrightarrow{pushed(m_A)\ n_B} \nu\tilde{m}\langle Q\rangle S''$ and $S'' \xrightarrow{push(m_A)\ n_B} S'$, for some $\tilde{m}, S', S'', Q$ are also valid.

Here, $\tilde{m}$ is the set of private ambient names in process P, and $((fn(Q') \cup fn(S'')) \cap \{\tilde{m}\}) = \emptyset$. Since by τ-Push, $R = (\nu\tilde{m})(m_A[S'] \mid Q)$, hence we have

$$m_A[P] \xrightarrow{\tau} (\nu\tilde{m})(m_A[S'] \mid Q)$$

Now using part 4 of Lemma 6.1, we have
$P \equiv \nu\tilde{m}(push(m_A)\ n_B.P_1 \mid n_B[P_3] \mid P_2)$,
$Q \equiv n_B[P_3],\ S'' \equiv push(m_A)\ n_B.P_1 \mid P_2,\ S' \equiv P_1 \mid P_2,$ for some $P_1, P_2,$ and P_3 where $m_A \notin \tilde{m}$, we now have

$$\begin{aligned} m_A[P] &\equiv m_A[\nu\tilde{m}(push(m_A)\ n_B.P_1 \mid P_2 \mid n_B[P_3])] \\ &\equiv \nu\tilde{m}(m_A[push(m_A)\ n_B.P_1 \mid P_2 \mid n_B[P_3]]) \quad \text{(Struct Res Amb)} \\ &\quad (\text{ where it is assumed wlog that } m_A \notin \tilde{m}) \end{aligned}$$

By Red Push $m_A[push(m_A)\ n_B.P_1 \mid P_2 \mid n_B[P_3]] \rightarrow m_A[P_1 \mid P_2] \mid n_B[P_3]$, so $\nu\tilde{m}\ (m_A[push(m_A)\ n_B.P_1 \mid P_2 \mid n_B[P_3]])$ rewrites as follows:

$$\begin{aligned} &\rightarrow \nu\tilde{m}\ (m_A[P_1 \mid P_2] \mid n_B[P_3]) && \text{(Red Res)} \\ &\equiv \nu\tilde{m}\ (m_A[S'] \mid n_B[P_3]) && \text{(Struct Amb)} \\ &\equiv \nu\tilde{m}\ (m_A[S'] \mid Q) && \text{(Struct Amb)} \end{aligned}$$

Now using the structural congruence rule Red $\equiv$ in Table 3.4, we have $m_A[P] \rightarrow \nu\tilde{m}\ (m_A[S'] \mid Q)$, since $\nu\tilde{m}\ (m_A[S'] \mid Q) \equiv R$, hence we obtain $m_A[S] \rightarrow R$ as required.

3. $S = (\nu m_A)P$, an ambient name m_A private in P.
In this case we show that

$$(\nu m)P \xrightarrow{\tau} R \Longrightarrow (\nu m)P \longrightarrow R \tag{6.5}$$

We assume $(\nu m)P \xrightarrow{\tau} R$.

Since $(\nu m)P \xrightarrow{\tau} R$, for some process R is valid by the transition rule λ-Res in Table 4.3, so we deduce that the premises namely $P \xrightarrow{\tau} P'$ of the transition rule is valid for some P'. By the rule Res-Amb the agent R is of the form $(\nu m)P'$, that is $R = (\nu m)P'$. Since $P \xrightarrow{\tau} P'$ is valid, so by the inductive hypothesis we get $P \to P'$.

Since the reduction $P \longrightarrow P'$ is valid, so by reduction rule Red Res in Table 3.4, we obtain $(\nu m)P \to (\nu m)P'$.

Now using the structural congruence rule (Red $\equiv$) in Table 3.4, we have $\nu mP \to \nu mP'$, since $R \equiv \nu mP'$, hence we obtain $\nu mP \to R$ as required.

4. $S = P \mid Q$, parallel composition of P and Q.
   In this case we show

$$P \mid Q \xrightarrow{\tau} R \Longrightarrow P \mid Q \longrightarrow R \tag{6.6}$$

We assume $P \mid Q \xrightarrow{\tau} R$ for some R.

There are three transition rules τ-In, λ-Par and τ-Pull in Tables 4.3 and 6.3, that can be used to derive a τ-transition of $P \mid Q$. So, to apply each rule separately, we divide this case into two sub-cases.

There are three transition rules τ-In, λ-Par and τ-Pull in Tables 4.2, 4.3 and 6.3 respectively, that can be used to derive $P \mid Q \xrightarrow{\tau} R$. Since the proofs for τ-In and λ-Par are identical to the proofs of the corresponding cases of Theorem 4.2, they are omitted.

(a) τ-Pull

Since $P \mid Q \xrightarrow{\tau} R$ is valid by τ-Pull rule, so we deduce that the premises of the rule are also valid. These premises are $P \xrightarrow{pulled(m_A)\ n_B} \nu\tilde{p}\langle P'\rangle P''$ for some P' and P'', and $Q \xrightarrow{move(m_A)\ n_B} \nu\tilde{q}\langle Q'\rangle Q''$ for some Q' and Q'', where $\tilde{p}$ and $\tilde{q}$ are the sets of ambient names that are private in P and Q respectively. Here, condition $(*)$ holds, which says that $(fn(P') \cup fn(P'')) \cap \tilde{q} = (fn(Q') \cup fn(Q'')) \cap \tilde{p} = \emptyset$. We use m_A and n_B for some ambient names. The agent R is of the form $(\nu\tilde{p})(\nu\tilde{q})(m_A[P' \mid Q'] \mid P'' \mid Q'')$, hence the resulting transition is

$$P \mid Q \xrightarrow{\tau} (\nu\tilde{p})(\nu\tilde{q})(m_A[P' \mid Q'] \mid P'' \mid Q'')$$

Now, by part 2 of Lemma 6.1 there exists P_1, P_2, P_3, such that
$P \equiv \nu\tilde{p}\ (m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid P_3),\ P' \equiv P_1 \mid P_2$ and $P'' \equiv P_3$
Furthermore, by part 3 of Lemma 6.1 there exists Q_1, Q_2, and we have

$$Q \equiv \nu\tilde{q}\ (n_B[Q_1] \mid Q_2),\ Q' \equiv n_B[Q_1] \text{ and } Q'' \equiv Q_2.$$

Hence, we deduce that

$$P \mid Q \equiv \nu\tilde{p}\ (m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid P_3) \mid \nu\tilde{q}\ (n_B[Q_1] \mid Q_2)$$

Since by $(*)$, members of $\tilde{q}$ are not free names in $\nu\tilde{p}\ (m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid P_3)$, and members of $\tilde{p}$ are not free names in $(n_B[Q_1] \mid Q_2)$, by Struct Res Par we obtain

$$\begin{aligned} P \mid Q &\equiv (\nu\tilde{p})(\nu\tilde{q})\ (m_A[pull(m_A)\ n_B.P_1 \mid P_2] \mid P_3 \mid n_B[Q_1] \mid Q_2) \\ &\rightarrow \nu\tilde{p}\ \nu\tilde{q}\ (m_A[P_1 \mid P_2 \mid n_B[Q_1]] \mid P_3 \mid Q_2) \qquad \text{(Red In)} \\ &\equiv \nu\tilde{p}\ \nu\tilde{q}\ (m_A[P' \mid Q'] \mid P'' \mid Q'') \end{aligned}$$

Now by Red $\equiv$ in Table 3.4, we obtain $P \mid Q \rightarrow \nu\tilde{p}\ \nu\tilde{q}\ (n_B[P' \mid Q'] \mid P'' \mid Q'')$ and $\nu\tilde{p}\ \nu\tilde{q}\ (n_B[P' \mid Q'] \mid P'' \mid Q'') \equiv R$, hence $P \mid Q \rightarrow R$ as required.

□

6.5 Conclusion

In this chapter CMC has been extended with additional mobility primitives, namely *push* and *pull* capabilities are introduced to model passive and active mobile structures in the setting of ubiquitous computing, giving us CMC_P. We have proposed transition operational semantics for CMC_P and proved that the semantics is sound and complete with respect to the standard reduction semantics. The operational semantics has used the notion of lookahead in the SOS rules. The usefulness of CMC_P has been exemplified by presenting case studies and examples.

Chapter 7

Context-Awareness: Location and Surrounding

In this chapter we add a context-awareness mechanism to CMC_P. We address very basic aspects of context-awareness, where agents are aware of their current locations and surroundings. This is done by adding two operators to the existing syntax of CMC_P thus obtaining CMC_PCA. These operators are (a) $ploc(x)$ for parent location that queries the name of the parent of an ambient, and (b) $sloc(x)$ for sibling location that queries the sibling's name of an ambient.

In smart indoor settings, location is considered an important entity for providing communication among various portable and static structures. We consider a scenario where an agent *server* instructs a mobile ambient *client* to move from its current location to some other location. The agent *server* takes source and target locations to calculate a path between the two locations, and outputs the path as a message to *client*. Such a setting is represented as follows:

$$server\ \boxed{\overline{a}(n_B).b(x,y).\overline{c}(path(T,x,y)).P_s}\ |$$

$$k_C\ \boxed{m_A\ \boxed{client\ \boxed{a(u).\overline{b}(m_A,u).c(z).z.P_c}\ |\ P_m}\ |\ n_B\ \boxed{P_n}}$$

where *server* instructs *client* to move from m_A to n_B, inside a building k_C. Here, the moving agent *client* sends the parent's name m_A to *server* whenever requested. The server in response calculates a path between the two locations. If *client* moves around the structure, its location may not be known. So, we need an operator to find out the current location of an ambient. We rewrite the same scenario by using the construct $ploc(x)$, which queries its parent's name as follows:

$$server\ \boxed{\overline{a}(n_B).b(x,y).\overline{c}(path(T,x,y)).P_s}\ |$$

$$k_C\ \boxed{m_A\ \boxed{client\ \boxed{a(u).ploc(x).\overline{b}(x,u).c(z).z.P_c}\ |\ P_m}\ |\ n_B\ \boxed{P_n}}$$

Now, *client* obtains the name of its parent by using $ploc(x)$. A more detailed explanation of how *ploc* and *sloc* operators work, is presented by example scenarios in Section 7.3.1.

