Flow-sensitive Type Systems and the Ambient Calculus

Torben Amtoft
Department of Computing and Information Sciences
Kansas State University
Manhattan KS 66506, USA

March 31, 2007

Abstract

The Ambient Calculus was developed by Cardelli and Gordon as a formal framework to study issues of mobility and migrant code. Numerous analyses have been developed for numerous variants of that calculus. We take up the challenge of developing, in a *type-based setting*, a relatively precise "topology" analysis for the *original* version of the calculus. To compensate for the lack of "co-capabilities" (an otherwise increasingly popular extension), the analysis is flow-sensitive, with the actions of processes being summarized by "behaviors".

A subject reduction property guarantees that for a well-typed process, the location of any ambient is included in what is predicted by its type; additionally it ensures that communicating subprocesses agree on their "topic of conversation". Based on techniques borrowed from finite automata theory, type checking of type-annotated processes is decidable.

1 Introduction

The ambient calculus (AC) was developed by Cardelli & Gordon [CG98] as a framework for mobile computation where "ambients", containing active processes (and not just passive code), can move around—in and out of other ambients, thus forming a dynamic tree structure. The model also features communication, in that values can be exchanged between neighboring processes. Over the years, numerous variants and extensions of the "classical" ambient calculus have been proposed: "safe ambients" [LS00], "boxed ambients" [BCC01], BioAmbients [NNP04], etc.

Early type systems [CG99] for AC were flow-insensitive[1], designed to ensure that each ambient has a unique "topic of conversation"; this precludes configurations like

[1] That is, $M_1.M_2$ (first M_1 then M_2) and $M_2.M_1$ are analyzed the same way.

$$r[\langle 7\rangle \mid (z : \mathsf{int}).\mathsf{in}\ q.\langle z = 42\rangle] \mid q[\mathsf{open}\ r.(y : \mathsf{bool}).P] \tag{1}$$

where even though r allows both integers and booleans to be communicated, all values received are of the expected type, as seen by the fact that the only possible reduction sequence is

$$\begin{aligned}
(1) &\longrightarrow \text{(integer 7 is output and bound to } z\text{)}\\
r[\mathsf{in}\ q.\langle 7 = 42\rangle] \mid q[\mathsf{open}\ r.(y : \mathsf{bool}).P] &\longrightarrow \text{(ambient } r \text{ enters ambient } q\text{)}\\
q[r[\langle 7 = 42\rangle] \mid \mathsf{open}\ r.(y : \mathsf{bool}).P] &\longrightarrow \text{(ambient } r \text{ is dissolved inside } q\text{)}\\
q[\langle 7 = 42\rangle \mid (y : \mathsf{bool}).P] &\longrightarrow \text{(boolean is output and bound to } y\text{)}\\
q[P[y := \mathsf{false}]]
\end{aligned}$$

To allow for multiple topics of conversation, [AKPG01] proposed a flow-sensitive type system, assigning to each process a "behavior", containing temporal information. (This is akin to the session types [GH99], or the graph types [Yos96], for the π-calculus.)

A key challenge for any analysis of AC, with potential applications for security, is to predict the shape of the dynamic tree structures:

which ambients can end up where? (2)

This was first addressed by the flow-*in*sensitive analysis of [CGG00]. Note, however, that a precise answer to (2) requires flow-sensitivity, as illustrated by the two processes below which are equal modulo permutation of "capabilities":

$$p[\mathsf{open}\ q] \mid q[\mathsf{in}\ p.\mathsf{in}\ r.\mathsf{out}\ r] \mid r[\mathbf{0}] \tag{3}$$
$$p[\mathsf{open}\ q] \mid q[\mathsf{in}\ r.\mathsf{out}\ r.\mathsf{in}\ p] \mid r[\mathbf{0}] \tag{4}$$

Both processes are deterministic, with reduction sequences given by

$$\begin{aligned}
(3) &\longrightarrow\\
p[q[\mathsf{in}\ r.\mathsf{out}\ r] \mid \mathsf{open}\ q] \mid r[\mathbf{0}] &\longrightarrow\\
p[\mathsf{in}\ r.\mathsf{out}\ r] \mid r[\mathbf{0}] &\longrightarrow\\
r[p[\mathsf{out}\ r]] &\longrightarrow\\
p[\mathbf{0}] \mid r[\mathbf{0}]
\end{aligned}
\qquad
\begin{aligned}
(4) &\longrightarrow\\
p[\mathsf{open}\ q] \mid r[q[\mathsf{out}\ r.\mathsf{in}\ p]] &\longrightarrow\\
p[\mathsf{open}\ q] \mid q[\mathsf{in}\ p] \mid r[\mathbf{0}] &\longrightarrow\\
p[q[\mathbf{0}] \mid \mathsf{open}\ q] \mid r[\mathbf{0}] &\longrightarrow\\
p[\mathbf{0}] \mid r[\mathbf{0}]
\end{aligned}$$

