Equational Properties of Mobile Ambients

Andrew D. Gordon[1] and Luca Cardelli[2]

[1] Microsoft Research, http://research.microsoft.com/~adg
[2] Microsoft Research, http://www.luca.demon.co.uk

Abstract. The ambient calculus is a process calculus for describing mobile computation. We develop a theory of Morris-style contextual equivalence for proving properties of mobile ambients. We prove a context lemma that allows derivation of contextual equivalences by considering contexts of a particular limited form, rather than all arbitrary contexts. We give an activity lemma that characterizes the possible interactions between a process and a context. We prove several examples of contextual equivalence. The proofs depend on characterizing reductions in the ambient calculus in terms of a labelled transition system.

1 Motivation

This paper develops tools for proving equations in the ambient calculus.

In earlier work [6], we introduced the ambient calculus by adding *ambients*—mobile, hierarchical protection domains—to a framework for concurrency extracted from the π-calculus [12]. The ambient calculus is an abstract model of mobile computation, including both mobile software agents and mobile hardware devices. The calculus models access control as well as mobility. For example, a process may move into or out of a particular ambient only if it possesses the appropriate capability.

This paper focuses on behavioural equivalence of mobile ambients. In particular, we study a form of Morris' contextual equivalence [14] for ambients and develop some proof techniques. Our motivation is to prove a variety of equations. Some of these equations express and confirm some of the informal principles we had in mind when designing the calculus. As in other recent work [1, 2], some of the equations establish security properties of systems modelled within the calculus.

The inclusion of primitives for mobility makes the theory of the ambient calculus more complex than that of its ancestor, the π-calculus. The main contribution of this paper is to demonstrate that some standard tools—a labelled transition system, a context lemma, and an activity lemma—may be recast in the setting of the ambient calculus. Moreover, the paper introduces a new technique—based on what we call the hardening relation—for factoring the definition of the labelled transition system into a set of rules that identify the individual processes participating in a transition, and a set of rules that express how the participant processes interact.

We begin, in Section 2, by reviewing the syntax and reduction semantics of the ambient calculus. The semantics consists of a structural congruence relation $P \equiv Q$ (which says that P may be structurally rearranged to yield Q) and a reduction relation $P \rightarrow Q$ (which says that P may evolve in one step of computation to yield Q).

We introduce contextual equivalence $P \simeq Q$ in Section 3. We define a predicate, $P \Downarrow n$, which means intuitively that an observer may eventually detect an ambient named n at the top-level of the process P. Then we define $P \simeq Q$ to mean that, whenever P and Q are placed within an arbitrary context constructed from the syntax of the calculus, any observation made of P may also be made of Q, and vice versa. We give examples of pairs of processes that are equivalent and examples of pairs that are inequivalent.

In Section 4, we describe some techniques for proving contextual equivalence. We introduce a second operational semantics for the ambient calculus based on a hardening relation and a labelled transition system. The hardening relation identifies the subprocesses of a process that may participate in a computation step. We use the hardening relation both for defining the labelled transition system and for characterizing whether an ambient of a particular name is present at the top-level of a process. Our first result, Theorem 1, asserts that the τ-labelled transition relation and the reduction relation are the same, up to structural congruence. So our two operational semantics are equivalent. The labelled transition system is useful for analyzing the possible evolution of a process, since we may read off the possible labelled transitions of a process by inspecting its syntactic structure. Our second result, Theorem 2 is a context lemma that allows us to prove contextual equivalence by considering a limited set of contexts, known as harnesses, rather than all arbitrary contexts. A harness is a context with a single hole that is enclosed only within parallel compositions, restrictions, and ambients. The third result of this section, Theorem 3, is an activity lemma that elaborates the ways in which a reduction may be derived when a process is inserted into a harness: either the process reduces by itself, or the harness reduces by itself, or there is an interaction between the harness and the process.

We exercise these proof techniques on examples in Section 5, and conclude in Section 6.

2 The Ambient Calculus (Review)

We briefly describe the syntax and semantics of the calculus. We assume there are infinite sets of *names* and *variables*, ranged over by m, n, p, q, and x, y, z, respectively. The syntax of the ambient calculus is based on categories of *expressions* and *processes*, ranged over by M, N, and P, Q, R, respectively. The calculus inherits a core of concurrency primitives from the π-calculus: a restriction $(\nu n)P$ creates a fresh name n whose scope is P; a composition $P \mid Q$ behaves as P and Q running in parallel; a replication $!P$ behaves as unboundedly many replicas of P running in parallel; and the inactive process $\mathbf{0}$ does nothing. We augment these π-calculus processes with primitives for mobility—ambients,

$n[P]$, and the exercise of capabilities, $M.P$—and primitives for communication—input, $(x).P$, and output, $\langle M \rangle$.

Here is an example process that illustrates the new primitives for mobility and communication:

$$m[p[\mathit{out}\ m.\mathit{in}\ n.\langle M \rangle]] \mid n[\mathit{open}\ p.(x).Q]$$

The effect of the mobility primitives in this example is to move the ambient p out of m and into n, and then to open it up. The input $(x).Q$ may then consume the output $\langle M \rangle$ to leave the residue $m[] \mid n[Q\{x \leftarrow M\}]$. We may regard the ambients m and n in this example as modelling two machines on a network, and the ambient p as modelling a packet sent from m to n. Next, we describe the semantics of the new primitives in more detail.

An ambient $n[P]$ is a boundary, named n, around the process P. The boundary prevents direct interactions between P and any processes running in parallel with $n[P]$, but it does not prevent interactions within P. Ambients may be nested, so they induce a hierarchy. For example, in the process displayed above, the ambient named m is a parent of the ambient named p, and the ambients named m and n are siblings.