We now review the related work. The inspiration for the work presented in this chapter comes initially from [62, 63, 64], where Satoh has researched spatial organisation of systems and concluded that technological advancements have enabled computing devices to become aware of their surroundings. Location awareness has turned out to be useful in many applications, in particular, in determining position, navigation, tracking, and monitoring of ubiquitous computing devices.

Leonhardt [33] classified location models into two major categories, namely geometric and symbolic models. In geometric models locations are represented as coordinates systems, whereas symbolic location models use the notion of place and labelling the locations. We use the notion of place to model location, and represent the structure of our system by a hierarchical space tree. The nodes represent the places, objects or computing devices, whereas the edges represent the containment relations between objects. Each node or object is represented by named ambient, which may contain nested ambients inside [10].

The Calculus of Context Aware Ambients (CCA) [18] describes the context-awareness requirements of the mobile systems. It introduces the notion of context expression that constraints the capabilities. This makes the computations context dependent. It introduces the notion of context expression that constraints the ambient capability. The context guarded capability has the form $k?M$, where k is a context expression and M is a capability. This capability can only be performed if the environment satisfies its guard $k?$. We also add basic forms of context awareness mechanism to our calculus. The new capabilities *ploc(x).P* and *sloc(x).P* allow an ambient to acquire the name of its parent and sibling respectively, and pass it as x to P.

Conversation Calculus [76, 9] is a process calculus designed for expressing and analysing service based systems. It proposes a spatial communication topology where conversation contexts are used as message exchange patterns. The coordinating participants may join or leave a conversation dynamically. In our CMC we do not have any such contextual communication but the agents communicate globally using ports as in CCS [76]. The construct *here(x)* that allows access to the

conversation medium in Conversation Calculus is similar to the *ploc(x)* and *sloc(x)* constructs of our calculus. The capabilities *ploc(x)* and *sloc(x)* enable ambients to be aware of their current locations and surroundings respectively, these are not precisely used for only communication, whereas in Conversation Calculus conversation contexts are proposed as communication medium that controls information sharing among processes.

This chapter is organised as follows. We give the extended calculus CMC_{PCA} in Section 7.1, where two new location awareness constructs are introduced. In Section 7.2 reduction semantics for *ploc* and *sloc* is given, which is followed by their transition semantics. The usefulness of the extended calculus is illustrated with small examples in Section 7.3. Section 7.4 we conjecture that the transition semantics is sound and complete with respect to the standard reduction semantics. The application of CMC_{PCA} is presented by in two case studies of interactive shopping mall and devices automatically switching their modes in Section 7.5. Finally, Section 7.6 concludes the chapter.

7.1 Context Awareness Primitives

We start by presenting the existing syntax of CMC_{P} in Table 7.1, in addition we introduce new constructs, namely $ploc(x)$ for parent location that queries the parent's name of an ambient, as well as $sloc(x)$ for sibling's location that queries the sibling name of an ambient. We extend the definition of μ in Table 7.2 to include further *ploc(x)* and *sloc(x)*. Also, the definition of λ in Table 7.2 is extended to include further auxiliary labels *ploc1(x)*, *sloc1(x)* and *amb* n_B. CMC_{P} extended with the *ploc(x)* and *sloc(x)* primitives becomes the calculus CMC_{PCA}.

$Names:$	$m_A, n_B, k_C ... \in \mathcal{N}$							
$Actions:$	$\alpha, \beta, ... \in Act$							
$Variables:$	$x, y, ... \in \mathcal{X}$							
$Processes:$	$P, Q ::=$	D	$\mid$	$C.P$	$\mid$	$a(z).P$	$\mid$	$\overline{a}(x).P$
	$\mid$	$m_A[P]$	$\mid$	$P + Q$	$\mid$	$P \mid Q$	$\mid$	$(\nu m_A)P$
	$\mid$	$(\nu l)P$	$\mid$	$P[f]$				
$Capabilities:$	$C ::=$	x	$\mid$	μ	$\mid$	ϵ	$\mid$	$C.C'$

Table 7.1: Syntax of CMC_{PCA}

Ambient Prefixes :	$\mu ::=$	$in\ n_B$	$\mid$	$out\ n_B$		
	$\mid$	$push(m_A)\ n_B$	$\mid$	$pull(m_A)\ n_B$		
	$\mid$	$ploc(x)$	$\mid$	$sloc(x)$		
Action Prefixes :	$\alpha ::=$	$a(z)$	$\mid$	$\overline{b}(z)$	$\mid$	τ
Ambient Action :	$\lambda ::=$	$enter\ n_B$	$\mid$	$exit\ n_B$	$\mid$	$move\ n_B$
	$\mid$	$pulled(m_A)\ n_B$	$\mid$	$pushed(m_A)\ n_B$	$\mid$	$move(m_A)\ n_B$
	$\mid$	$ploc1(x)$	$\mid$	$sloc1(x)$	$\mid$	μ
Labels :	$\ell ::=$	μ	$\mid$	α		
	$\mid$	λ	$\mid$	τ		
Outcomes :	$O ::=$	P	$\mid$	K		
Concretions :	$K ::=$	$(\nu\tilde{m})\langle P\rangle Q$				

Table 7.2: Prefixes, labels, concretions and outcomes

7.2 Reduction Semantics for CMC_{PCA}

The reduction semantics of CMC_{PCA} is given in terms of structural congruence, $\equiv$, and the reduction relation, $\rightarrow$. The reductions for *ploc*(x) and *sloc(x)* are given in Table 7.3.

$$m_A[n_B[ploc(x).P \mid Q] \mid R] \rightarrow m_A[n_B[P\{x \leftarrow m_A\} \mid Q] \mid R] \qquad \text{(Red Ploc)}$$

$$m_A[P] \mid n_B[sloc(y).Q \mid S] \rightarrow m_A[P] \mid n_B[Q\{y \leftarrow m_A\} \mid S] \qquad \text{(Red Sloc)}$$

Table 7.3: Reduction axioms for *ploc* and *sloc*

Structural congruence, $\equiv$, for CMC_{PCA} processes is as in Section 6.2 where capabilities C include additionally *ploc(x)* and *sloc(x)*. The reduction relation, $\rightarrow$, for CMC_{PCA} processes is as in Section 6.2 except that it satisfies the additional axioms in Table 7.3. To show some basic reduction computations, assume that our agent is of the form

$$n_B[m_A[ploc(x).P \mid Q] \mid R]$$

In this setting ambient m_A is the child of ambient n_B. The construct *ploc*(x) enables m_A to gain the knowledge of its parent's name. *ploc*(x) acts as an action guarding P. By the reduction rule Red Ploc, the term $n_B[m_A[ploc(x).P \mid Q] \mid R]$ reduces to $n_B[m_A[P\{x \leftarrow n_B\} \mid Q] \mid R]$, where $P\{x \leftarrow n_B\}$ denotes process P with all occurrences of variable x replaced by ambient n_B.

Next, assume that our agent is of the form

$$m_A[sloc(y).P \mid Q] \mid n_B[R]$$

In this setting two ambients m_A and n_B exists in parallel. The construct $sloc(y)$ enables the ambient m_A to find out its sibling's name. $sloc(y)$ acts as an action guarding P. By the reduction rule Red Sloc, the term $m_A[sloc(y).P \mid Q] \mid n_B[R]$ reduces to $m_A[P\{y \leftarrow n_B\} \mid Q] \mid n_B[R]$, where $P\{y \leftarrow n_B\}$ denotes process P with all occurrences of variable y replaced by ambient n_B.

7.3 Transition Semantics for Ploc and Sloc

We develop an operational semantics for the *ploc* and *sloc* primitives of CMC_{PCA}. The SOS rules for the extended calculus are presented in Tables 7.4, 7.5, and we also use SOS rules in Table 4.3. This extension allows ambients (i) to have a knowledge of their location (parent ambient name) by the virtue of $ploc(x)$ construct, and (ii) to have a knowledge of their surroundings (sibling ambient name) by the virtue of $sloc(x)$ construct. In τ-Ploc and τ-Sloc rules we have used look-ahead terms as in [68, 69] and concretions of the form $\nu\tilde{p}\langle P'\rangle P''$ as in [44, 36, 37].

We consider some examples showing the usefulness of $ploc(x)$ and $sloc(x)$ constructs and, at the same time explain the SOS rules for them given in Tables 7.4, 7.5, and 4.3.

Firstly, we explain the use of τ-Ploc by considering the agent

$$\nu\tilde{p}(m_A[n_B[ploc(x).P_1 \mid P_2] \mid Q])$$

In this example an ambient n_B executes in parallel with an agent Q inside an ambient m_A, and $\tilde{p}$ is the set of private ambient names. Processes P_1 and P_2 are executing in parallel inside ambient n_B. The ambient n_B queries parent's name by the virtue of its $ploc(x)$ capability. Since by Red Ploc in Table 7.3 the agent $m_A[n_B[ploc(x).P_1 \mid P_2] \mid Q]$ reduces to $m_A[n_B[P_1\{x \leftarrow m_A\} \mid P_2] \mid Q]$, so by Red Res we obtain

$$\nu\tilde{p}(m_A[n_B[ploc(x).P_1 \mid P_2] \mid Q]) \longrightarrow \nu\tilde{p}(m_A[n_B[P_1\{x \leftarrow m_A\} \mid P_2] \mid Q]) \qquad (7.1)$$

