AN EXPANDED AMBIENT CALCULUS WITH π-CALCULUS EMBEDDING

ELSA GUNTER, ERIC PETERSON

Contents

1. Introduction

Formal methods is the branch of computer science most concerned with the mathematical formalization of computational frameworks and the specification of programs and programming languages. The formulation of computability most familiar to an undergraduate is most likely the Turing machine, which models computation in a way intuitive to the layperson; a Turing machine consists of a malleable tape of input information and a set of transition rules that guide a reading and writing head back and forth along the tape.

While plenty fruitful and easy to come to grips with, the Turing machine concept is mathematically unsatisfying for a variety of reasons, which are addressed more appropriately by the second most familiar formalization: Church and Kleene's λ-calculus. The λ-calculus is specified by a grammar, which dictates what strings of symbols are valid λ-calculus expressions, an equivalence relation called "structural equivalence" that dictates when two λ-calculus expressions should be considered to be the same, and a set of transition rules that embody the actual computations that one can perform with these expressions.

The λ-calculus provides an appealing mathematical framework in which to study computability theory, but it is not the only one. The so-called Church-Turing thesis states that Turing machines and λ-calculus expressions encode equivalent computations, neither one is more expressive than the other. However, just as they have different strengths and weaknesses, we might produce other calculi with the same expressiveness as the λ-calculus but within which certain kinds of computations are easier to write down and manipulate. One topic which both the λ-calculus and Turing machines are particularly awkward about modeling is concurrent computation, a problem addressed by Milner, Parrow, and Walker's π-calculus, introduced in [4]. The π-calculus provides a quite minimalistic set of tools for modelling paired communication between independent processes over channels of communication; much of the complexity of the π-calculus arises because of the ability to transmit channel names between processes, allowing for much more flexible communication.

Widepsread modern application of concurrent devices – most notably the mere existence of the Internet – means that flexible and powerful process calculi are in high demand. The core π-calculus fails to adequately address two important issues that arise in Internet communication: secrecy of communication and broadcast communication. The spi-calculus was introduced in [1] with the intention of extending the familiar π-calculus with constructs suited for secrecy, and various models of broadcast communication were developed, reflecting the innumerable differing standards for broadcast communication itself. Recently, in [3], a calculus extending

the ambient family of calculi was introduced that addressed both these issues simultaneously, and addtionally allowed a more flexible notion of topology on the communication graph.

One notable design decision was that the broadcast sends did not wait for any kind of confirmation of receipt, a good model for honest physical communication but awkward for the kind of communicaton that the π-calculus supports, which is itself sometimes useful. The same authors produced in [2] an embedding of the π-calculus into their SBA calculus, i.e., an inductively defined map from π-calculus expressions into SBA configurations such that various properties of the expressions are preserved (exactly which properties and what "preserved" means we will make clear later). In this paper, we produce another ambient calculus in the spirit of [3] that resolves some of that calculus' issues with the communication and movement of scoped ambients. We also produce an embedding of the π-calculus in the spirit of [2] and prove one direction of two kinds of process similarity preservation: structural similarity and bisimilarity.

2. Semantics

In this paper, we will give semantics for the relevant calculi in a framework referred to as "labeled transition semantics." Labeled transition semantics for a system are given in three parts: a grammar, structural equivalence, and labeled transition rules, where a rule is an ordered triple consisting of two words generated by the grammar and a "label," which we think of as a symbol living in some arbitrary but fixed set of labels. The grammar is meant to generate the set of legal strings in our language. Structural equivalence captures two conceptual elements of sameness: when two strings are identical up to a selection of syntax, like α-equivalence in the λ-calculus, and when strings are equivalent via some more complex semantic construct analogous to the heating and cooling of CHAM. Finally, the transitions are used for one-way transitions, and their labels are used to broadcast the effects of these events that are "world-observable." In the context of our calculi, this typically means some kind of send or receive event, since information transmitted from one machine to another can be listened to, or observed.

2.1. π-calculus semantics. The core of the π-calculus rests in a collection of processes undergoing parallel execution, along with a collection of channel names that allow for synchronized communication between pairs of processes. We also provide a nil process to signify the end of execution, replication to allow for unrestricted reproduction of a piece of code, and a tool called ν that allows us to generate channel names that are guaranteed to be unused.