An action $M.P$ exercises the capabilities represented by M, and then behaves as P. The action either affects an enclosing ambient or one running in parallel. A capability is an expression derived from the name of an ambient. The three basic capabilities are *in* n, *out* n, and *open* n. An action *in* $n.P$ moves its enclosing ambient into a sibling ambient named n. An action *out* $n.P$ moves its enclosing ambient out of its parent ambient, named n, to become a sibling of the former parent. An action *open* $n.P$ dissolves the boundary of an ambient $n[Q]$ running in parallel; the outcome is that the residue P of the action and the residue Q of the opened ambient run in parallel. In general, the expression M in $M.P$ may stand for a finite sequence of the basic capabilities, which are exercised one by one. Finite sequences are built up using concatenation, written $M.M'$. The empty sequence is written ϵ.

The final two process primitives allow communication of expressions. Expressions include names, variables, and capabilities. An output $\langle M \rangle$ outputs the expression M. An input $(x).P$ blocks until it may consume an output running in parallel. Then it binds the expression being output to the variable x, and runs P. In $(x).P$, the variable x is bound; its scope is P. Inputs and outputs are local to the enclosing ambient. Inputs and outputs may not interact directly through an ambient boundary. Hence we may think of there being an implicit input/output channel associated with each ambient.

We formally specify the syntax of the calculus as follows:

Expressions and processes:

$M, N ::=$	expressions	$P, Q, R ::=$	processes
x	variable	$(\nu n)P$	restriction
n	name	$\mathbf{0}$	inactivity
in M	can enter M	$P \mid Q$	composition

$out\ M$	can exit M	$!P$	replication
$open\ M$	can open M	$M[P]$	ambient
ϵ	null	$M.P$	action
$M.M'$	path	$(x).P$	input
		$\langle M \rangle$	output

In situations where a process is expected, we often write just M as a shorthand for the process $M.\mathbf{0}$. We often write just $M[]$ as a shorthand for the process $M[\mathbf{0}]$. We write $(\nu\vec{p})P$ as a shorthand for $(\nu p_1)\cdots(\nu p_k)P$ where $\vec{p} = p_1, \ldots, p_k$.

We let $fn(M)$ and $fv(M)$ be the sets of *free names* and *free variables*, respectively, of an expression M. Similarly, $fn(P)$ and $fv(P)$ are the sets of *free names* and *free variables* of a process P. If a phrase ϕ is an expression or a process, we write $\phi\{x \leftarrow M\}$ and $\phi\{n \leftarrow M\}$ for the outcomes of capture-avoiding substitutions of the expression M for each free occurrence of the variable x and the name n, respectively, in ϕ. We identify processes up to consistent renaming of bound names and variables.

We formally define the operational semantics of ambient calculus in the chemical style, using structural congruence and reduction relations:

Structural Congruence: $P \equiv Q$

$P \mid Q \equiv Q \mid P$	$P \equiv P$
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	$Q \equiv P \Rightarrow P \equiv Q$
$!P \equiv P \mid !P$	$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	
$n \notin fn(P) \Rightarrow (\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$	$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$
$n \neq m \Rightarrow (\nu n)m[P] \equiv m[(\nu n)P]$	$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$
$P \mid \mathbf{0} \equiv P$	$P \equiv Q \Rightarrow !P \equiv !Q$
$(\nu n)\mathbf{0} \equiv \mathbf{0}$	$P \equiv Q \Rightarrow M[P] \equiv M[Q]$
$!\mathbf{0} \equiv \mathbf{0}$	$P \equiv Q \Rightarrow M.P \equiv M.Q$
$\epsilon.P \equiv P$	$P \equiv Q \Rightarrow (x).P \equiv (x).Q$
$(M.M').P \equiv M.M'.P$	

Reduction: $P \rightarrow Q$

$n[in\ m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$	$P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$
$m[n[out\ m.P \mid Q] \mid R] \rightarrow n[P \mid Q] \mid m[R]$	$P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$
$open\ n.P \mid n[Q] \rightarrow P \mid Q$	$P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$
$\langle M \rangle \mid (x).P \rightarrow P\{x \leftarrow M\}$	$P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q'$

For example, the process displayed earlier has the following reductions:

$$\begin{aligned} m[p[out\ m.in\ n.\langle M \rangle]] \mid n[open\ p.(x).P] &\rightarrow m[] \mid p[in\ n.\langle M \rangle] \mid n[open\ p.(x).P] \\ &\rightarrow m[] \mid n[p[\langle M \rangle] \mid open\ p.(x).P] \\ &\rightarrow m[] \mid n[\langle M \rangle \mid (x).P] \\ &\rightarrow m[] \mid n[P\{x \leftarrow M\}] \end{aligned}$$

The syntax allows the formation of certain processes that may not participate in any reductions, such as the action $n.P$ and the ambient $(in\, n)[P]$. The presence of these nonsensical processes is harmless as far as the purposes of this paper are concerned. They may be ruled out by a simple type system [7].

This concludes our brief review of the calculus. An earlier paper [6] explains in detail the motivation for our calculus, and gives several programming examples.

3 Contextual Equivalence

Morris-style contextual equivalence [14] (otherwise known as may-testing equivalence [8]) is a standard way of saying that two processes have the same behaviour: two processes are contextually equivalent if and only if they admit the same elementary observations whenever they are inserted inside any arbitrary enclosing process. In the setting of the ambient calculus, we shall define contextual equivalence in terms of observing the presence, at the top-level of a process, of an ambient whose name is not restricted.

Let us say that a process P *exhibits a name* n just if P is a process with a top-level ambient named n, that is not restricted:

Exhibition of a Name: $P \downarrow n$

$$P \downarrow n \quad \triangleq \quad \text{there are } \vec{m},\ P',\ P'' \text{ with } n \notin \{\vec{m}\} \text{ and } P \equiv (\nu \vec{m})(n[P'] \mid P'')$$

Let us say that a process P *converges to a name* n just if after some number of reductions, P exhibits n:

Convergence to a Name: $P \Downarrow n$

$$\text{(Conv Exh)}\ \frac{P \downarrow n}{P \Downarrow n} \qquad \text{(Conv Red)}\ \frac{P \to Q \qquad Q \Downarrow n}{P \Downarrow n}$$

Next, let a *context*, $\mathcal{C}()$, be a process containing zero or more holes. We write a hole as $()$. We write $\mathcal{C}(P)$ for the outcome of filling each of the holes in the context $\mathcal{C}$ with the process P. Variables and names free in P may become bound in $\mathcal{C}(P)$. For example, if $P = n[\langle x \rangle]$ and $\mathcal{C}() = (\nu n)(x).()$, the variable x and the name n have become bound in $\mathcal{C}(P) = (\nu n)(x).n[\langle x \rangle]$. Hence, we do not identify contexts up to renaming of bound variables and names.