Now, we show the corresponding τ-transition of $\nu\tilde{p}(m_A[n_B[ploc(x).P_1 \mid P_2] \mid Q])$ which can be derived by τ-Ploc in Table 7.4. In τ-Ploc, lookahead terms are used that help in replacing all the occurrences of variable x by the parent ambient name m_A. For actual substitution of m_A to occur we must consider two parts of our

$$\text{(Act-Ploc)}\ \frac{}{ploc(x).P \overset{ploc(z)}{\longrightarrow} P\{x \leftarrow z\}}\ (z \text{ doesn't appear in } P)$$

$$\text{(τ-Ploc)}\ \frac{P \overset{ploc1(z)}{\longrightarrow} \nu\tilde{p}\langle P'\rangle Q \qquad P' \overset{ploc(z)}{\longrightarrow} P''}{m_A[P] \overset{\tau}{\rightarrow} (\nu\tilde{p})m_A[P''\{z \leftarrow m_A\} \mid Q])}$$

$$\text{(Ploc1)}\ \frac{}{m_A[P] \overset{ploc1(z)}{\longrightarrow} \langle m_A[P]\rangle 0}$$

$$\text{(Amb-Ploc)}\ \frac{P \overset{ploc(z)}{\longrightarrow} P'}{n_B[P] \overset{ploc(z)}{\longrightarrow} n_B[P']}$$

$$\text{(Par-Ploc)}\ \frac{P \overset{ploc(z)}{\longrightarrow} P'}{P \mid Q \overset{ploc(z)}{\longrightarrow} P' \mid Q}\ (z \notin fn(Q))$$

$$\text{(Res-Ploc)}\ \frac{P \overset{ploc(z)}{\longrightarrow} P'}{(\nu u)P \overset{ploc(z)}{\longrightarrow} (\nu u)P'}\ (u \neq z)$$

$$\text{(Par-Ploc1)}\ \frac{P \overset{ploc1(z)}{\longrightarrow} O}{P \mid Q \overset{ploc1(z)}{\longrightarrow} O \mid Q}\ (z \notin fn(Q))$$

$$\text{(Res-Ploc1)}\ \frac{P \overset{ploc1(z)}{\longrightarrow} O}{(\nu u)P \overset{ploc1(z)}{\longrightarrow} (\nu u)O}\ (u \neq z)$$

Table 7.4: SOS rules for *ploc*

ambient, namely the sub-agent where parent's name substitutes the variable x, and the sub-agent that remains unchanged. To model these two possibilities, we use concretions of the form $P \overset{def}{=} \nu\tilde{p}\langle P'\rangle P''$. Here, P' represents the agent where m_A substitutes all the occurrences of variable x, P'' is the unchanged agent, and $\tilde{p}$ is the set of private ambient names in P. Now, we introduce a new action $ploc1(z)$ and by rule Ploc1 in Table 7.4, we obtain

$$n_B[ploc(x).P_1 \mid P_2] \overset{ploc1(z)}{\longrightarrow} \langle n_B[ploc(x).P_1 \mid P_2]\rangle 0$$

By using the Par-Ploc1 we have

$$n_B[ploc(x).P_1 \mid P_2] \mid Q \overset{ploc1(z)}{\longrightarrow} \langle n_B[ploc(x).P_1 \mid P_2]\rangle (0 \mid Q) \qquad \text{(A)}$$

where $z \notin fn(Q)$. The transition A corresponds to the first premise of τ-Ploc. By this transition the agent $n_B[ploc(x).P_1 \mid P_2]$ enquires the name of its parent.

The simplest transition performed by ambient n_B in $n_B[ploc(x).P_1 \mid P_2]$ is

$ploc(x).P_1 \stackrel{ploc(z)}{\longrightarrow} P_1\{x \leftarrow z\}$. By Act-Ploc in Table 7.4, z is a fresh variable that does not appear in process P_1, and $P_1\{x \leftarrow z\}$ replaces all occurrences of variable x by z in process P_1. By using Par-Ploc, we have $ploc(x).P_1 \mid P_2 \stackrel{ploc(z)}{\longrightarrow} P_1\{x \leftarrow z\} \mid P_2$, where $z \notin fn(P_2)$. By τ-Amb, we obtain

$$n_B[ploc(x).P_1 \mid P_2] \stackrel{ploc(z)}{\longrightarrow} n_B[P_1\{x \leftarrow z\} \mid P_2] \tag{B}$$

The transition B corresponds to the second premise of τ-Ploc.

Since we have derived A and B, we obtain by the application of τ-Ploc the following:

$$\nu\tilde{p}(m_A[n_B[ploc(x).P_1 \mid P_2] \mid Q]) \stackrel{\tau}{\longrightarrow} \nu\tilde{p}(m_A[n_B[P_1\{x \leftarrow z\} \mid P_2]\{z \leftarrow m_A\} \mid Q])$$

Since z does not appear free in P_2 by rules for substitution, the target of this transition becomes

$$\nu\tilde{p}(m_A[n_B[ploc(x).P_1 \mid P_2] \mid Q]) \stackrel{\tau}{\longrightarrow} \nu\tilde{p}(m_A[n_B[P_1\{x \leftarrow m_A\} \mid P_2] \mid Q])$$

Next, we explain how ambients enquire names of their siblings with the $sloc(x)$ operator. Consider the process

$$\nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3) \mid n_B[R]$$

The ambient m_A is running in parallel with the ambient n_B with R executing inside. P_1 and P_2 are executing in parallel inside ambient m_A and P_3 executes in parallel with m_A, and $\tilde{p}$ is the set of private ambient names. The ambient m_A queries its sibling's name by the $sloc(x)$ capability.

By the axiom (Red Sloc) in Table 7.3 we obtain the following reduction for our process

$$\nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3) \mid n_B[R] \longrightarrow \nu\tilde{p}(m_A[P_1\{x \leftarrow n_B\} \mid P_2] \mid P_3) \mid n_B[R]$$

We show the corresponding τ-transition of $\nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3)$ by τ-Sloc in Table 7.5. Lookahead terms are used in τ-Sloc to successfully replace all the occurrences of variable x in P_1 by the sibling's ambient name. When this substitution occurs we must consider two parts of our ambient, namely the sub-agent where sibling ambient name substitutes all the occurrences of variable x, and the part of agent that remains unchanged. To model these two situations we use concretions of the form $P \stackrel{def}{=} \nu\tilde{p}\langle P'\rangle P''$. Here, P' represents the agent where variable x is replaced by the sibling ambient name, P'' is the unchanged agent and $\tilde{p}$ is the set of private

$$\text{(Act-Sloc)}\ \frac{}{sloc(x).P \xrightarrow{sloc(z)} P\{x \leftarrow z\}}\ (z \text{ doesn't appear in } P)$$

$$\text{(Sloc1)}\ \frac{}{m_A[P] \xrightarrow{sloc1(z)} \langle m_A[P]\rangle 0}$$

$$\text{(Amb-Sloc)}\ \frac{P \xrightarrow{sloc(z)} P'}{m_A[P] \xrightarrow{sloc(z)} m_A[P']}$$

$$\text{(Par-Sloc)}\ \frac{P \xrightarrow{sloc(z)} P'}{P \mid Q \xrightarrow{sloc(z)} P' \mid Q}\ (z \notin fn(Q))$$

$$\text{(Res-Sloc)}\ \frac{P \xrightarrow{sloc(z)} P'}{(\nu u)P \xrightarrow{sloc(z)} (\nu u)P'}\ (u \neq z)$$

$$\text{(Par-Sloc1)}\ \frac{P \xrightarrow{sloc1(z)} O}{P \mid Q \xrightarrow{sloc1(z)} O \mid Q}\ (z \notin fn(Q))$$

$$\text{(Res-Sloc1)}\ \frac{P \xrightarrow{sloc1(z)} O}{(\nu u)P \xrightarrow{sloc1(z)} (\nu u)O}\ (u \neq z)$$

$$\text{(Sib-Amb)}\ \frac{}{n_B[P] \xrightarrow{amb\, n_B} P}$$

$$\text{(Par-Amb)}\ \frac{P \xrightarrow{amb\, n_B} P'}{P \mid Q \xrightarrow{amb\, n_B} P'}$$

$$\text{(Res-Amb)}\ \frac{P \xrightarrow{amb\, n_B} P'}{(\nu u)P \xrightarrow{amb\, n_B} P'}\ (n_B \neq u)$$

$$(\tau\text{-Sloc})\ \frac{P \xrightarrow{sloc1(z)} \nu\tilde{p}\langle P'\rangle P''' \quad P' \xrightarrow{sloc(z)} P'' \quad Q \xrightarrow{amb\, n_B} Q'}{P \mid Q \xrightarrow{\tau} \nu\tilde{p}(P''\{z \leftarrow n_B\} \mid P''') \mid Q}$$

Table 7.5: SOS rules for *sloc*

ambient names in P. We introduce a new action $sloc1(z)$, and by rule Sloc1 in Table 7.5, we obtain

$$m_A[sloc(x).P_1 \mid P_2] \xrightarrow{sloc1(z)} \langle m_A[P_1 \mid P_2]\rangle 0$$

We get $m_A[sloc(x).P_1 \mid P_2] \mid P_3 \xrightarrow{sloc1(z)} \langle m_A[P_1 \mid P_2]\rangle(0 \mid P_3)$ by Par-Sloc1, where $z \notin fn(P_3)$, and by Res-Sloc1 we obtain

$$\nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3) \xrightarrow{sloc1(z)} \nu\tilde{p}\langle m_A[P_1 \mid P_2]\rangle(0 \mid P_3), \text{ where } z \notin \tilde{p} \qquad \text{(C)}$$

The transition C corresponds to the first premise of τ-Sloc. By this transition the agent $m_A[sloc(x).P_1 \mid P_2]$ enquires the name of its sibling.