First, we introduce the necessary syntactic constructs for our language. In what follows, P, Q, and R are metavariables that range over the set of processes, x and y are metavariables that take values in a set of variables, and m is a metavariable that takes values in a set of channel names.

$P ::= \mathbf{0}$	(the nil process)
$\mid P \mid\mid Q$	(parallel composition)
$\mid \nu x.P$	(scoped variable creation)
$\mid \overline{m}\langle n\rangle.P$	(send n on channel m)
$\mid m(x).P$	(receive x from channel m)
$\mid !P$	(replicate P)

For the semantics we intend to put on this language, this grammar is too fine-grained; there are a number of superficially different strings that ought to have the same semantic meaning, and so we introduce an equivalence relation, called structural equivalence and denoted $\equiv$, to quotient the language down to what we want. In particular, there is a notion of α-equivalence inherited from the usual definition in the context of the λ-calculus, which says that we can rename bound variables to whatever we like so long as we fulfill two conditions:

(1) We must similarly rename all instances of that same variable, and
(2) We do not select a name that conflicts with the name of some other variable that lives below the binding construct in the parse tree (refered to as "variable capture").

It remains then to give rules for what π-calculus constructs bind variables. The two relevant constructs are $\overline{M}\langle x\rangle.P$, which binds the variable x to the message received on M throughout P, and $\nu x.P$, which binds the variable x to a fresh channel name throughout P.

$$
\begin{aligned}
P &\equiv Q \text{ when } P \text{ and } Q \text{ are } \alpha\text{-equivalent} \\
!P &\equiv P||!P \\
P||Q &\equiv Q||P \\
P||\mathbf{0} &\equiv P \\
P||(Q||R) &\equiv (P||Q)||R \\
(\nu x.P)||Q &\equiv \nu x.(P||Q) \text{ when } x \text{ is not free in } Q \\
\nu x.\nu y.P &\equiv \nu y.\nu x.P \\
\nu x.\mathbf{0} &\equiv \mathbf{0}
\end{aligned}
$$

We are nearly in a position to actually describe how the syntactic constructs interact with a labeled transition semantics – before that, we need a set of labels to draw from. We will use ρ for our metavariable that binds to labels, and it will take values in the following set:

$$\rho \in \{\overline{m}\langle n\rangle \mid m \text{ a channel name, } n \text{ a message } \} \cup \{m(n) \mid m \text{ is a channel name, } n \text{ a message } \} \cup \{\tau\}.$$

Now, finally, we can give a labeled transition semantics for the π-calculus, with rules enumerated in the following table:

$$\text{ParallelExecution:}\frac{P \xrightarrow{\rho} P'}{P||Q \xrightarrow{\rho} P'||Q}, \quad \text{Send:}\frac{}{\overline{m}\langle n\rangle.P \xrightarrow{\overline{m}\langle n\rangle} P}, \quad \text{Receive:}\frac{}{m(x).P \xrightarrow{m(n)} P[n/x]},$$

$$\text{Quotient:}\frac{P \equiv P' \quad P' \xrightarrow{\rho} Q' \quad Q' \equiv Q}{P \xrightarrow{\rho} Q}, \quad \text{Bind:}\frac{P \xrightarrow{\rho} P' \quad x \text{ does not appear in } \rho}{\nu x.P \xrightarrow{\rho} \nu x.P'},$$

$$\text{Communicate:}\frac{P \xrightarrow{m(n)} P' \quad Q \xrightarrow{\overline{m}\langle n\rangle} Q'}{P||Q \xrightarrow{\tau} P'||Q'}.$$

The transition rule and structural equivalence rule for lifting transitions outside of a ν operator are worth discussing because they allow for private communication among the processes. In particular, the channel name x bound in P by $\nu x.P$ can never be seen in labels outside of that ν; if two processes wish to discuss on or about a private channel, they both have to be brought inside the ν via the structural equivalence rule. Communications that do not concern the private channel, however, can occur unhindered between process both in contained in the ν or even with only one communicating party contained in the ν.