Now, we can formally define contextual equivalence of processes:

Contextual Equivalence: $P \simeq Q$

$$P \simeq Q \quad \triangleq \quad \text{for all contexts } \mathcal{C}() \text{ and names } n,\ \mathcal{C}(P) \Downarrow n \Leftrightarrow \mathcal{C}(Q) \Downarrow n$$

The following two propositions state some basic properties enjoyed by contextual equivalence. Let a relation $\mathcal{R}$ be a *precongruence* if and only if, for all

P, Q, and $\mathcal{C}()$, if $P \mathrel{\mathcal{R}} Q$ then $\mathcal{C}(P) \mathrel{\mathcal{R}} \mathcal{C}(Q)$. If, in addition, $\mathcal{R}$ is reflexive, symmetric, and transitive, we say it is a *congruence*. For example, the structural congruence relation has these properties. Moreover, by a standard argument, so has contextual equivalence:

Proposition 1. *Contextual equivalence is a congruence.*

Structural congruence preserves exhibition of or convergence to a name, and hence is included in contextual equivalence:

Lemma 1. *Suppose $P \equiv Q$. If $P \downarrow n$ then $Q \downarrow n$. Moreover, if $P \Downarrow n$ then $Q \Downarrow n$ with the same depth of inference.*

Proposition 2. *If $P \equiv Q$ then $P \simeq Q$.*

The following two examples illustrate that to show that two processes are contextually inequivalent, it suffices to find a context that distinguishes them.

Example 1. If $m \neq n$ then $m[] \not\simeq n[]$.

Proof. Consider the context $\mathcal{C}() = ()$. Since $\mathcal{C}(m[]) \equiv m[]$, we have $\mathcal{C}(m[]) \downarrow m$. By (Conv Exh), $\mathcal{C}(m[]) \Downarrow m$. On the other hand, the process $n[]$ has no reductions, and does not exhibit m. Hence, we cannot derive $\mathcal{C}(n[]) \Downarrow m$. □

Example 2. If $m \neq n$ then $\mathit{open}\, m.\mathbf{0} \not\simeq \mathit{open}\, n.\mathbf{0}$.

Proof. Let $\mathcal{C}() = m[p[]] \mid ()$. Then $\mathcal{C}(\mathit{open}\, m.\mathbf{0}) \Downarrow p$ but not $\mathcal{C}(\mathit{open}\, n.\mathbf{0}) \Downarrow p$. □

On the other hand, it is harder to show that two processes are contextually equivalent, since one must consider their behaviour when placed in an arbitrary context. For example, consider the following contextual equivalence:

Example 3. $(\nu n)(n[] \mid \mathit{open}\, n.P) \simeq P$ if $n \notin \mathit{fn}(P)$.

The restriction of the name n in the process $(\nu n)(n[] \mid \mathit{open}\, n.P)$ implies that no context may interact with this process until it has reduced to P. Therefore, we would expect the equation to hold. But to prove this and other equations formally we need some further techniques, which we develop in the next section. We return to Example 3 in Section 5.

4 Tools for Proving Contextual Equivalence

The tools we introduce are relations and theorems that help prove contextual equivalence.

4.1 A Hardening Relation

In this section, we define a relation that explicitly identifies the top-level subprocesses of a process that may be involved in a reduction. This relation, the *hardening* relation, takes the form,

$$P > (\nu p_1, \ldots, p_k)\langle P' \rangle P''$$

where the phrase $(\nu p_1, \ldots, p_k)\langle P' \rangle P''$ is called a *concretion*. We say that P' is the *prime* of the concretion, and that P'' is the *residue* of concretion. Both P' and P'' lie in the scope of the restricted names $p_1, \ldots, p_k$. The intuition is that the process P, which may have many top-level subprocesses, may harden to a concretion that singles out a prime subprocess P', leaving behind the residue P''. By saying that P' has a top-level occurrence in P, we mean that P' is a subprocess of P not enclosed within any ambient boundaries. In the next section, we use the hardening relation to define an operational semantics for the ambient calculus in terms of interactions between top-level occurrences of processes.

Concretions were introduced by Milner in the context of the π-calculus [10]. For the ambient calculus, we specify them as follows, where the prime of the concretion must be an action, an ambient, an input, or an output:

Concretions:

$C, D ::=$	concretions
$(\nu\vec{p})\langle M.P \rangle Q$	action, $M \in \{in\ n, out\ n, open\ n\}$
$(\nu\vec{p})\langle n[P] \rangle Q$	ambient
$(\nu\vec{p})\langle (x).P \rangle Q$	input
$(\nu\vec{p})\langle \langle M \rangle \rangle Q$	output

The order of the bound names $p_1, \ldots, p_k$ in a concretion $(\nu p_1, \ldots, p_k)\langle P' \rangle P''$ does not matter and they may be renamed consistently. When $k = 0$, we may write the concretion as $(\nu)\langle P' \rangle P''$.