By Act-Sloc, the term $sloc(x).P_1$ performs an action $sloc(z)$. By this rule z is a fresh variable that doesn't appear in P_1, and all occurrences of variable x in P_1 are replaced by z. Now, using Act-Sloc the simplest transition of $sloc(z)$ action induced by m_A in $m_A[sloc(x).P_1 \mid P_2]$ is $sloc(x).P_1 \overset{sloc(z)}{\longrightarrow} P_1\{x \leftarrow z\}$. By using Par-Sloc we have

$$sloc(x).P_1 \mid P_2 \overset{sloc(z)}{\longrightarrow} P_1\{x \leftarrow z\} \mid P_2, \text{ where } z \notin fn(P_2) \tag{D}$$

The transition D corresponds to the second premise of τ-Sloc. This transition shows that when $sloc(z)$ action is performed, all the occurrences of variable x in process P_1 are replaced by z and the process P_2 remains unchanged because $z \notin fn(P_2)$. Moreover, to achieve a τ- transition of $\nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3) \mid n_B[R]$, we define a new action $amb\ n_B$ for the ambient $n_B[R]$, and by Sib-Amb we obtain

$$n_B[R] \overset{amb\ n_B}{\longrightarrow} R \tag{E}$$

Transition E corresponds to the premise of τ-Sloc.
Since we have derived C, D and E, by the application of τ-Sloc we obtain finally

$$\begin{aligned}&\nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3) \mid n_B[R] \overset{\tau}{\rightarrow}\\ &\nu\tilde{p}(m_A[P_1\{x \leftarrow z\} \mid P_2]\{z \leftarrow n_B\} \mid P_3) \mid n_B[R].\end{aligned}$$

Since z does not appear free in P_2 by rules for substitution, the target of this transition becomes

$$\nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3) \mid n_B[R] \overset{\tau}{\rightarrow} \nu\tilde{p}(m_A[P_1\{x \leftarrow n_B\} \mid P_2] \mid P_3) \mid n_B[R]$$

7.3.1 Applications of Ploc and Sloc

In smart indoor settings location is considered as an important entity for providing communication among various portable and static structures as, for example, in our *Path* example given in Section 5.3.1, where a system instructs an agent to move from its current location to some other location. The system takes the source and target locations, and calculates a path between the two locations. In that case the system expects to receive the source name from the moving agent. The moving agent presumably keeps the parent's name. We explain the usefulness of parent-awareness and sibling-awareness features with two examples.

Example 7.1. Parent-awareness

We illustrate the parent-awareness feature of our calculus. We extend the example given in Section 5.3.1 by introducing a new construct $ploc(x)$, that queries the parent's name of an ambient. Now, our new system is of the form:

$$\nu abc\ (sys[\overline{a}(n).b(x,y).\overline{c}(path(T,x,y))] \mid k[m_1[m_A[a(u).ploc(x).\overline{b}(x,u).c(y).y.0]] \mid n_1[n[0]]])$$

The structure of our system is similar to the structure given in Section 5.3.1, where, we consider ambient k as a building with three rooms m_1, n_1 and n which is inside n_1, and the agent m_A, where $a, b, c \in A$, is a moving ambient. Furthermore, there exists an independent system in parallel with agent k. The system sys instructs the agent m_A to move from its current location to the ambient n. The above given expression shows the sequence of actions between the system sys and the moving ambient m_A. The interaction steps between the agents are: Initially, the output system sys sends the target location n on port/channel a to the moving agent m_A. Next, the agent m_A gets the name of its parent and sends it back to the server. In this specific case, the construct $ploc(x)$ enquires the parent's name m_1 of ambient m_A. Finally, by using the function $path(T, x, y)$ calculates the path between the source and target values received, and sends it to the moving ambient m_A. Here, T represents the tree structure of the setting.

As discussed earlier the system calculates the path from the source to target locations. To do so, we wrote several functions which are given in Appendix A. The general expression for the path from source location s to the target location t in a tree T is calculated by using functions given in Appendix A, and is as follows:

$$path(T,s,t) \stackrel{def}{=} \text{Sequence(Moves(Join(Path}(s,T),\ \text{Path}(t,T),\ \text{Index(Path}(s,T),\ \text{Path}(t,T)))))$$

In this particular example, by using the above given expression the path calculated by the system from the source location m_1 to the target location n is

$$out\ m_1.in\ n_1.in\ n.$$

The sequence of transitions that completes the communication between the two agents is:

$\nu\ abc\ (sys[\overline{a}(n).b(x,y).\overline{c}(path(T,x,y))] \mid k[m_1[m_A[a(u).ploc(x).\overline{b}(x,u).c(y).y.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu\ abc\ (sys[b(x,y).\overline{c}(path(T,x,y))] \mid k[m_1[m_A[ploc(x).\overline{b}(x,n).c(y).y.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu\ abc\ (sys[b(x,y).\overline{c}(path(T,x,y))] \mid k[m_1[m_A[\overline{b}(m_1,n).c(y).y.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu\ abc\ (sys[\overline{c}(path(T,m_1,n))] \mid k[m_1[m_A[c(y).y.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu\ abc\ (sys[0] \mid k[m_1[m_A[out\ m_1.in\ n_1.in\ n.0]] \mid n_1[n[0]]]).$

The resulting transition gives the expression with ambient capabilities. The path $out\ m_1.in\ n_1.in\ n$ instructs the ambient m_A to move from current location to the target location. The transitions that show all possible ambient moves are as follows:

$\nu\ abc\ (sys[0] \mid k[m_1[m_A[out\ m_1.in\ n_1.in\ n.0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu\ abc\ (sys[0] \mid k[m_1[0] \mid m_A[in\ n_1.in\ n.0] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu\ abc\ (sys[0] \mid k[m_1[0] \mid n_1[m_A[in\ n.0] \mid n[0]]]) \xrightarrow{\tau}$

$\nu\ abc\ (sys[0] \mid k[m_1[0] \mid n_1[n[m_A[0] \mid 0]]]).$

Now, after successful transitions, m_A has moved from m_1 to n.

Example 7.2. Sibling-awareness
In this example we present a scenario where a moving agent is aware of its surrounding. In our setting mobile ambient inside a building expects to receive a target location name from an independent system running outside the building. The system is ready to transmit the required information but it needs to know the device id or name on which the requesting ambient could view the information sent by the server. In this particular case, ambient n is the target location that m_A wants to receive from sys via dev_B. To model this scenario, we extend the example given in Section 5.3.1 by introducing a construct $sloc(x)$, that finds out the sibling's name. Now, our system is of the form:

$\nu\ abc\ (sys[a(x).\overline{b}(n).0] \mid k[m_1[m_A[sloc(x).\overline{a}(x).c(y).P] \mid dev[b(z).\overline{c}(z).0]] \mid n_1[n[0]]]),$

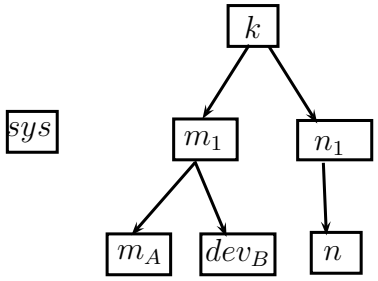


Figure 7.1: Sibling awareness

where $a, b, c \in A$ and $a, b, c \in B$. The structure of our system shares similarities with the structure given in Section 5.3.1. However, certain differences exist between the two. Firstly, an ambient dev_B exists in parallel with the mobile ambient m_A. Secondly, m_A sends its sibling's name to the system sys. Finally, the system further directs the requested information to dev_B.

The sequence of τ-transitions among three agents is:

$\nu abc\ (sys[a(x).\overline{b}(n).0] \mid k[m_1[m_A[sloc(x).\overline{a}(x).c(y).P] \mid dev[b(z).\overline{c}(z).0]] \mid$

$n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc\ (sys[a(x).\overline{b}(n).0] \mid k[m_1[m_A[\overline{a}(dev).c(y).P] \mid dev[b(z).\overline{c}(z).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc\ (sys[\overline{b}(n).0] \mid k[m_1[m_A[c(y).P] \mid dev[b(z).\overline{c}(z).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc\ (sys[0] \mid k[m_1[m_A[c(y).P] \mid dev[\overline{c}(n).0]] \mid n_1[n[0]]]) \xrightarrow{\tau}$

$\nu abc\ (sys[0] \mid k[m_1[m_A[P] \mid dev[0]] \mid n_1[n[0]]]).$

Here, using $sloc(x)$, m_A has successfully received the required information from sys via dev.

7.4 Correspondence of Transition Semantics and Reduction Semantics

In this section we explore if the transition semantics of CMC_{PCA} coincides with the reduction semantics. We consider, as in Section 4.2, a sub-calculus T'' of CMC_{PCA} that consists of all operators of CMC_{PCA} apart from the prefixing with actions (including τ) operators, the choice operator and the relabelling operator. There are "soundness" and "completeness" parts of this correspondence.

We easily have the soundness part of this correspondence between the two semantics:

Theorem 7.1. $\forall P,\ P' \in T''.\ P \to P' \Longrightarrow \exists\, Q \in T''.\ P \xrightarrow{\tau} Q \equiv P'.$

Proof. By induction where we consider cases of reductions of terms depending on the structure of the terms. □

We conjecture that the completeness part of the correspondence between the transition semantics and reduction semantics is also valid:

Conjecture 7.1. $\forall P, R \in T''.\ P \xrightarrow{\tau} R \Longrightarrow P \to R.$

The proof of Theorem 7.1 is similar to the proof of Theorem 6.2, and it relies on several auxiliary statements given in the lemma below.

Lemma 7.1.

1. $\forall P, P'.P \xrightarrow{ploc(z)} P'$, (where variable z does not appear in P) $\Longrightarrow \exists\, \tilde{p}, P_1, P_2$ *such that* $P \equiv \nu\tilde{p}\ (ploc(x).P_1 \mid P_2)$ *and* $P' \equiv \nu\tilde{p}\ (P_1\{x \leftarrow z\} \mid P_2)$, *where* $z \notin fn(P_2)$ *and* $\tilde{p}$ *is a set of ambient names private in* P.

2. $\forall P, P', P'', P'''.\ P \xrightarrow{ploc1(z)} (\nu\tilde{p})\ \langle P'\rangle P'''$ and $P' \xrightarrow{ploc(z)} P'' \Longrightarrow \exists\, P_1, P_2, P_3, n_B$ *with* $n_B \notin \tilde{p}$ *such that* $P \equiv \nu\tilde{p}(n_B[ploc(x).P_1 \mid P_2] \mid P_3), P' \equiv n_B[ploc(x).P_1 \mid P_2], P''' \equiv P_3$ *and* $P'' \equiv n_B[P_1\{x \leftarrow z\} \mid P_2]$, *where* $z \notin fn(P_2)$ *and* $\tilde{p}$ *is a set of ambient names private in* P.