Some variants of the π-calculus also include operators for nondeterministic choice and for matching, but for brevity we omit them from our construction. We also made the choice to use early binding, as reflected in our Receive rule; for more information on this choice, its alternatives, and the π-calculus in general, the reader is directed to [4].

2.2. Broadcast semantics. For this section, we follow in the footsteps of [2]. The major differences between the π-calculus outlined above and our next calculus, an ambient calculus variant, is that our semantics are altered to allow for broadcast sends and complex network topologies. This means in particular that we have a new piece of information to keep track of: a graph of ambients, representing connectivity of the network.

The most common presentation of such an object is as a forest, but this is unsatisfying for a variety of reasons; in particular, it doesn't allow a single ambient to reside inside two other ambients simultaneously, which is exactly how devices like routers behave. To successfully model this behavior, we allow our connectivity graphs to be arbitrary directed acyclic graphs, where nodes represent ambients and an edge from ambient m to an ambient m' represents m as a child of m' – such an object is called an *ambgraph*, following [3]. Graphs of this form can be modeled linearly as a topologically sorted list of pairs $(m :: \mathcal{L})$, where m is an ambient name and $\mathcal{L}$ is its set of parents.

We now inductively define our syntactic categories as follows:

- Actions:

$$\begin{array}{ll}
\pi ::= \langle n \rangle^m & \text{(send } n \text{ on } m\text{)} \\
\mid (x)^m & \text{(receive } x \text{ from } m\text{)} \\
\mid \text{in } m & \text{(request entry into } m\text{)} \\
\mid \text{out } m & \text{(leave } m\text{)} \\
\mid \overline{\text{in}}\ m & \text{(permit } m \text{ to enter)} \\
\mid \overline{\text{in}}\ _ & \text{(permit anonymous entry)} \\
\mid \text{newamb}(m, P) & \text{(spawn a new ambient)}
\end{array}$$

- Processes:

$$\begin{array}{ll}
P ::= \text{nil} & \text{(the nil process)} \\
\mid P||Q & \text{(parallel process composition)} \\
\mid \pi.P & \text{(sequential composition)} \\
\mid !P & \text{(replication)}
\end{array}$$

- Systems:

$$\begin{array}{ll}
S ::= \text{nilsystem} & \text{(the nil system)} \\
\mid m[P] & \text{(attach } P \text{ to ambient } m\text{)} \\
\mid \nu x :: \mathcal{L}.(S) & \text{(scoped ambient creation)} \\
\mid S||S' & \text{(parallel ambient composition)}
\end{array}$$

- Configurations: $C ::= G \triangleleft S$.

As with the π-calculus, we want to define a notion of α-equivalence on our systems. Similar to the previous case, the three binding constructs in the broadcast calculus are $\langle x \rangle^m.P$, $\nu x :: \mathcal{L}.(S)$, and $\text{newamb}(x, P).Q$, which bind x through P, S, and both P and Q respectively. We use α-equivalence to define broadcast ambient structural equivalence as the congruence closure of the following ruleset:

$$\begin{aligned}
G \triangleleft S &\equiv G' \triangleleft S' \ \text{ whenever } G \triangleleft S \text{ and } G' \triangleleft S' \text{ are } \alpha\text{-equivalent} \\
G \triangleleft S &\equiv G' \triangleleft S \ \text{ whenever } G \text{ and } G' \text{ represent the same graph} \\
(S_1||S_2)||S_3 &\equiv S_1||(S_2||S_3) \\
S_1||S_2 &\equiv S_2||S_1 \\
S||\text{nilsystem} &\equiv S \\
m[\text{nil}] &\equiv \text{nilsystem} \\
m[P||Q] &\equiv m[P]||m[Q] \\
\nu x :: \mathcal{L}.(S_1||S_2) &\equiv S_1||\nu x :: \mathcal{L}.S_2 \text{ when } x \text{ does not appear unbound in } S_1 \\
\nu x :: \mathcal{L}_1.\nu y :: \mathcal{L}_2.S &\equiv \nu y :: \mathcal{L}_2.\nu x :: \mathcal{L}_1.S \ \text{ when } x \neq y,\ x \notin \mathcal{L}_2, \text{ and } y \notin \mathcal{L}_1 \\
m[\text{newamb}(m', P).Q] &\equiv \nu m' :: \emptyset.(m'[P]||m[Q]) \ \text{ when } m \neq m' \\
\nu x :: \mathcal{L}.\text{nilsystem} &\equiv \text{nilsystem} \\
(P \equiv Q) &\Rightarrow (m[P] \equiv m[Q]) \\
!P &\equiv newamb(p, P).\text{nil}||!P
\end{aligned}$$