We now introduce the basic ideas of the hardening relation informally. If P is an action $in\ n.Q$, $out\ n.Q$, $open\ n.Q$, an ambient $n[Q]$, an input $(x).Q$, or an output $\langle M \rangle$, then P hardens to $(\nu)\langle P \rangle \mathbf{0}$. Consider two processes P and Q. If either of these hardens to a concretion, then their composition $P \mid Q$ may harden to the same concretion, but with the other process included in the residue of the concretion. For example, if $P > (\nu)\langle P_1 \rangle P_2$ then $P \mid Q > (\nu)\langle P_1 \rangle (P_2 \mid Q)$. If a process P hardens to a concretion, then the replication $!P$ may harden to the same concretion, but with $!P$ included in the residue of the concretion—a replication is not consumed by hardening. Finally, if a process P hardens to a concretion C, then the restriction $(\nu n)P$ hardens to a concretion written $\overline{(\nu n)}C$, which is the same as C but with the restriction (νn) included either in the list of bound names, the prime, or the residue of C. We define $\overline{(\nu n)}C$ by:

Restricting a concretion: $\overline{(\nu n)}C$ where $C = (\nu\vec{p})\langle P_1 \rangle P_2$ and $n \notin \{\vec{p}\}$

(1) If $n \in fn(P_1)$ then:

(a) If $P_1 = m[P_1']$, $m \neq n$, and $n \notin fn(P_2)$, let $\overline{(\nu n)}C \triangleq (\nu \vec{p})\langle m[(\nu n)P_1']\rangle P_2$.
(b) Otherwise, let $\overline{(\nu n)}C \triangleq (\nu n, \vec{p})\langle P_1\rangle P_2$.

(2) If $n \notin fn(P_1)$ let $\overline{(\nu n)}C \triangleq (\nu \vec{p})\langle P_1\rangle(\nu n)P_2$.

Next, we define the hardening relation by the following:

Hardening: $P > C$

(Harden Action) $\dfrac{M \in \{in\ n, out\ n, open\ n\}}{M.P > (\nu)\langle M.P\rangle \mathbf{0}}$ (Harden ϵ) $\dfrac{P > C}{\epsilon.P > C}$ (Harden .) $\dfrac{M.(N.P) > C}{(M.N).P > C}$

(Harden Amb) $\dfrac{}{n[P] > (\nu)\langle n[P]\rangle \mathbf{0}}$ (Harden Input) $\dfrac{}{(x).P > (\nu)\langle (x).P\rangle \mathbf{0}}$ (Harden Output) $\dfrac{}{\langle M\rangle > (\nu)\langle\langle M\rangle\rangle \mathbf{0}}$

(Harden Par 1) (for $\{\vec{p}\} \cap fn(Q) = \varnothing$) $\dfrac{P > (\nu \vec{p})\langle P'\rangle P''}{P \mid Q > (\nu \vec{p})\langle P'\rangle(P'' \mid Q)}$

(Harden Par 2) (for $\{\vec{q}\} \cap fn(P) = \varnothing$) $\dfrac{Q > (\nu \vec{q})\langle Q'\rangle Q''}{P \mid Q > (\nu \vec{q})\langle Q'\rangle(P \mid Q'')}$

(Harden Repl) $\dfrac{P > (\nu \vec{p})\langle P'\rangle P''}{!P > (\nu \vec{p})\langle P'\rangle(P'' \mid !P)}$ (Harden Res) $\dfrac{P > C}{(\nu n)P > \overline{(\nu n)}C}$

For example, the process $P = (\nu p)(\nu q)(n[p[]] \mid q[])$ may harden in two ways:

$$P > (\nu)\langle n[(\nu p)p[]]\rangle(\nu q)(\mathbf{0} \mid q[])$$
$$P > (\nu q)\langle q[]\rangle(\nu p)(n[p[]] \mid \mathbf{0})$$

The next two results relate hardening and structural congruence.

Lemma 2. *If $P > (\nu \vec{p})\langle P'\rangle P''$ then $P \equiv (\nu \vec{p})(P' \mid P'')$.*

Proposition 3. *If $P \equiv Q$ and $Q > (\nu \vec{r})\langle Q'\rangle Q''$ and then there are P' and P'' with $P > (\nu \vec{r})\langle P'\rangle P''$, $P' \equiv Q'$, and $P'' \equiv Q''$.*

These results follow from inductions on the derivations of $P > (\nu \vec{p})\langle P'\rangle P''$ and $P \equiv Q$, respectively. Using them, we may characterize exhibition of a name independently of structural congruence:

Proposition 4. *$P \downarrow n$ if and only if $P > (\nu \vec{p})\langle n[P']\rangle P''$ and $n \notin \{\vec{p}\}$.*

Now, we can show that the hardening relation is image-finite:

Lemma 3. *For all P, $\{C : P > C\}$ is finite.*

The proof of this lemma is by induction on the structure of P, and suggests a procedure for the enumerating the set $\{C : P > C\}$. Given Proposition 4, it follows that the predicate $P \downarrow n$ is decidable.

4.2 A Labelled Transition System

The labelled transition system presented in this section allows for an analysis of the possible reductions from a process P in terms of the syntactic structure of P. The definition of the reduction relation does not directly support such an analysis, because of the rule $P' \equiv P, P \to Q, Q \equiv Q' \Rightarrow P' \to Q'$, which allows for arbitrary structural rearrangements of a process during the derivation of a reduction.

We define a family of transition relations $P \xrightarrow{\alpha} Q$, indexed by a set of labels, ranged over by $\alpha ::= \tau \mid \mathit{in}\ n \mid \mathit{out}\ n \mid \mathit{open}\ n$. An M-transition $P \xrightarrow{M} Q$ means that the process P has a top-level process exercising the capability M; these transitions are defined by the rule (Trans Cap) below. A τ-transition $P \xrightarrow{\tau} Q$ means that P evolves in one step to Q; these transitions are defined by the other rules below.