We see that (3) allows p to be located inside r, whereas (4) does not. The difference is that when p opens q in (3), q has an $\mathsf{in}\ r$ "capability" left, whereas when p opens q in (4), that capability has already been consumed. The implications are that *when analyzing the occurrence of* $\mathsf{open}\ q$ *in* p*, we must know which of the capabilities of* q *are still left*, as those will be the ones "unleashed" within p. This cannot be done in a flow-insensitive setting, a fact that was early recognized as a general obstacle to a precise analysis, cf. the discussion in [CGG99] where a naive analysis cannot declare immobile an ambient that opens a packet which *has* moved.

The above conundrum can be resolved at the *language level*, most radically by removing open from the language, as first done in boxed ambients [BCC01] (at the price of allowing communication not just between siblings but also between

parent and child). The most popular approach, however, is to let the execution of capabilities be multilateral rather than unilateral. In particular, in order for p to execute $\mathsf{open}\ q$, the ambient q must be ready to execute a corresponding "co-open". This device was first employed in *safe ambients* [LS00] and later employed, e.g., in [AKPG01, GYY01]. (Safe ambients additionally include co-in and co-out.) In the language of [AKPG01], the process (1) becomes

$$r[\langle 7\rangle \mid (z : \mathsf{int}).\mathsf{in}\ q.\mathsf{co\text{-}open}\ r.\langle z = 42\rangle] \mid q[\mathsf{open}\ r.(y : \mathsf{bool}).P]$$

and it is now easy to give r the type $\mathsf{amb}[\mathsf{put}(\mathsf{bool})]$ saying that when r is opened, it will unleash the writing of a boolean—and nothing more. Therefore it is safe for q, expecting a boolean, to open r, even though inside r there is a subprocess writing an integer.

To address (2), we could apply the same device and, e.g., require the process (3) to be rewritten into

$$p[\mathsf{open}\ q] \mid q[\mathsf{in}\ p.\mathsf{co\text{-}open}\ q.\mathsf{in}\ r.\mathsf{out}\ r] \mid r[\mathbf{0}]$$

from which it is easy to see what is unleashed when q is opened. But we shall **aim at an analysis which is able to address (2) for classical** AC.

Several such analyses have been proposed, based on various paradigms. *Control flow analysis* is employed in [NNHJ99]. *Abstract interpretation* is employed in [HJNN99], and in [LM04] where one analysis keeps track of the context one level up; this is sufficient to achieve a quite precise analysis, yet is "only" polynomial (n^7). [NN01] employs *tree grammars*, computing a set of grammars such that at any step in the reduction process, the current tree structure can be described by one of these grammars—the method is very precise, but potentially also very expensive. A *3-valued logic* is employed in [NNS00] to estimate the possible shapes of the tree structure; the framework allows for trade-offs wrt. precision versus costs.

During the summer of 2001, the author succeeded in solving (2) in the setting of *type systems*, as reported and detailed in [Amt02], and to be highlighted in this paper. The **main contributions of that work** are that *(i)* it provides the first type-based system which keeps track of how the location of an ambient changes over time, thus allowing a precise estimate concerning when it can be opened; *(ii)* the type system smoothly incorporates a communication analysis which allows for multiple topics of conversation, whereas the analyses listed in the previous paragraph are for a communication-free subset of classical AC. A key feature of the approach is that behaviors are finite automata, allowing (as in [AKPG01]) type checking to use techniques from automaton theory.

In 2002, the author took part in the early development of PolyA, an even more general type system [AMW04]. Here a process is given a type which is essentially an upper bound of all shapes it can evolve to (much as what is done in [NNP04] for BioAmbients). The analysis is very precise, and also smoothly incorporates communication; a spin-off is the generic tool Poly [MW05] which can handle a very broad range of mobile calculi. Still, one can argue that this approach is not

in the proper "spirit" of type systems[2], since types are almost indistinguishable from processes (as is also the case in [CDC02]), and that [Amt02] is still the solution to Question 2 that best conforms to the paradigm of type and effect systems. In subsequent sections, we shall motivate and present that system. Proofs can be found in Appendix (or alternatively in [Amt02]).

2 Tracking Locations

We now illustrate how to keep track of the location of an ambient, so as to estimate when it is opened. For the process in (4), we observe that q starts being at top-level, which we write as "q is in #" where # is a "global" ambient. After executing $\mathsf{in}\ r$, q will be enclosed[3] by r. After executing $\mathsf{out}\ r$, it is not immediately clear where q will be, since r might have moved while containing q. This motivates that the type of an ambient should tell where the ambient may be located (cf. the type system for boxed ambients in [MS02]). In this case, the information that r is enclosed by # only, will enable us to infer that q will again be enclosed by #. Finally, q gets enclosed by p after executing $\mathsf{in}\ p$.