3. $\forall P, P', P'', P''', Q, Q'.\ P \xrightarrow{sloc1(z)} (\nu\tilde{p})\ \langle P'\rangle P'''$ and $P' \xrightarrow{sloc(z)} P''$ and $Q \xrightarrow{amb\, n_B} Q' \Longrightarrow \exists\, P_1, P_2, P_3, m_A, n_B$ *with* $(m_A, n_B) \notin \tilde{p}$ *such that* $P \equiv \nu\tilde{p}(m_A[sloc(x).P_1 \mid P_2] \mid P_3), P' \equiv m_A[sloc(x).P_1 \mid P_2], P'' \equiv m_A[P_1\{x \leftarrow z\} \mid P_2], P''' \equiv P_3$, *where* $z \notin fn(P_2), Q \equiv n_B[Q']$*and* $\tilde{p}$ *is a set of ambient names private in* P.

7.5 Applications of $\mathrm{CMC_{PCA}}$

This section illustrates the expressiveness and usefulness of $\mathrm{CMC_{PCA}}$ by presenting two case studies of interactive shopping mall and devices automatically switching their *ON* and *OFF* modes depending on their location and the users who are using them.

7.5.1 Interactive Shopping Mall

This case study illustrates the usefulness of global communication, *push* and *pull*, and *ploc(x)* features of $\mathrm{CMC_{PCA}}$. The shopping mall consists of a number of retail outlets, clients and devices such as PDAs. To offer clients a high level of services, there is a server that delivers services to clients on requests via PDAs which are distributed inside the mall. The tree representation of the shopping mall is given in Figure 7.2, where the initial setting is given on the left-hand side and the final setting is on the right hand side. In this figure, the ambient *sm* is the shopping mall with two retail outlets m and n. For simplicity we have only one client and one PDA, represented by the ambients *client* and *pda* respectively, which are inside m.

Scenario: The client wishes to move from her current location m to a target location n inside the mall. She picks up a *pda* and sends the two locations to the *server* and requests for a path from m to n. The server generates this path as a sequence of capabilities and delivers it to the *client* via *pda*.

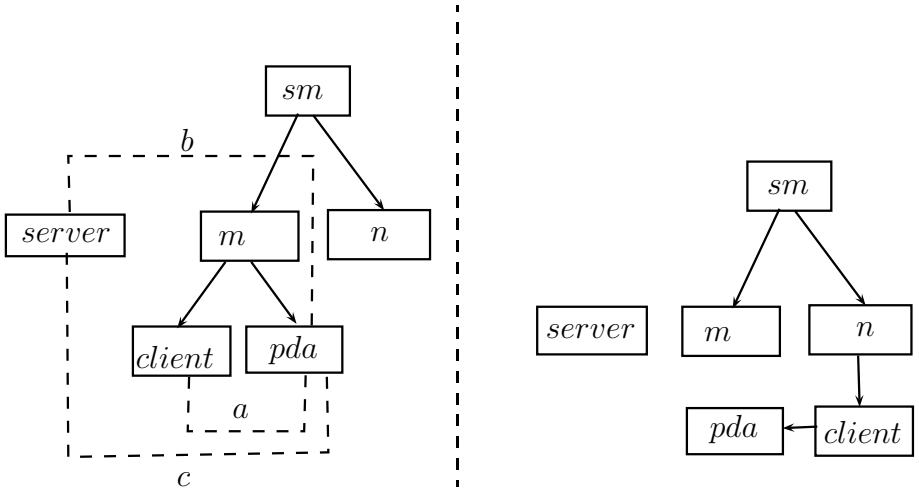


Figure 7.2: Interactive Shopping Mall settings

We define our setting as follows where, C', P' and S' are some processes:

$\nu abc\ (sm[m[client[pull(client)\ pda.ploc(x).\overline{a}(x,n).a(u).u.C'] \mid$
$pda[a(y_1,y_2).\overline{b}(y_1,y_2).c(z).\overline{a}(z).P']] \mid n[\,] \mid server[b(x_1,x_2).\overline{c}(path(T,x_1,x_2)).S'])$

The ambient *client* initiates an interaction with the PDA by its *pull(client) pda* capability. Here, T is the tree representation of the setting as in Figure 7.2. After the resulting τ-transition, *pda*, the sibling of *client*, becomes a child of *client*, namely

$\nu abc\ (sm[m[client[ploc(x).\overline{a}(x,n).a(u).u.C' \mid pda[a(y_1,y_2).\overline{b}(y_1,y_2).c(z).\overline{a}(z).P']]]$
$\mid n[\,] \mid server[b(x_1,x_2).\overline{c}(path(T,x_1,x_2)).S'])$

The only possible execution sequence from this state is $\xrightarrow{\tau_{ploc}} \xrightarrow{\tau_a} \xrightarrow{\tau_b} \xrightarrow{\tau_c} \xrightarrow{\tau_a} S''$, for some S''. In this sequence *client* acquires parent's name by $ploc(x)$ and sends her source and the target locations to *server* via a. The *server* in response calculates the *path(T,m,n)* between the two locations and delivers it back to the *client*. In this particular case, the path calculated from m to n is *out m.in n*. Now the system has the form

$$S'' \equiv \nu abc\ (sm[m[client[out\ m.in\ n.C' \mid pda[P']]] \mid n[\,] \mid server[S']).$$

After executing *out m.in n* the final state of the system becomes

$$\nu abc\ (sm[m[\,] \mid n[client[C' \mid pda[P']]]] \mid server[S']),$$

and is represented on the right hand side of Figure 7.2.

7.5.2 Devices Automatically Switching Mode

In this example we consider a smart *PDA* that automatically switches its *ON* and *OFF* mode depending on its location and the user who is using it. For example, the *PDA* is in *ON* mode if the owner is holding it, and it switches to *OFF* mode if the owner puts it down, or if any unknown user picks it up. Assume that agent *Bob* is an authorised user of the device *PDA*. The device switches to *ON* mode when *Bob* is holding it, and goes to *OFF* state otherwise.

We model the smart *PDA* as an ambient named *pda* as follows

$$PDA \stackrel{def}{=} pda[P_{pda}],$$

where P_{pda} is a process specifying the behaviour of the device PDA, namely,

$$P_{pda} \stackrel{def}{=} ploc(x).(if\,(x = bob)\ then\ \overline{on}.P_{pda}\ else\ \overline{off}.P_{pda})$$

Ambient *pda* enquires for its parent ambient name by the virtue of its $ploc(x)$ capability. We start specifying the behaviour of agent *Bob* as follows:

$$\begin{array}{rcl} P_{bob} & \stackrel{def}{=} & pull(bob)\ pda.P'_{bob} \\ P'_{bob} & \stackrel{def}{=} & push(bob)\ pda.P_{bob} \end{array}$$

Processes P_{bob} and P'_{bob} specify the behaviour of agent *Bob* w.r.t its pick and drop capabilities. The agent *Bob* represented as an ambient *bob* with its *pull* capability is modelled as follows:

$$\begin{array}{rcl} Bob & \stackrel{def}{=} & bob[P_{bob}] \\ & \equiv & bob[pull(bob)\ pda.P'_{bob}] \end{array}$$

Now we define Bob' to specify the behaviour of the agent *Bob* along with the *PDA* he is carrying with him, namely

$$\begin{array}{rcl} Bob' & \stackrel{def}{=} & bob[P'_{bob} \mid pda[P_{pda}]] \\ & \equiv & bob[push(bob)\ pda.P_{bob} \mid pda[P_{pda}]] \end{array}$$

Here ambient *bob* may push out the ambient *pda* by the virtue of its $push(bob)\ pda$ capability.

Similarly, we define agent *Nina* as follows:

$$\begin{aligned} P_{nina} &\stackrel{def}{=} pull(nina)\ pda.P'_{nina} \\ P'_{nina} &\stackrel{def}{=} push(nina)\ pda.P_{nina} \end{aligned}$$

Here P_{nina} and P'_{nina} are process specifying the *pull* and *push* capabilities of agent *Nina*. The agent *Nina* represented as an ambient *nina* with its *pull* capability is modelled as follows:

$$\begin{aligned} Nina &\stackrel{def}{=} nina[P_{nina}] \\ &\equiv nina[pull(nina)\ pda.P'_{nina}] \end{aligned}$$

Next we define *Nina'* to show the behaviour of *Nina* while holding the PDA.

$$\begin{aligned} Nina' &\stackrel{def}{=} nina[P'_{nina} \mid pda[P_{pda}]] \\ &\equiv nina[push(nina)\ pda.P_{nina} \mid pda[P_{pda}]] \end{aligned}$$

The ambient *nina* may push out the ambient *pda* by the virtue of its $push(nina)\ pda$ capability.

Overall we model the three corresponding ambients *bob*, *nina* and *pda* by composing them in parallel.

$$Bob \mid PDA \mid Nina \equiv bob[pull(bob)\ pda.P'_{bob}] \mid pda[P_{pda}] \mid nina[pull(nina)\ pda.P'_{nina}]$$

The ambients *bob* and *nina* may pick the device *pda* by the virtue of their *pull* capabilities. The device automatically switches its *ON* and *OFF* modes depending on the user holding it. The ambient *pda* by the virtue of its $ploc(x)$ capability identifies its owner and changes its modes automatically. The transition graph representing the parallel composition of the three agents is given in Figure 7.3.

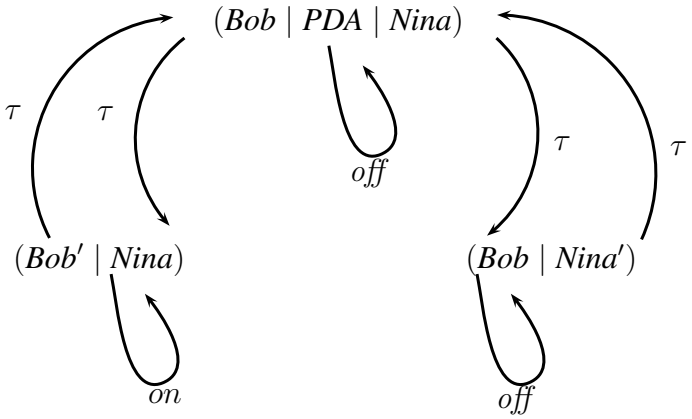


Figure 7.3: Devices switching ON/OFF modes automatically

7.6 Conclusion

In this chapter we have extended further CMC_{P} by including a basic mechanism of context awareness via primitives that bind the name of parent or sibling ambient in a process. We have also proposed operational semantics using the notions of concretions and lookahead in the SOS rules. The operational semantics has been proved sound with respect to the standard semantics. We conjecture that the operational semantics is complete with respect to the standard reductions semantics. The final calculus CMC_{PCA} combines Mobile Ambients, Push and Pull ambient Calculus, Context Aware Ambients and CCS. The usefulness of CMC_{PCA} has been illustrated in a number of small case studies.