Labelled transitions: $P \xrightarrow{\alpha} P'$ **where** $\alpha ::= \tau \mid \mathit{in}\ n \mid \mathit{out}\ n \mid \mathit{open}\ n$

(Trans Amb)
$$\frac{P > (\nu\vec{p})\langle n[Q]\rangle P' \quad Q \xrightarrow{\tau} Q'}{P \xrightarrow{\tau} (\nu\vec{p})(n[Q'] \mid P')}$$

(Trans Cap)
$$\frac{P > (\nu\vec{p})\langle M.P'\rangle P'' \quad \mathit{fn}(M) \cap \{\vec{p}\} = \varnothing}{P \xrightarrow{M} (\nu\vec{p})(P' \mid P'')}$$

(Trans In) (where $\{\vec{r}\} \cap \mathit{fn}(n[Q]) = \varnothing$ and $\{\vec{r}\} \cap \{\vec{p}\} = \varnothing$)
$$\frac{P > (\nu\vec{p})\langle n[Q]\rangle R \quad Q \xrightarrow{\mathit{in}\ m} Q' \quad R > (\nu\vec{r})\langle m[R']\rangle R''}{P \xrightarrow{\tau} (\nu\vec{p},\vec{r})(m[n[Q'] \mid R'] \mid R'')}$$

(Trans Out) (where $n \notin \{\vec{q}\}$)
$$\frac{P > (\nu\vec{p})\langle n[Q]\rangle P' \quad Q > (\nu\vec{q})\langle m[R]\rangle Q' \quad R \xrightarrow{\mathit{out}\ n} R'}{P \xrightarrow{\tau} (\nu\vec{p})((\nu\vec{q})(m[R'] \mid n[Q']) \mid P')}$$

(Trans Open)
$$\frac{P > (\nu\vec{p})\langle n[Q]\rangle P' \quad P' \xrightarrow{\mathit{open}\ n} P''}{P \xrightarrow{\tau} (\nu\vec{p})(Q \mid P'')}$$

(Trans I/O) (where $\{\vec{q}\} \cap \mathit{fn}(\langle M\rangle) = \varnothing$)
$$\frac{P > (\nu\vec{p})\langle\langle M\rangle\rangle P' \quad P' > (\nu\vec{q})\langle(x).P''\rangle P'''}{P \xrightarrow{\tau} (\nu\vec{p})(P' \mid (\nu\vec{q})(P''\{x{\leftarrow}M\} \mid P'''))}$$

The rules (Trans In), (Trans Out), and (Trans Open) derive a τ-transition from an M-transition. We introduced the M-transitions to simplify the statement of these three rules. (Trans I/O) allows for exchange of messages. (Trans Amb) is a congruence rule for τ-transitions within ambients.

Given its definition in terms of the hardening relation, we may analyze the transitions derivable from any process by inspection of its syntactic structure. This allows a structural analysis of the possible reductions from a process, since the τ-transition relation corresponds to the reduction relation as in the following theorem, where $P \xrightarrow{\tau} \equiv Q$ means there is R with $P \xrightarrow{\tau} R$ and $R \equiv Q$.

Theorem 1. *$P \to Q$ if and only if $P \xrightarrow{\tau} \equiv Q$.*

As corollaries of Theorem 1 and Lemma 4, we get that the transition system is image-finite, and that the reduction relation is image-finite up to structural congruence:

Lemma 4. *For all P and α, the set $\{R : P \xrightarrow{\alpha} R\}$ is finite.*

Lemma 5. *For all P, the set $\{\{R : Q \equiv R\} : P \to Q\}$ is finite.*

4.3 A Context Lemma

The context lemma presented in this section is a tool for proving contextual equivalence by considering only a limited set of contexts, rather than all contexts. Many context lemmas have been proved for a wide range of calculi, starting with Milner's context lemma for the combinatory logic form of PCF [9].

Our context lemma is stated in terms of a notion of a *harness*:

Harnesses

$H ::=$	harnesses
$-$	process variable
$(\nu n)H$	restriction
$P \mid H$	left composition
$H \mid Q$	right composition
$n[H]$	ambient

Harnesses are analogous to the evaluation contexts found in context lemmas for some other calculi. Unlike the contexts of Section 3, harnesses are identified up to consistent renaming of bound names. We let $fn(H)$ and $fv(H)$ be the sets of names and variables, respectively, occurring free in a harness H. There is exactly one occurrence of the process variable $-$ in any harness. If H is an harness, we write $H\{P\}$ for the outcome of substituting the process P for the single occurrence of the process variable $-$. Names restricted in H are renamed to avoid capture of free names of P. For example, if $H = (\nu n)(- \mid \mathit{open}\ n)$ then $H\{n[]\} = (\nu n')(n[] \mid \mathit{open}\ n')$ for some $n' \neq n$.

Let a *substitution*, σ, be a list $x_1 \leftarrow M_1, \ldots, x_k \leftarrow M_k$, where the variables x_1, $\ldots$, x_k are pairwise distinct, and $fv(M_i) = \varnothing$ for each $i \in 1..k$. Let $dom(\sigma) = \{x_1, \ldots, x_k\}$. Let $P\sigma$ be the process $P\{x_1 \leftarrow M_1\} \cdots \{x_k \leftarrow M_k\}$. Let a process or a harness be *closed* if and only if it has no free variables (though it may have free names).

Here is our context lemma:

Theorem 2 (Context). *For all processes P and Q, $P \simeq Q$ if and only if for all substitutions σ with $dom(\sigma) = fv(P) \cup fv(Q)$, and for all closed harnesses H and names n, that $H\{P\sigma\} \Downarrow n \Leftrightarrow H\{Q\sigma\} \Downarrow n$.*

A corollary is that for all closed processes P and Q, $P \simeq Q$ if and only if for all closed harnesses H and names n, that $H\{P\} \Downarrow n \Leftrightarrow H\{Q\} \Downarrow n$.

In general, however, we need to consider the arbitrary closing substitution σ when using Theorem 2. This is because a variable free in a process may become bound to an expression once the process is placed in a context. For example, let $P = x[n[]] \mid \mathit{open}\ y.\mathbf{0}$ and $Q = \mathbf{0}$. Consider the context $\mathcal{C}() = \langle m, m\rangle \mid (x, y).()$.

We have $\mathcal{C}(P) \Downarrow n$ but not $\mathcal{C}(Q) \Downarrow n$. So P and Q are not contextually equivalent but they do satisfy $H\{P\} \Downarrow n \Leftrightarrow H\{Q\} \Downarrow n$ for all closed H and n.