We need one more piece of structures to be able to give our labeled transition semantics: the labels themselves. This time, ρ will take values in the following set of labels:

$$\begin{aligned}\rho \in & \{(m', \text{in } m) \mid m \text{ and } m' \text{ ambient names}\} \\ & \cup \{(m', \overline{\text{in}}\ m) \mid m \text{ and } m' \text{ ambient names or } m = \_\} \\ & \cup \{\langle n \rangle^m \mid m \text{ an ambient name, } n \text{ a message}\} \\ & \cup \{(n)^m \mid m \text{ an ambient name, } n \text{ a message}\} \\ & \cup \{\nu n :: \mathcal{L}.\langle n \rangle^m \mid m \text{ an ambient name, } n \text{ a message, } \mathcal{L} \text{ a set of ambients}\} \\ & \cup \{\tau\} \\ & \cup \{\nu m :: \mathcal{L}.\tau \mid m \text{ an ambient name, } \mathcal{L} \text{ a set of ambients}\}.\end{aligned}$$

The actual semantics are quite complicated, and we divide them into sections for easier digestion. The following table contains the rules for parallel composition, the basics of manipulating scoped variables, and dealing with structural equivalence:

$$\text{Quotient:}\frac{G_1 \triangleleft S_1 \equiv G_1' \triangleleft S_1' \qquad G_1' \triangleleft S_1' \xrightarrow{\rho} G_2' \triangleleft S_2' \qquad G_2' \triangleleft S_2' \equiv G_2 \triangleleft S_2}{G_1 \triangleleft S_1 \xrightarrow{\rho} G_2 \triangleleft S_2},$$

$$\text{Scoping:}\frac{G\#x :: \mathcal{L} \triangleleft S \xrightarrow{\rho} G\#x :: \mathcal{L}' \triangleleft S' \qquad x \text{ does not appear free in } \rho}{G \triangleleft \nu x :: \mathcal{L}.S \xrightarrow{\rho} G \triangleleft \nu x :: \mathcal{L}'.S'},$$

$$\text{ParallelExecution:}\frac{G \triangleleft S_1 \xrightarrow{\rho} G \triangleleft S_1'}{G \triangleleft S_1||S_2 \xrightarrow{\rho} G \triangleleft S_1'||S_2}.$$

Our next task is to describe how communication works in this calculus. The basic setup is to let a process send via putting a message into the transition label, provided the network topology allows such an action, and then let other ambients hear the message by pairing the send and receive labels with the rules titled "Communicate." This is all made quite complicated with the introduction of scoped ambients; what we should do when a message contains or is transmitted over a scoped variable is not immediately clear. The solution that we adopt here stems from the ideology that scoped variables should be thought of as "secrets," and that broadcasting a secret publicly means losing any secrecy you may have previously had.

$$\text{PubSend:}\frac{G = G_1\#(m :: \mathcal{L})\#G_2 \qquad m' \in \mathcal{L} \cup \{m\}}{G \triangleleft m[\langle n \rangle^{m'}.P] \xrightarrow{\langle n \rangle^{m'}} G \triangleleft m[P]},$$

$$\text{Receive:}\frac{G = G_1\#(m :: \mathcal{L})\#G_2 \qquad m' \in \mathcal{L} \cup \{m'\}}{G \triangleleft m[(x)^{m'}.P] \xrightarrow{(n)^{m'}} G \triangleleft m[P[n/x]]},$$

$$\text{PubCommunicate:}\frac{G \triangleleft S_1 \xrightarrow{\langle n \rangle^{m'}} G \triangleleft S_1' \qquad G \triangleleft S_2 \xrightarrow{(n)^{m'}} G \triangleleft S_2'}{G \triangleleft S_1||S_2 \xrightarrow{\langle n \rangle^{m'}} G \triangleleft S_1'||S_2'}.$$