Chapter 8

Conclusion and Furture Work

This chapter summarises the work done in this thesis, which is followed by a short evaluation of the work and some directions for future research.

8.1 Thesis Summary

This thesis presents a process calculus for specifying behaviour of mobile communicating agents. We have developed a Calculus of Communication and Mobility (CMC_{PCA}), for the modelling of mobility, communication and context awareness in the setting of ubiquitous computing. The calculus contains a new form of direct and global communication similar to that in Milner's CCS. We have defined the notion of equivalence for CMC in terms of observation predicate and action transitions, and have defined two forms of barbs. We have showed that the equivalence relations defined with the two forms of barbs imply each other. The calculus also contains a basic form of context awareness mechanism that allows ambients to query their location. We present reduction semantics and labelled transition system semantics of CMC and argue that the semantics coincide. The usefulness of the calculus is illustrated by two case studies. The main contributions of each chapter are discussed briefly below.

In Chapter 3 we have revised the syntax and semantics of the calculus of Mobile Ambients (MA), and reused only the mobility part of MA and called the calculus, a Calculus of Mobility (CM). We have developed an operational semantics for CM and have showed that the operational semantics is sound with respect to the standard reduction semantics. We have shown in examples that the proposed operational semantics are not complete with respect to the standard reduction semantics for some unusual cases. These examples have helped us to develop a complete operational semantics in Chapter 4, where the SOS rules use concretions $\nu\tilde{m}\langle P\rangle Q$ as introduced

by Milner and used by [36, 37, 40]. The correspondence of the operational semantics and the reduction semantics for CM has been shown.

In Chapter 5 we have introduced a direct and global style of communication in CM, and have presented the Calculus of Mobility and Communication (CMC). The extended calculus comprises of the *in* and *out* capabilities of MA [11] and global communication like in Milner's CCS [43]. We have combined the communication primitives and SOS rules from CCS with CM. Also, we have modified the definition of ambient as $m_A[P]$, where m is the name of the ambient, A is the set of actions that m is allowed to communicate on, and P is an executing agent. We have developed a labelled transition system semantics for CMC, which inherits the SOS rules and communication primitives from [43] with an additional SOS rule for global communication between ambients. The usefulness of CMC is exemplified by a case study of intelligent hospital setting where services follow doctor while he moves around the building and deals with patients, and a number of small examples. A new form of behavioural equivalence for CMC has been defined where we show that the congruence relations of barbed bisimulation and capability barb bisimulation imply each other.

In Chapter 6 we have extended CMC by adding further mobility primitives to also model passive mobile structures in the ubiquitous computing setting. We have added the capabilities *push* and *pull* of Phillips and Vigliotti's Push and Push Ambient Calculus [54] (PAC) and thus we have obtained CMC_P. A new and first operational semantics has been developed for PAC which is proved sound and complete with respect to the standard reduction semantics. The usefulness of CMC_P is shown by a number of small examples.

In Chapter 7 we further have extended CMC_P by introducing a new form of context awareness mechanism, thus obtaining CMC_PCA. We have added *ploc* and *sloc* primitives that help ambients to have a knowledge of their parent and siblings respectively. An operational semantics for the extended calculus CMC_PCA has been developed which is sound, and we conjecture that the operational semantics is complete with respect to the reduction semantics. The usefulness of the calculus has been illustrated by the case studies of interactive shopping mall, and devices automatically switching their mode.

8.2 Evaluation

This section discusses the strengths of our thesis as well as some of the issues that arose during the research. The work in this thesis is of theoretical, and we have

started this work with the following research statements.

1. To develop a process calculus based on Mobile Ambients, its variants and other process calculi for the modelling of ubiquitous computing features, namely
   (a) physical mobility (active and passive),
   (b) global communication,
   (c) location or structure of systems,
   (d) context awareness.
2. To develop operational semantics for the proposed calculus and deriving provable results based on the operational semantics of the calculus, and to show the expressiveness of the calculus.
3. To define appropriate notions of behavioural equivalences for the calculi and to prove properties of these equivalences.

We draw the following conclusions based on the contributions of this thesis and relate them to the research statements discussed above.

1. A Calculus of Mobility and Communication
2. Operational Semantics and Corresponding Results
   (a) Operational Semantics for CM
   (b) Operational Semantics for CMC
   (c) Operational Semantics for Push and Pull
   (d) Operational Semantics for Ploc and Sloc
3. Behavioural equivalences for CMC
4. Expressiveness and usefulness of CMC_{PCA}.

The above mentioned achievements are discussed as follows.

1. A Calculus of Mobility and Communication

   We have developed a Calculus of Mobility and Communication (CMC_{PCA}), based on ambient calculi and CCS for the modelling of mobility, global communication and context awareness in the setting of ubiquitous computing. In our calculus we have modelled the structure of system and physical mobility of active and passive mobile structures by Mobile Ambients (MA) [11] and Push

and Pull Ambient Calculus (PAC) [54] after some modifications as discussed in Section 8.1. The global communication has been achieved by adding Milner's CCS style communication in CM. Finally, we have added a new form of context awareness feature in our calculus, inspired by [17, 18].

2. Operational Semantics and Corresponding Results

   (a) We have developed a new and simple operational semantics for CM which is inspired by that in [36, 37] except that they have used concretions in their operational semantics and we have developed our transition rules without using concretions. Our operational semantics is simple, and sound with respect to the standard reduction semantics. We have discovered that the semantics is not complete for certain unusual cases where ambients with the same name intend to perform *in* or *out* capabilities. The SOS rules have no ability to distinguish between the ambients with identical name and matching capabilities. This limitation has been overcome by developing a new operational semantics for CM using concretions as in [36, 37]. The new operational semantics has been proved sound and complete with respect to the standard reduction semantics.

   (b) We have also developed operational semantics for CMC, where we have added an additional Global-Com rule. This extension allows our ambients to communicate globally with ambients nested inside other ambients.

   (c) We have continued developing a new operational semantics for PAC that has been used for the modelling of passive mobile structures in our calculus. To the best of our knowledge this is the new and first operational semantics that has been developed for PAC. We have also discovered that using concretions are not enough to develop the SOS rules, therefore we have also used the notion of lookahead. We have proved that our new operational semantics is sound and complete with respect to the standard reduction semantics.

   (d) We have proposed a reduction semantics as well as an operational semantics for the context awareness feature of our calculus. In context awareness, we have used *ploc* and *sloc* primitives that help ambients to have knowledge of their parent and siblings respectively. In literature, we have not found an operational semantics for the modelling of ambients' location and surrounding. Our context awareness constructs *ploc(x)* and *sloc(x)* bear similarity with the Conversation Calculus construct *here(x)* [76, 9] that allows a process running inside a given context to access its

identity. In in Conversation Calculus conversation contexts are proposed as communication medium that controls information sharing among processes, whereas our constructs are not precisely used for only communication. Based on the information about parent or sibling name, an ambient may communicate globally or move from one location to another in an indoor setting.

3. Behavioural equivalences for CMC

   A new form of behavioural equivalence has been introduced for CMC, where observable behaviour of two processes are considered, this is inspired by [36, 37]. We have defined barbed bisimulation and congruence, and capability barbed bisimulation and congruence. In order to prove successfully that the respective congruence relations of the two forms of barbs agree, the global communication primitives are used. We have not used co-capabilities in our calculus as in [37], and hence conclude that such results could be proved without the help of co-capabilities.

4. The calculus CMC_{PCA} has been proposed with real-world applications in mind and its expressiveness and usefulness has been illustrated in several case studies and small examples, where various features (active and passive mobility, global communication and location awareness) of our calculus have been modelled using relevant constructs.

8.3 Future Work

Research is a never ending journey, it starts with a problem and solution to the problem unlocks a new question. The work in this thesis was started with the aim of investigating different formalisms for the modelling of ubiquitous and mobile computing where computing devices are available throughout the physical setting. These devices are distributed and could be mobile, and interactions among them are concurrent and often depend on the location of the devices. The work presented in this thesis is what has been produced within the given time constraints, where various features of the proposed problem domain have been formalised systematically. The initially calculus CM is extended uniformly by including *push*, *pull*, *ploc* and *sloc* to obtain the final calculus CMC_{PCA}. We now discuss several directions in which CMC_{PCA} can be further extended.

A. **Major directions**

(a) In Chapter 5 we have formalised the notion of equivalence for CMC in terms of observation predicate and action transitions ($\xrightarrow{\alpha}$). We have defined barbed bisimulation and capability barbed bisimulation and proved that the congruence relations of barbed bisimulation and capability barbed bisimulation for *move* n_B coincide. We conjecture that the congruence relations of the two forms of barbs for the other capabilities of CMC agree. This needs a proof. The characterisation of barbed bisimilarity may be achieved by defining a version of early bisimulation as in [36, 37].

(b) To investigate a suitable methodology to integrate ubiquitous data with CMC_{PCA}. In general, databases are fixed and static, but in the ubiquitous computing settings devices are distributed, and interactions are concurrent and dynamic. Therefore, data is generated continuously and there is live data streaming [20, 19, 21]. It is a challenging task to investigate a suitable formalism for modelling ubiquitous data streaming, and combining them with the traditional Relational Database modelling formalisms.

(c) Some events of the ubiquitous computing systems are time sensitive and expect responses without delay. The behaviour of such systems depends on the instance of time at which an event is generated or input is received from an external environment. Timed versions of various process calculi have been presented [81, 66, 3] in order to deal with time dependent behaviour of systems in different ways. For example, as in Timed CCS, *TCCS* for short, a real time system has been modelled by adding time to CCS [81]. The syntax of TCCS is the same as CCS except that the action prefix of CCS has been modified to $\mu@t.P$ where t is a time variable. Timed version of Mobile Ambients have been presented in [1, 2] to model efficient resource allocation and immediate responses to time critical events. Therefore, we also envision that adding time to CMC_{PCA} could make the calculus more realistic. It could also lead to timed versions of behavioural equivalences.