Some process calculi enjoy stronger context lemmas. Let processes P and Q be *parallel testing equivalent* if and only if for all processes R and names n, that $P \mid R \Downarrow n \Leftrightarrow Q \mid R \Downarrow n$. We might like to show that any two closed processes are contextually equivalent if and only if they are parallel testing equivalent. This would be a stronger result than Theorem 2 because it would avoid considering contexts that include ambients. Such a result is true for CCS [8], for example, but it is false for the ambient calculus. To see this, let $P = \mathit{out}\, p.\mathbf{0}$ and $Q = \mathbf{0}$. We may show that $P \mid R \Downarrow n \Leftrightarrow Q \mid R \Downarrow n$ for all n and R. Now, consider the context $\mathcal{C}() = p[m[()]]$. We have $\mathcal{C}(P) \Downarrow m$ but not $\mathcal{C}(\mathbf{0}) \Downarrow m$. So P and Q are parallel testing equivalent but not contextually equivalent.

4.4 An Activity Lemma

When we come to apply Theorem 2 we need to analyze judgments of the form $H\{P\} \downarrow n$ or $H\{P\} \to Q$. In this section we formalize these analyses.

We begin by extending the structural congruence, hardening, and reduction relations to harnesses as follows:

- Let $H \equiv H'$ hold if and only if $H\{P\} \equiv H'\{P\}$ for all P.
- Let $H > (\nu\vec{p})\langle H'\rangle Q$ hold if and only if $H\{P\} > (\nu\vec{p})\langle H'\{P\}\rangle Q$ for all P such that $\{\vec{p}\} \cap \mathit{fn}(P) = \varnothing$.
- Let $H > (\nu\vec{p})\langle Q\rangle H'$ hold if and only if $H\{P\} > (\nu\vec{p})\langle Q\rangle(H'\{P\})$ for all P such that $\{\vec{p}\} \cap \mathit{fn}(P) = \varnothing$.
- Let $H \to H'$ hold if and only if, for all P, $H\{P\} \to H'\{P\}$.

We need the following lemma about hardening:

Lemma 6. *If $H\{P\} > C$ then either:*

(1) *$H > (\nu\vec{r})\langle H'\rangle R$ and $C = (\nu\vec{r})\langle H'\{P\}\rangle R$, or*
(2) *$H > (\nu\vec{r})\langle R\rangle H'$ and $C = (\nu\vec{r})\langle R\rangle(H'\{P\})$, or*
(3) *$H > (\nu\vec{r})\langle -\rangle R$, $P > (\nu\vec{p})\langle P'\rangle P''$, $C = (\nu\vec{r},\vec{p})\langle P'\rangle R'$ with $R' \equiv P'' \mid R$,*

where in each case $\{\vec{r}\} \cap \mathit{fn}(P) = \varnothing$.

Proposition 5. *If $H\{P\} \downarrow n$ then either (1) $H\{Q\} \downarrow n$ for all Q, or (2) $P \downarrow n$, and for all Q, $Q \downarrow n$ implies that $H\{Q\} \downarrow n$.*

Proof. By Proposition 4, $H\{P\} \downarrow n$ means there are $\vec{p}$, P', P'' such that $H\{P\} > (\nu\vec{p})\langle n[P']\rangle P''$ with $n \notin \{\vec{p}\}$. Hence, the proposition follows from Lemma 6. □

Intuitively, there are two ways in which $H\{P\} \downarrow n$ can arise: either the process P exhibits the name by itself, or the harness H exhibits the name n by itself. Proposition 5 formalizes this analysis. Similarly, there are three ways in which a reduction $H\{P\} \to Q$ may arise: either (1) the process P reduces by itself, or (2) the harness H reduces by itself, or (3) there is an interaction between the process and the harness. Theorem 3 formalizes this analysis. Such a result is sometimes known as an activity lemma [15].

Theorem 3 (Activity). *$H\{P\} \to R$ if and only if:*

(Act Proc) *there is a reduction $P \to P'$ with $R \equiv H\{P'\}$, or*
(Act Har) *there is a reduction $H \to H'$ with $R \equiv H'\{P\}$, or*
(Act Inter) *there are H' and $\vec{r}$ with $\{\vec{r}\} \cap fn(P) = \varnothing$, and one of the following holds:*

- **(Inter In)** $H \equiv (\nu\vec{r})H'\{m[- \mid R'] \mid n[R'']\}$, $P \xrightarrow{in\ n} P'$, *and* $R \equiv (\nu\vec{r})H'\{n[m[P' \mid R'] \mid R'']\}$
- **(Inter Out)** $H \equiv (\nu\vec{r})H'\{n[m[- \mid R'] \mid R'']\}$, $P \xrightarrow{out\ n} P'$, *and* $R \equiv (\nu\vec{r})H'\{m[P' \mid R'] \mid n[R'']\}$
- **(Inter Open)** $H \equiv (\nu\vec{r})H'\{- \mid n[R']\}$, $P \xrightarrow{open\ n} P'$, *and* $R \equiv (\nu\vec{r})H'\{P' \mid R'\}$
- **(Inter Input)** $H \equiv (\nu\vec{r})H'\{- \mid \langle M\rangle\}$, $P > (\nu\vec{p})\langle(x).P'\rangle P''$, *and* $R \equiv (\nu\vec{r})H'\{(\nu\vec{p})(P'\{x{\leftarrow}M\} \mid P'')\}$, *with* $\{\vec{p}\} \cap fn(M) = \varnothing$
- **(Inter Output)** $H \equiv (\nu\vec{r})H'\{- \mid (x).R'\}$, $P > (\nu\vec{p})\langle\langle M\rangle\rangle P'$, *and* $R \equiv (\nu\vec{r})H'\{(\nu\vec{p})(P' \mid R'\{x{\leftarrow}M\})\}$, *with* $\{\vec{p}\} \cap fn(R') = \varnothing$
- **(Inter Amb)** $P > (\nu\vec{p})\langle n[Q]\rangle P'$ *and one of the following holds:*
  1. $Q \xrightarrow{in\ m} Q'$, $H \equiv (\nu\vec{r})H'\{- \mid m[R']\}$, $\{\vec{p}\} \cap fn(m[R']) = \varnothing$, *and* $R \equiv (\nu\vec{r})H'\{(\nu\vec{p})(P' \mid m[n[Q'] \mid R'])\}$
  2. $Q \xrightarrow{out\ m} Q'$, $H \equiv (\nu\vec{r})H'\{m[- \mid R']\}$, $m \notin \{\vec{p}\}$, *and* $R \equiv (\nu\vec{r})H'\{(\nu\vec{p})(n[Q'] \mid m[P' \mid R'])\}$
  3. $H \equiv (\nu\vec{r})H'\{m[R' \mid in\ n.R''] \mid -\}$, $\{\vec{p}\} \cap fn(m[R' \mid in\ n.R'']) = \varnothing$, *and* $R \equiv (\nu\vec{r})H'\{(\nu\vec{p})(n[Q \mid m[R' \mid R'']] \mid P')\}$
  4. $H \equiv (\nu\vec{r})H'\{- \mid open\ n.R'\}$, $n \notin \{\vec{p}\}$, *and* $R \equiv (\nu\vec{r})H'\{(\nu\vec{p})(Q \mid P') \mid R'\}$