These three rules capture the easy part of broadcast communication. As a result of the syntax for scoped ambients, when a message is being transmitted on a scoped ambient that is about to lose scope, there are no other ambients farther up in the proof tree that can participate in this communication action. It is therefore safe to throw away the message label and replace it with a τ label, thereby allowing private communication to take place – without this deletion rule, private communication would not be allowed, since the label for propagating a private communication contains the name of the scoped ambient.

$$\text{PubClose:}\frac{G\#x :: \mathcal{L} \triangleleft S \xrightarrow{\langle n \rangle^x} G\#x :: \mathcal{L} \triangleleft S'}{G \triangleleft \nu x :: \mathcal{L}.S \xrightarrow{\tau} G \triangleleft \nu x :: \mathcal{L}.S'}.$$

The transmission of a message that contains a scoped name also requires special attention, since we cannot lift the public send transition rule's label through the scoping rule. Thinking of scoped names as secrets, transmitting a secret corresponds roughly to letting the cat out of the bag. To account for this, when we transmit a secret message, we have to "let go" of the scope of the message and make it (temporarily) public information. If the message is transmitted over a public ambient, we have no control over who can hear the message – on the other hand, if the message is transmitted over a private ambient, when that ambient passes out of scope we know that no other ambients can participate in the communication, and the original secret therefore remains a shared secret among those ambients that we recorded as participating.

$$\text{PrivSend:}\frac{G = G_1\#(m :: \mathcal{L}')\#G_2 \quad m' \in \mathcal{L}' \cup \{m\} \quad x \text{ is fresh in } G}{G \triangleleft \nu x :: \mathcal{L}.m[\langle x \rangle^{m'}.S] \xrightarrow{\nu x :: \mathcal{L}.\langle x \rangle^m} G\#(x :: \mathcal{L}) \triangleleft S},$$

$$\text{PrivCommunicate:}\frac{G \triangleleft S \xrightarrow{\nu n :: \mathcal{L}.\langle n \rangle^{m'}} G \triangleleft S' \quad G \triangleleft m[P] \xrightarrow{(x)^{m'}} G \triangleleft m[P'] \quad x \text{ free in } G}{G \triangleleft S||m[P] \xrightarrow{\nu n :: \mathcal{L}.\langle n \rangle^{m'}} G \triangleleft S'||m[P'][n/x]},$$

$$\text{PrivClose:}\frac{G\#x :: \mathcal{L} \triangleleft S \xrightarrow{\nu n :: \mathcal{L}'.\langle x \rangle^n} G'\#x :: \mathcal{L} \triangleleft S' \quad G' \equiv G\#(n :: \mathcal{L}')}{G \triangleleft \nu x :: \mathcal{L}.S \xrightarrow{\tau} G \triangleleft \nu x :: \mathcal{L}.\nu n :: \mathcal{L}.S'}.$$

The remaining section of semantics to describe are those governing ambient mobility. As with communication, we give semantics for the public case first, then deal with the more complex private case afterward. The basic idea is that when an ambient m wants to enter an ambient n, they perform a handshake before the graph topology is allowed to change – in particular, n can allow either the specific entry of m or the entry of any ambient at all, symbolized by the "name" $\_$. For m to later exit n requires no handshake; this is motivated by the operation of physical networks, where members of a network are not required to announce their departure before exiting.

$$\text{Enter:}\frac{G = G_1\#m' :: \mathcal{L}'\#G_2\#m :: \mathcal{L}\#G_3 \quad G' = G_1\#m' :: \mathcal{L}'\#G_2\#m :: \mathcal{L} \cup \{m'\}\#G_3}{G \triangleleft m[\text{in } m'.P] \xrightarrow{(m, \text{in } m')} G' \triangleleft m[P]},$$