(d) Security is also an interesting area for future work in the setting of ubiquitous computing. Some of the risks associated with the ubiquitous computing settings are addressed in [67]. Security is crucial for most of the computer science applications. However, a lot of research in recent years has been directed at solving security problems raised by distributed sys-

tems and in ambient calculi, as in [53]. Ubiquitous computing is an omnipresent and devices that do not look like computers are endowed with computing capabilities. The challenge to protect the security of the ubiquitous computing devices is still an open research topic.

B. **Minor directions**

(a) In Chapter 7 there is a conjecture that the operational semantics developed for context awareness is complete with respect to the standard reduction semantics. This result needs a proof.

(b) Chapter 5 presents CMC, where ambients mobility is modelled by the *in* n_B and *out* n_B capabilities of MA, and communication is similar to that in Milner's CCS. We present an LTS for CMC and show that the operational semantics of the mobility part of the calculus is sound and complete with respect to the standard reduction semantics. Likewise, we conjecture that CMC not only extends MA but also CCS, namely if we take a CMC term and assume that there are no ambients used in the term, then the term works like a CCS agent.

(c) In past few years, extensive theoretical discoveries on the ambient calculi have been made, however there has been relatively smaller amount of research carried out in developing real mobile applications based on ambient calculi. We believe that the implementation of these theoretical aspects could be an interesting research direction, for example, as in [53] a distributed abstract machine for boxed ambient calculus has been implemented.

Appendix A

A.1 Operations

In this section we present various useful operations on trees, that could be used to calculate a path between two given nodes, such as, a path from a source node to a target node in a given tree.

1. Parent(n, T):
   This function returns parent of node n in tree T. If n is the root node then it returns null node which indicates that we are navigating off the tree.

2. Root(T):
   This function returns the root node in a given tree T.

3. Append(n, $list$):
   This function appends an element n to the left of a list $list$.

4. Path(n, T): $list$:
   This function returns the path from a current node n to the root in a given tree T. The current node can be a source node or a target node.

5. Index(l_1, l_2):
   This function takes the two lists that is output from the function Path(n, T), and returns a list of two elements. The two elements point to the index of the first common elements of the first and the second list respectively.

6. Join($list_1$, $list_2$, $list_3$):
   This function takes three lists $list_1$, $list_2$ and $list_3$ as arguments from the functions Path($source$, T), Path($target$, T) and Index(l_1, l_2) respectively, and returns a complete path (list of nodes from the source node to the target node) after joining the two lists using the index points.

7. Moves($list$):
   This function takes a list as an argument from the Join function and returns the list of nodes with appropriate *in* and *out* prefixes.

8. Sequence($list$):
   Takes a list of nodes as an argument from the Move function and returns a sequence of nodes with a dot (.) between the two consecutive nodes.

A.2 Examples

Next, we present set of functions designed in terms of above given operations.

We write a function to take a tree and list the labels of all the nodes (path) from a node n to the root of a tree T. This function is given in figure A.1.

```
function Path(n, T) : list
{
  Bool rootFound = False;
  list[ ];
  list = Append(n, list);
  While(! rootFound)
  {
    n = Parent(n, T);
    if(n == root(T))
    {
    rootFound = True;
    }
  list = Append(n, list);
  }
reverse(list);
return list;
}
```

Figure A.1: Function returning a path from a node to the root of a tree

To show that how this works, we consider a tree structure given in figure A.2.

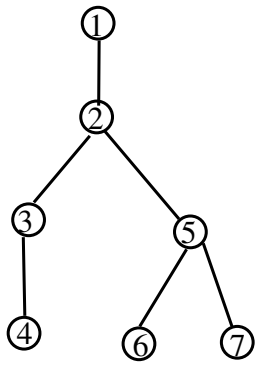


Figure A.2: Tree structure

To find the path between two nodes of a tree structure we assume node (4) and node (6) as source and target nodes respectively.

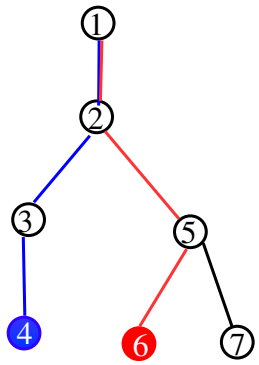


Figure A.3: Tree representing source and target nodes

The blue edges shows the path from source node (4) to the root node (1) of the given tree, while the red edges shows the path from target node (6) to the root node (1) of the given tree, and this is given as below:

$$list1 = [4, 3, 2, 1] \text{ and } list2 = [6, 5, 2, 1].$$

```
function Index(list1, list2) : list
{
  Bool match = False;
  int index1 = 0;
  int index2 = 0;
  list[ ];
  for(int i = 0; i < list1.length − 1; i++)
  {
    for(int j = 0; j < list2.length − 1; j++)
    {
    if(lis1[i] == list2[j])
    {
      index1 = i;
      index2 = j;
      match = True;
    }
    if(match);
      {
      break;
      }
list = Append(index1, Append(index2, list));
  }
return list;
}
```

Figure A.4: Function returning first common nodes of the two given lists

The function given in figure A.4 returns a list of two elements representing the indices of first common elements of the two lists. In our case indices of the first common nodes are 2,2. The green color node with a label 2 shown in figure A.5 is the first common node of the two lists.

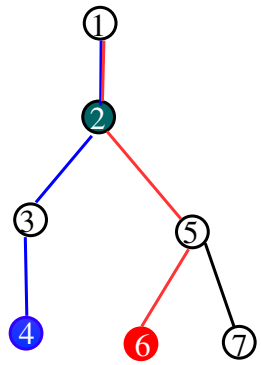


Figure A.5: First common node of the two lists

In Figure A.6 we write a function that joins the two lists in such a way that it lists the labels of all the nodes from the source (4) to the target node (6) of the given tree T.

```
function Join(list1, list2, list3) : list
{
  int index1 = head(list3);
  int index2 = head(tail(list));
  path[ ];                                    //empty list
  for(int i = 0; i < index2; i++)
    {
path = Append(list2[i], path);
    }
  for(int j = index1 − 1; j > 0; j − −)
    {
path = Append(list1[j], path);
    }
return list;
}
```

Figure A.6: Joining two lists

In Figure A.7, we give a function that takes a list and returns the same list with nodes prefixed by appropriate *in* or *out* moves (capabilities).

```
function Moves(list) : list
{
  move[ ];
  for(int i = 0; i < list.length − 1; i++)
  {
    if(list[i + 1] == Parent(Head(list), T))
    {
      move = Append(out list[i], move);
    }
    else if(Head(list) == Parent(list[i + 1], T))
    {
      move = Append(in list[i + 1], move);
    }
list = Tail(list)
  }
reverse list;
return list;
}
```

Figure A.7: Prefixing *in* and *out* moves

Bibliography

[1] Aman, B. and Ciobanu, G. Timers and proximities for mobile ambients. In *Computer Science - Theory and Applications*, volume 4649 of *Lecture Notes in Computer Science*, pages 33–43. Springer-Verlag, 2007.

[2] Aman, B. and Ciobanu, G. Timed mobile ambients for network protocols. In *Formal Techniques for Networked and Distributed Systems*, volume 5048 of *Lecture Notes in Computer Science*, pages 234–250. Springer-Verlag, 2008.

[3] Baeten, J. C. M and Middelburg, C. A. *Process Algebra with Timing*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 2002.

[4] Bergstra, J. A., and Klop, J. W. Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137, 1984.

[5] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A and Riboni, D. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180, 2010.

[6] Bugliesi, M., Castagna, G. and Crafa, S. Boxed ambients. In *Theoretical Aspects of Computer Software*, *Lecture Notes in Computer Science*, pages 38–63. Springer-Verlag, 2001.

[7] Bugliesi, M., Crafa, S., Merro, M. and Sassone, V. Communication interference in mobile boxed ambients. In *Foundations of Software Technology and Theoretical Computer Science*, volume 2556 of *Lecture Notes in Computer Science*, pages 71–84. Springer-Verlag, 2002.

[8] Bugliesi, M., Crafa, S., Merro, M. and Sassone, V. Communication and mobility control in boxed ambients. *Information and Computation*, 202(1):39–86, 2005.

[9] Caires, L. and Vieira, H. T. Analysis of service oriented software systems with the conversation calculus. *Lecture Notes in Computer Science*, pages 6–33. Springer-Verlag, 2010.

[10] Cardelli, L. and Gordon, A. D. Mobile ambients. In M. Nivat, editor, *Proceedings of Foundations of Software Science and Computation Structures*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer-Verlag, 1998.

[11] Cardelli, L., and Gordon, A. D. Mobile ambients. *Theoretical Computer Science*, 240:177–213, 2000.

[12] Cardelli, L., Ghelli, G. and Gordon, A. D. Types for the ambient calculus. *Information and Computation*, 177:160–194, 2002.

[13] de Frutos-Escrig, D. and Alonso, O. M. and Rosa-Velardo, F. Ubiquitous systems and petri nets. In *Computational Science and Its Applications - ICCSA 2005, International Conference, Singapore, May 9-12, 2005, Proceedings, Part II*, volume 3481 of *Lecture Notes in Computer Science*, pages 1156–1166. Springer-Verlag, 2005.

[14] Desel, J. and Reisig, W. Place or Transition Petri Nets. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, pages 122–173. Springer-Verlag, 1996.

[15] Fencott, C. *Formal Methods for Concurrency*. International Thomson, 1996.

[16] Ferrari, G. L., Montanari, U. and Tuosto, E. An LTS semantics of ambients via graph synchronization with mobility. In *ICTCS*, volume 2202 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2001.

[17] François S., Cau, A. and Zedan, H. CCA: A calculus of context-aware ambients. In *AINA Workshops*, pages 972–977. IEEE Computer Society, 2009.