5 Examples of Contextual Equivalence

In this section, two examples demonstrate how we may apply Theorem 2 and Theorem 3 to establish contextual equivalence.

5.1 Opening an Ambient

We can now return to and prove Example 3 from Section 3.

Lemma 7. *If $H\{(\nu n)(n[] \mid open\ n.P)\} \Downarrow m$ and $n \notin fn(P)$ then $H\{P\} \Downarrow m$.*

Proof. By induction on the derivation of $H\{(\nu n)(n[] \mid open\ n.P)\} \Downarrow m$, with appeal to Propositions 4 and 5, and Theorems 1 and 3. □

Proof of Example 3 $(\nu n)(n[] \mid open\ n.P) \simeq P$ *if* $n \notin fn(P)$.

Proof. By Theorem 2, it suffices to prove $H\{((\nu n)(n[] \mid open\ n.P))\sigma\} \Downarrow m \Leftrightarrow H\{P\sigma\} \Downarrow m$ for all closed harnesses H and names m and for all substitutions σ with $dom(\sigma) = fv(P)$. Since the name n is bound, we may assume that $n \notin$

$fn(\sigma(x))$ for all $x \in dom(\sigma)$. Therefore, we are to prove that: $H\{(\nu n)(n[] \mid open\ n.P\sigma\} \Downarrow m \Leftrightarrow H\{P\sigma\} \Downarrow m$ where $n \notin fn(P\sigma)$.

We prove each direction separately. First, suppose that $H\{P\sigma\} \Downarrow m$. Since $(\nu n)(n[] \mid open\ n.P\sigma) \rightarrow P\sigma$, we get $H\{(\nu n)(n[] \mid open\ n.P\sigma)\} \rightarrow H\{P\sigma\}$. By (Exh Red), we get $H\{(\nu n)(n[] \mid open\ n.P\sigma)\} \Downarrow m$. Second, suppose that $H\{(\nu n)(n[] \mid open\ n.P\sigma)\} \Downarrow m$. By Lemma 7, we get $H\{P\sigma\} \Downarrow m$. □

5.2 The Perfect Firewall Equation

Consider a process $(\nu n)n[P]$, where n is not free in P. Since the name n is known neither inside the ambient $n[P]$, nor outside it, the ambient $n[P]$ is a "perfect firewall" that neither allows another ambient to enter nor to exit. The following two lemmas allow us to prove that $(\nu n)n[P]$ is contextually equivalent to $\mathbf{0}$, when $n \notin fn(P)$, which is to say that no context can detect the presence of $(\nu n)n[P]$.

Lemma 8. *If $H\{(\nu n)n[P]\} \Downarrow m$ and $n \notin fn(P)$ then $H\{\mathbf{0}\} \Downarrow m$.*

Proof. By induction on the derivation of $H\{(\nu n)n[P]\} \Downarrow m$.

(Conv Exh) Here $H\{(\nu n)n[P]\} \downarrow m$. By Proposition 5, either (1), for all Q, $H\{Q\} \downarrow m$, or (2), $(\nu n)n[P] \downarrow m$. In case (1), we have, in particular, that $H\{\mathbf{0}\} \downarrow m$. Hence, $H\{\mathbf{0}\} \Downarrow m$, by (Conv Exh). Case (2) cannot arise, since, by Proposition 4, $(\nu n)n[P] \downarrow m$ implies that $(\nu n)n[P] > (\nu \vec{p})\langle m[P']\rangle P''$ with $m \notin \{\vec{p}\}$, which is impossible.

(Conv Red) Here $H\{(\nu n)n[P]\} \rightarrow R$ and $R \Downarrow m$. By Theorem 3, one of three cases pertains:

- **(Act Proc)** Then $(\nu n)n[P] \rightarrow P''$ with $R \equiv H\{P''\}$. By Theorem 1, there is Q with $(\nu n)n[P] \xrightarrow{\tau} Q$ and $Q \equiv P''$. Since $(\nu n)n[P] > (\nu n)\langle n[P]\rangle\mathbf{0}$ is the only hardening derivable from $(\nu n)n[P]$, the transition $(\nu n)n[P] \xrightarrow{\tau} Q$ can only be derived using (Trans Amb), with $P \xrightarrow{\tau} P'$ and $Q = (\nu n)(n[P'] \mid \mathbf{0})$. Therefore, there is a reduction $P \rightarrow P'$ and $P'' \equiv (\nu n)n[P']$. We may show that $P \rightarrow P'$ implies $fn(P') \subseteq fn(P)$, and so $n \notin fn(P')$. We have $R \equiv H\{(\nu n)n[P']\}$ with $n \notin fn(P')$. By Lemma 1, we may derive $H\{(\nu n)n[P']\} \Downarrow m$ by the same depth of inference as $R \Downarrow m$. By induction hypothesis, $H\{\mathbf{0}\} \Downarrow m$.
- **(Act Har)** Then $H \rightarrow H'$ with $R \equiv H'\{(\nu n)n[P]\}$. By Lemma 1, we may derive $H'\{(\nu n)n[P]\} \Downarrow m$ by the same depth of inference as $R \Downarrow m$. By induction hypothesis, $H'\{\mathbf{0}\} \Downarrow m$. From $H \rightarrow H'$ we obtain $H\{\mathbf{0}\} \rightarrow H'\{\mathbf{0}\}$ in particular. By (Conv Red), we get $H\{\mathbf{0}\} \Downarrow m$.