$$\text{LetEnter:}\frac{G = G_1\#m :: \mathcal{L}\#G_2 \quad m' \notin \mathcal{L} \quad m' \neq m \quad \phi \in \{\_, m\} \quad G' = G_1\#m :: \mathcal{L} \cup \{m'\}\#G_2}{G \triangleleft m[\overline{\text{in}}\ m'.P] \xrightarrow{(m, \overline{\text{in}}\ m')} G' \triangleleft m[P]},$$

$$\text{DoEnter:}\frac{G \triangleleft m[P] \xrightarrow{(m, \text{in } m')} G' \triangleleft m[P'] \quad G \triangleleft m'[Q] \xrightarrow{(m', \overline{\text{in}}\ \phi)} G' \triangleleft m'[Q']}{G \triangleleft (m[P]||m'[Q]) \xrightarrow{\tau} G' \triangleleft (m[P']||m'[Q'])},$$

$$\text{Exit:}\frac{G = G_1\#m' :: \mathcal{L}\#G_2 \quad m \in \mathcal{L} \quad G' = G_1\#m' :: \mathcal{L} \setminus \{m\}\#G_2}{G \triangleleft m[\text{out } m'.P] \xrightarrow{\tau} G' \triangleleft m[P]}.$$

This naïve presentation of ambient motion semantics actually does allow scoped ambients to have a limited range of motion: private ambients are permitted to move into public ambients, but not vice versa. If a public ambient knows a private ambient's name, tries to enter, and the private ambient offers to handshake with $\overline{\text{in}}\ \_$, we do not want the behavior that the private ambient can somehow detect that the public ambient is public, and therefore not on the approved guest list of people that can make use of their knowledge of the secret ambient's name, but (so far) this is the behavior that these semantics exhibit.

To make sense of this, we can use the same idea of throwing the ambient's topological information into the transition label and then wrapping up scope where (and if) appropriate.

$$\text{ThrowTau:}\frac{G\#m :: \mathcal{L} \triangleleft S \xrightarrow{\tau} G' \triangleleft S' \qquad G' = G_1\#m :: \mathcal{L}'\#G_2}{G \triangleleft \nu m :: \mathcal{L}.S \xrightarrow{\nu m::\mathcal{L}'.\tau} G' \triangleleft S'},$$

$$\text{ReinsertTau:}\frac{G \triangleleft S \xrightarrow{\nu m::\mathcal{L}.\tau} G'\#m :: \mathcal{L} \triangleleft S'}{G \triangleleft S \xrightarrow{\tau} G' \triangleleft \nu m :: \mathcal{L}.S'}.$$

These rules allow us to manipulate the motion of scoped ambients exactly the way we want. However, because we cannot tell whether an in / $\overline{\text{in}}$ pairing generated the τ action or one of our other rules (Exit, PubClose, or PrivClose), we must demonstrate that these last rules do not introduce any new behavior in those settings.

Theorem 1. (Coherence) *When the Scoping rule applies to a τ-transition, the result is coherent with applying the ThrowTau and ReinsertTau transition rules.*

Proof. By hypothesis, we have a proof tree of the form $\dfrac{G\#m :: \mathcal{L} \triangleleft S \xrightarrow{\tau} G\#m :: \mathcal{L}' \triangleleft S'}{G \triangleleft \nu m :: \mathcal{L}.S \xrightarrow{\tau} G \triangleleft \nu m :: \mathcal{L}'.S'}$.

The equivalent proof tree with the τ-scoping rules is $\dfrac{\dfrac{G\#m :: \mathcal{L} \triangleleft S \xrightarrow{\tau} G'\#m :: \mathcal{L}' \triangleleft S'}{G \triangleleft \nu m :: \mathcal{L}.S \xrightarrow{\nu m::\mathcal{L}'.\tau} G'\#m :: \mathcal{L}' \triangleleft S'}}{G \triangleleft \nu m :: \mathcal{L}.S \xrightarrow{\tau} G' \triangleleft \nu m :: \mathcal{L}'.S'}$. □

3. An embedding

We now adapt the embedding of the π-calculus into the Secure Broadcast Ambient calculus found in [2] to our broadcast ambient calculus; because of the relatively simple constructs in the π-calculus, very little work has to be done to accomplish this.