[18] François S., Cau, A. and Zedan, H. The calculus of context-aware ambients. *Journal of Computer and System Sciences*, 77(4):597–620, 2011.

[19] Franklin, M. J. Challenges in ubiquitous data management. In *Informatics - 10 Years Back. 10 Years Ahead.*, volume 2000 of *Lecture Notes in Computer Science*, pages 24–33. Springer-Verlag, 2001.

[20] Gaber, M. M., Gama, J., Krishnaswamy, S., Gomes, J. and Stahl, F. Data stream mining in ubiquitous environments: state-of-the-art and current directions. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 4(2):116–138, 2014.

[21] Gaber, M. M., Zaslavsky, A. and Krishnaswamy, S. Mining data streams: A review. *SIGMOD Rec.*, 34(2):18–26, June 2005.

[22] Gottmann, S. and Nachtigall, N. and Hoffmann, K. On modelling communication in ubiquitous computing systems using algebraic higher order nets. *ECEASST*, 51, 2012.

[23] Gul, N. *Modelling Ubiquitous Computing.* Master's thesis, University of Leicester, 2010.

[24] Han, S. and Youn, H. Y. Modeling and analysis of time-critical context-aware service using extended interval timed colored petri nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 42(3):630–640, 2012.

[25] Han, S. and Youn, H. Y. Petri net-based context modeling for context-aware systems. *Artificial Intelligence Review*, 37(1):43–67, 2012.

[26] Heckel, R. Graph transformation in a nutshell. *Electronic Notes in Theoretical Computer Science 148*, page 187198, 2006.

[27] Heckel, R. Tutorial introduction to graph transformation. In *Graph Transformations, 4th International Conference, ICGT 2008, Leicester, United Kingdom, September 7-13, 2008. Proceedings*, volume 5214 of *Lecture Notes in Computer Science*, pages 458–459. Springer-Verlag, 2008.

[28] Hennessy, M. *The Semantics of Programming Languages.* Wiley, 1993.

[29] Hoare, C. A. R. Communicating sequential processes. *Communications of ACM*, 21(8):666–677, 1978.

[30] Hoare, C. A. R. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[31] Hoareau, C. and Satoh. I. Modeling and processing information for context-aware computing: A survey. *New Generation Computation*, 27(3):177–196, 2009.

[32] Hoffmann, K. and Mossakowski, T. Algebraic higher-order nets: Graphs and petri nets as tokens. In *Recent Trends in Algebraic Development Techniques, 16th International Workshop, WADT*, volume 2755 of *Lecture Notes in Computer Science*, pages 253–267. Springer-Verlag, 2002.

[33] Leonhardt, U. *Supporting Location-Awareness in Open Distributed Systems.* PhD thesis, Imperial College London, 1998.

[34] F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Principles of Programming Languages*, pages 352–364. ACM, 2000.

[35] Levi, F. and Sangiorgi, D. Mobile safe ambients. *ACM Transactions on Programming Languages and Systems*, 25(1):1–69, 2003.

[36] Merro, M. and Hennessy, M. Bisimulation congruences in safe ambients. In *Principles of Programming Languages*, pages 71–80. ACM, 2002.

[37] Merro, M. and Hennessy, M. A bisimulation-based semantic theory of safe ambients. *ACM Transactions on Programming Languages and Systems*, 28(2):290–330, 2006.

[38] Merro, M. and Nardelli, F. Z. Bisimulation proof methods for mobile ambients. In *ICALP*, *Lecture Notes in Computer Science*, pages 584–598. Springer-Verlag, 2003.

[39] Merro, M. and Nardelli, F. Z. Behavioural theory for mobile ambients. In *IFIP TCS*, pages 549–562, 2004.

[40] Merro, M. and Nardelli, F. Z. Behavioral theory for mobile ambients. *Journal of ACM*, 52(6):961–1023, 2005.

[41] Merro, M. and Sassone, V. Typing and subtyping mobility in boxed ambients. In *CONCUR*, volume 2421 of *Lecture Notes in Computer Science*, pages 304–320. Springer-Verlag, 2002.

[42] Milner, R. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[43] Milner, R. *Communication and Concurrency*. Prentice Hall Europe, 1989.

[44] Milner, R. *Communicating and Mobile Systems: The π-calculus*. Cambridge University Press, 1999.

[45] Milner, R. Bigraphical reactive systems. volume 2154 of *Lecture Notes in Computer Science*, pages 16–35. Springer-Verlag, 2001.

[46] Milner, R. Bigraphs as a model for mobile interaction. *Lecture Notes in Computer Science*, pages 8–13. Springer-Verlag Berlin Heidelberg, 2002.

[47] Milner, R. Bigraphs and their algebra. *Electronic Notes Theoratical Computer Science*, 209:5–19, 2008.

[48] Milner, R. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[49] Milner, R., Parrow, J., and Walker, D. A calculus of mobile processes, part I. *Information and Computation*, 100(1):1–40, 1992.

[50] Milner, R., Parrow, J., and Walker, D. A calculus of mobile processes, part II. *Information and Computation*, 100(1):41–77, 1992.

[51] Mousavi, M. R., Phillips, I. C. C., Reniers, M. A and Ulidowski, I. Semantics and expressiveness of Ordered SOS. *Information and Computation*, 207(2):85–119, 2009.

[52] Olaru, A. and Gratie, C. Agent-based, context-aware information sharing for ambient intelligence. *International Journal on Artificial Intelligence Tools*, 20(6):985–1000, 2011.

[53] Phillips, A., Yoshida, N. and Eisenbach, S. A distributed abstract machine for boxed ambient calculi. In *ESOP*, pages 155–170, 2004.

[54] Phillips, I. C. C. and Vigliotti, M. G. On reduction semantics for the push and pull ambitent calculus. In *IFIP TCS*, pages 550–562. Kluwer, 2002.

[55] Plotkin, G. D. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

[56] Poslad, S. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. Wiley, 2009.

[57] A. Ranganathan and A. H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, 2003.

[58] Ranganathan, A. and Campbell, R. H. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, 2003.

[59] Riboni, D. and Bettini, C. Context-aware activity recognition through a combination of ontological and statistical reasoning. In *Ubiquitous Intelligence and Computing, 6th International Conference, UIC 2009, Brisbane, Australia, July 7-9, 2009. Proceedings*, volume 5585 of *Lecture Notes in Computer Science*, pages 39–53. Springer-Verlag, 2009.

[60] Rosa-Velardo, F. and Alonso, O. M. and de Frutos-Escrig, D. Mobile synchronizing petri nets: A choreographic approach for coordination in ubiquitous systems. *Electr. Notes Theor. Comput. Sci.*, 150(1):103–126, 2006.

[61] Rosa-Velardo, F. and de Frutos-Escrig, D. and Alonso, O. M. Replicated ubiquitous nets. In *Computational Science and Its Applications - ICCSA 2006, International Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part IV*, volume 3983 of *Lecture Notes in Computer Science*, pages 158–168. Springer-Verlag, 2006.

[62] Satoh, I. Location-based services in ubiquitous computing environments. *Service-Oriented Computing - ICSOC 2003*, pages 527–542, 2003.

[63] Satoh, I. A location model for pervasive computing environments. In *Pervasive Computing and Communications, IEEE International Conference on*, pages 215–224. IEEE Computer Society, 2005.

[64] Satoh, I. A spatial model for ubiquitous computing services. In *IEICE Transactions on Communications*, volume E88-B, pages 923–931, 2005.

[65] Satoh, I. A location model for smart environments. *Pervasive and Mobile Computing*, 3(2):158–179, 2007.

[66] Schneider, S. *Concurrent and Real Time Systems: The CSP Approach*. John Wiley & Sons, Inc., New York, USA, 1st edition, 1999.

[67] Stajano, F. *Security for Ubiquitous Computing*. John Wiley and Sons, 2002.

[68] Ulidowski, I. Equivalences on observable processes. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science, Santa Cruz, California, USA, June 22-25, 1992*, pages 148–159. IEEE Computer Society, 1992.

[69] Ulidowski, I. *Local Testing and Implementable Concurrent Processes*. PhD thesis, Imperial College London, 1994.

[70] Ulidowski, I. Communication and concurrency. Lecture Notes, University of Leicester, 2014.

[71] Ulidowski, I. and Phillips, I. C. C. Formats of ordered SOS rules with silent actions. In *TAPSOFT*, *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, 1997.

[72] Ulidowski, I. and Phillips, I. C. C. Ordered SOS process languages for branching and eager bisimulations. *Information and Computation*, 178(1):180–213, 2002.

[73] Ulidowski, I. and Yuen, S. Process languages with discrete relative time based on the ordered SOS format and rooted eager bisimulation. *Journal of Logic and Algebraic Programming*, 60-61:401–460, 2004.

[74] Valk, R. Concurrency in communicating object petri nets. In *Concurrent Object-Oriented Programming and Petri Nets, Advances in Petri Nets*, volume 2001 of *Lecture Notes in Computer Science*, pages 164–195. Springer-Verlag, 2001.

[75] Velardo, F. R. and de Frutos-Escrig, D. Symbolic semantics for the verification of security properties of mobile petri nets. In *Automated Technology for Verification and Analysis, 4th International Symposium, ATVA 2006, Beijing, China, October 23-26, 2006*, volume 4218 of *Lecture Notes in Computer Science*, pages 461–476. Springer-Verlag.

[76] Vieira, H. T., Caires, L. and Seco, J. C. The conversation calculus: A model of service-oriented computation. volume 4960 of *Lecture Notes in Computer Science*, pages 269–283. Springer-Verlag, 2008.

[77] Vigliotti, M. G. *Reduction Semantics for Ambient Calculi*. PhD thesis, Imperial College London, 2004.

[78] Weiser, M. The computer for the 21st century. *Scientific American*, February 1991.

[79] Weiser, M. Some Computer Science Issues in Ubiquitous Computing. In *Communications of the ACM*, volume 36, pages 75–84, 1993.

[80] Winskel, G. *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, Cambridge, MA, 1993.

[81] Yi, W. CCS + Time = an interleaving model for real time systems. In *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, volume 510 of *Lecture Notes in Computer Science*, pages 217–228. Springer-Verlag, 1991.