(Act Inter) Then there are H' and $\vec{r}$ with $\{\vec{r}\} \cap fn(P) = \varnothing$ and one of several conditions must hold. Since the only hardening or transition from $(\nu n)n[P]$ is $(\nu n)n[P] > (\nu n)\langle n[P]\rangle\mathbf{0}$, only the rule (Inter Amb) applies. According to Theorem 3, there are four possibilities to consider.

(1) Here, $P \xrightarrow{in\ m} P'$, $H \equiv (\nu\vec{r})H'\{- \mid m[R']\}$, $\{n\} \cap fn(m[R']) = \varnothing$, and $R \equiv (\nu\vec{r})H'\{(\nu n)(\mathbf{0} \mid m[n[P'] \mid R'])\}$. We have $R \equiv (\nu\vec{r})H'\{m[R' \mid (\nu n)n[P']]\}$ and that $n \notin fn(P')$. By Lemma 1, we get $(\nu\vec{r})H'\{m[R' \mid (\nu n)n[P']]\} \Downarrow m$ with the same depth of inference as $R \Downarrow m$. By induction hypothesis, $(\nu\vec{r})H'\{m[R' \mid \mathbf{0}]\} \Downarrow m$. Moreover, $H\{\mathbf{0}\} \equiv (\nu\vec{r})H'\{m[R' \mid \mathbf{0}]\}$, and therefore $H\{\mathbf{0}\} \Downarrow m$.

(2) Here, $P \xrightarrow{out\ m} P'$, $H \equiv (\nu\vec{r})H'\{m[- \mid R']\}$, $m \notin \{n\}$, and also $R \equiv (\nu\vec{r})H'\{(\nu n)(n[P'] \mid m[\mathbf{0} \mid R'])\}$. We have $R \equiv (\nu\vec{r})H'\{m[R'] \mid (\nu n)n[P']\}$ and that $n \notin fn(P')$. By Lemma 1, we get $(\nu\vec{r})H'\{m[R'] \mid (\nu n)n[P']\} \Downarrow m$ with the same depth of inference as $R \Downarrow m$. By induction hypothesis, $(\nu\vec{r})H'\{m[R'] \mid \mathbf{0}\} \Downarrow m$. Moreover, $H\{\mathbf{0}\} \equiv (\nu\vec{r})H'\{m[R'] \mid \mathbf{0}\}$ and therefore $H\{\mathbf{0}\} \Downarrow m$.

The other possibilities, (3) and (4), are ruled out because the name n is restricted in the concretion $(\nu n)\langle n[P]\rangle\mathbf{0}$. □

By a similar induction, we can also prove:

Lemma 9. *If $H\{\mathbf{0}\} \Downarrow m$ then $H\{P\} \Downarrow m$.*

By combining Theorem 2, Lemmas 8 and 9, we get:

Example 4. If $n \notin fn(P)$ then $(\nu n)n[P] \simeq \mathbf{0}$.

Our first proof of this equation (which was stated in an earlier paper [6]) was by a direct quantification over all contexts. The proof above using the context lemma is simpler.

6 Conclusions

We developed a theory of Morris-style contextual equivalence for the ambient calculus. We showed that standard tools such as a labelled transition system, a context lemma, and an activity lemma, may be adapted to the ambient calculus. We introduced a new technique, based on a hardening relation, for defining the labelled transition system. We employed these tools to prove equational properties of mobile ambients.

Our use of concretions to highlight those subprocesses of a process that may participate in a computation follows Milner [10, 11], and is an alternative to the use of membranes and airlocks in the chemical abstract machine of Berry and Boudol [5]. Unlike these authors, in the definition of our transition relation we use the hardening relation, rather than the full structural congruence relation, to choose subprocesses to participate in a transition. Hardening is more convenient in some proofs, such as the proof that the labelled transition system is image-finite, Lemma 4.

In the future, it would be of interest to study bisimulation of ambients. Various techniques adopted for higher-order [13, 17] and distributed [4, 3, 16] variants of the π-calculus may be applicable to the ambient calculus.

Acknowledgement Comments by Cédric Fournet, Georges Gonthier, and Tony Hoare were helpful.

References

1. M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. In *Proceedings LICS'98*, pages 105–116, 1998.
2. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*. To appear. An extended version appears as Digital Equipment Corporation Systems Research Center report No. 149, January 1998.
3. R. M. Amadio. An asynchronous model of locality, failure, and process mobility. In *Proceedings COORDINATION 97*, volume 1282 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.
4. R. M. Amadio and S. Prasad. Localities and failures. In *Proceedings FST&TCS'94*, volume 880 of *Lecture Notes in Computer Science*, pages 205–216. Springer-Verlag, 1994.
5. G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, April 1992.
6. L. Cardelli and A. D. Gordon. Mobile ambients. In *Proceedings FoSSaCS'98*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer-Verlag, 1998.
7. L. Cardelli and A. D. Gordon. Types for mobile ambients. In *Proceedings POPL'99*, 1999. To appear.
8. R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.
9. R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–23, 1977.
10. R. Milner. The polyadic π-calculus: A tutorial. Technical Report ECS–LFCS–91–180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, October 1991.
11. R. Milner. The π-calculus. Undergraduate lecture notes, Cambridge University, 1995.
12. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100:1–40 and 41–77, 1992.
13. R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proceedings ICALP'92*, volume 623 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.
14. J. H. Morris. *Lambda-Calculus Models of Programming Languages*. PhD thesis, MIT, December 1968.
15. G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.
16. J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *Proceedings POPL'98*, pages 378–390, 1998.
17. D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1992. Available as Technical Report CST–99–93, Computer Science Department, University of Edinburgh.