The basic setup for the translation is to construct an ambient for every communication channel, equipped with a process that allows two external processes to communicate at a time, an ambient to house every π-calculus process, and then translate the π-calculus send and receives into a complicated handshake protocol that guarantees either delivery or program failure.

To start, select a π-calculus process P. For every channel name M in P, construct an ambient $CA(M)$ given as follows:

$$CA(M) = !(\overline{\text{in}}\ \_.\text{nil})||!(\text{newamb}(m, \overline{\text{in}}\ \_.\text{nil}||\overline{\text{in}}\ \_.\text{nil}).\langle m\rangle^M.\text{nil})$$

The translation of π-calculus primitives to broadcast ambient primitives is given in the following table:

$$\begin{aligned}
J(\overline{M}\langle n\rangle.P) &= \text{in } M.(y)^M.\text{in } y.(z)^y.\langle n\rangle^y.(z)^y.\text{out } y.\text{out } M.J(P)\\
J(M(x).P) &= \text{in } M.(y)^M.\text{in } y.\langle y\rangle^y.(x)^y.\langle y\rangle^y.\text{out } y.\text{out } M.J(P)\\
J(P||Q) &= \text{newamb}(p, J(P)).\text{nil}||\text{newamb}(q, J(Q)).\text{nil}\\
J(!P) &= !J(P)\\
J(\mathbf{0}) &= \text{nil}\\
J(\nu x.P) &= \text{newamb}(x, CA(x)).J(P)
\end{aligned}$$

With these two functions available, we are equipped to translate P itself, given by

$$I(P) = \emptyset \triangleleft \nu top :: \emptyset.top[J(P)]|||\,(||_{M\in P} M[CA(M)])$$

4. Equivalences

We now wish to show that this map from π-calculus processes to broadcast ambient configurations is in some sense "faithful," for which we take the obvious meaning that I ought to preserve in both directions structural equivalence and a notion of equivalence attached to the labels of our labeled transition semantics called "bisimilarity." We give definitions and proofs for both of these below.

4.1. Structural embeddings. Perhaps the most obvious definition of what it means for an embedding to be faithful is to ask that it, while defined over the larger grammar, respects the quotient we've placed upon it by structural equivalence. That an embedding satisfy this is a necessary step in exploring any sort of more serious equivalence, such as bisimilarity, because our labeled transitions specifically allow labels to occur uniformly across process representations that lie inside the same structural equivalence class. Without first demonstrating that our embedding respects this partitioning, we cannot proceed.

Theorem 2. *Let P and Q be π-calculus processes. If $I(P) \equiv I(Q)$, then $P \equiv Q$.*

Proof. Since structural equivalence in the π-calculus is defined as the congruence closure of a particular ruleset, it is sufficient to demonstrate that if P and Q are related by one of those rules then so are $I(P)$ and $I(Q)$. The larger result then follows by structural induction. To this end, fix a π-calculus process P and suppose that Q falls into one of the following cases:

- Suppose that $P \equiv Q$ by α-equivalence. Since the embedding I takes binding constructs to binding constructs, $I(P) \equiv I(Q)$ by α-equivalence as well.
- Suppose now that $Q = !P$. Since $J(!P) = !J(P)$, we immediately have $I(P) \equiv I(Q)$.
- Parallel composition in the π-calculus and in the broadcast ambient calculus both form a commutative monoid with identities $\mathbf{0}$ and nil respectively, and I is a homomorphism of monoids. Thus, if P and Q are equivalent via one of the rules encoding that parallel composition in the π-calculus, then $I(P)$ and $I(Q)$ will be structurally equivalent via those same rules in the broadcast ambient context.
- Suppose that $P = (\nu x.P_1)||P_2$, that $Q = \nu x.(P_1||P_2)$, and that x is not free in P_2. After expanding out the defintitions of I, J, CA, and newamb$(-,-)$, these expressions are equivalent to

  $$\emptyset \triangleleft \nu top :: \emptyset.\nu x :: \emptyset.\nu p :: \emptyset.\nu q :: \emptyset.top[\text{nilsystem}]||x[CA(x)]||p[J(P_1)]||q[J(P_2)]$$

  via the rule that two ν-bound variables can have their binding sites interchanged when neither parent list mentions the other.
- Suppose that $P = \nu x.\nu y.P'$ and $Q = \nu y.\nu x.P'$. Because the ambients generated by newamb(, h)ave no parents, newamb$((,p).$newamb$((,q).P').$nil$).$nil and newamb$((,q).$newamb$((,p).P').$nil$).$nil are structurally equivalence in the broadcast ambient calculus, and $I(P) = I(Q)$.
- Suppose that $P = \nu x.\mathbf{0}$ and $Q = \mathbf{0}$. Because $\nu x :: \mathcal{L}.\text{nilsystem} \equiv \text{nilsystem}$, we have that $I(P) \equiv I(Q)$. □

Theorem 3. *Again, let P and Q be π-calculus processes. If $P \equiv Q$, then $I(P) \equiv I(Q)$.*

Proof. □

4.2. Bisimilarity. Given a labeled transition semantics and two processes P and Q, P is said to simulate Q if whenever P can transition to P' with label ρ, then there exists some process Q' and a ρ-transition Q to Q' such that P' and Q' have this same property. The processes are said to be *bisimilar* or *related by bisimulation* when P simulates Q and vice versa.

Theorem 4. *Let P and Q be π-calculus processes such that $I(P)$ and $I(Q)$ are bisimilar broadcast ambient processes. Then P and Q themselves are bisimilar.*

Proof. The proof proceeds by structural induction on the length of sequences of π-calculus transition labels. In the base case, when P and Q are unable to produce transition labels, they are seen to be structurally congruent to the empty process. Therefore, $I(P)$ and $I(Q)$ are structurally congruent to the empty configuration, and thus unable to produce any labels of their own, making them trivially bisimilar.

Now suppose we have a pair of processes which are known to be bisimilar on transition label sequences of length n. Due to the symmetry of the above argument, it is sufficient to show that if P transitions to P' by a label ρ, then there is a process Q' and a transition $Q \xrightarrow{\rho} Q'$; we now make a case analysis for ρ.

- Suppose $\rho = \overline{m}\langle n \rangle$ is a sending transition. Then $I(P)$ is capable of producing the following sequence of labels:

  $$\tau, \nu y :: \emptyset.\langle y \rangle^M, \tau, (z)^y, \langle n \rangle^y, (z)^y, \tau, \tau.$$

  By assumption, $I(Q)$ is also capable of emitting this sequence of labels, and Q is then immediately forced to contain the action $\overline{m}\langle n \rangle$ due to the fifth label in the above sequence, and so Q also admits a $\overline{m}\langle n \rangle$-labeled transition.

- Suppose $\rho = m(n)$ is a receiving transition. Then $I(P)$ is capable of producing the following sequence of labels:
$$\tau, \nu y :: \emptyset.\langle y \rangle^M, \tau, \langle y \rangle^y, (n)^y, \langle y \rangle^y, \tau, \tau.$$
By assumption, $I(Q)$ is also capable of emitting this sequence of labels, and Q is then immediately forced to contain the action $m(x)$ due to the fifth label in the above sequence. Q then also admits a $m(n)$-labeled transition.
- Suppose ρ is a τ-transition. In the π-calculus, a τ-transition is possible precisely when the Communicate inference rule applies, which is in turn predicated on the ability to produce matching send and receive transitions. By the above two points, Q is also able to produce these send and receive labels, and therefore able to undergo a τ-transition as well.

□

Theorem 5. *Again, let P and Q be bisimilar π-calculus processes. Then $I(P)$ and $I(Q)$ are bisimilar broadcast ambient processes.*

Proof. □

References

[1] Martín Abadi and Andrew Gordon. *A Calculus for Cryptographic Protocols: the Spi Calculus*. Fourth ACM Conference on Computer and Communications Security, 1997.

[2] Elsa Gunter and Ayesha Yasmeen. *Embedding the Pi Calculus in Secure Broadcast Ambients*. Preprint, 2008.

[3] Elsa Gunter and Ayesha Yasmeen. *Secure Broadcast Ambients*. Preprint, 2008.

[4] Robin Milner, Joachim Parrow, and David Walker. *Modal Logics for Mobile Processes*. Theoretical Computer Science, v. 114, 1993.