We conclude that q may be enclosed by either of r, p, or #, enabling us to give q the type $\mathsf{amb}_{rp\#}$. Moreover, we see that when q is enclosed by p, no capabilities are left, enabling us to further give q the type $\mathsf{amb}_{rp\#}[\{p : \varepsilon\}]$ where ε denotes the "empty" behavior. The $\{p : \varepsilon\}$ component expresses that when opened inside p, the empty behavior is "unleashed". Thus p doesn't gain any capabilities by opening q, so we can give p the type $\mathsf{amb}_{\#}$ which in particular shows that *p is never enclosed by r*.

Now look at (3), where the process within q may be given the behavior

$$^{?}\mathsf{enter}(p).\,^{\#}\mathsf{enter}(r).\,^{r}\mathsf{exit}(\#) \tag{5}$$

where we have exploited that r has type $\mathsf{amb}_{\#}$ (so that r can only be entered by an ambient which is enclosed by #). This behavior may seem contradictory: how can q first enter p, and next enter r *from #?* The explanation is that q has been *opened* when in p, justifying that we give q the type

$$\mathsf{amb}_{p\#}[\{p : \,^{\#}\mathsf{enter}(r).\,^{r}\mathsf{exit}(\#)\}]$$

From this typing we see that p by opening q gets a capability to enter r, so if amb_H is a typing for p then H has to include r. As expected, the analysis does not rule out that p may be enclosed by r.

Example 1 *As a larger example, consider the firewall first presented in [CG98]:*

$$w[k[\mathsf{out}\ w.\mathsf{in}\ k'.\mathsf{in}\ w] \mid \mathsf{open}\ k'.\mathsf{open}\ k''.P] \mid k'[\mathsf{open}\ k.k''[Q]]$$

[2]Anticipating that criticism, [MW05] invites the skeptical reader to instead use the term "versatile program analysis framework"!

[3]When writing "enclosed", we always mean "directly enclosed".

This process is deterministic: k will exit w and enter k' where it is dissolved; then k' will enter w where it is dissolved and afterwards k'' (carried into w by k') is also dissolved. Assuming P and Q do nothing of interest, we can type the ambients as follows:

$$\begin{array}{lcllcl} k & : & \mathsf{amb}_{wk'\#}[k' : {}^{\#}\mathsf{enter}(w)] & w & : & \mathsf{amb}_{\#} \\ k' & : & \mathsf{amb}_{w\#}[w : \varepsilon] & k'' & : & \mathsf{amb}_{wk'}[w : \varepsilon] \end{array}$$

In the non-causal analysis of [NNHJ99], w as well as k' can contain all of w, k, k', k''. As several other analyses in the recent literature, we are thus much more precise.

A problem with the above typing is that the "secret" name w appears in the typings of k' and k''. This motivates the introduction of *groups*, as in [CGG00], with the intention that each ambient belongs to exactly one group $g \in \mathsf{Grps}$ (a finite set). With W the group of w, etc, we have, e.g., $k' : \mathsf{amb}^{K'}_{W\#}[W : \varepsilon]$

Our type system is (implicitly) parameterized with respect to a set of dynamic "security constraints" of the form $\mathcal{O}(g_0, g)$, saying that an ambient of group g_0 is allowed to be opened while enclosed by an ambient of group g. One can view these constraints as prescriptive (thus provided by the user); establishing that a process is well-typed then verifies that no other interactions may happen. Alternatively, one can take a descriptive view: a type inference algorithm might deduce the least set of constraints needed for typability. In Example 1 we have $\mathcal{O}(K, K')$, $\mathcal{O}(K', W)$, $\mathcal{O}(K'', W)$.

We shall use G to range over sets of groups, and use H to range over *upwards closed* sets of groups, where G is upwards closed if $g \in G$ and $\mathcal{O}(g, g')$ implies $g' \in G$. The intuition is that if an ambient n can be directly enclosed in g, and g' can open g, then n might also be directly enclosed in g'. We let $g^{\uparrow}$ denote the least upwards closed set containing g. In Example 1, we have ${K'}^{\uparrow} = \{K', W\}$.

3 The Language

Processes and expressions are given by the syntax (we often write M for $M.\mathbf{0}$)

$$\begin{array}{rcl} P, Q, R \in \mathsf{Proc} & ::= & \mathbf{0} \mid P_1 \mid P_2 \mid !P \mid (\nu n : \tau).P \mid M.P \mid M[\![P]\!]^{\xi} \\ & \mid & (n_1 \ldots n_k : \tau_1 \ldots \tau_k).P \mid \langle M_1 \ldots M_k \rangle \\ M \in \mathsf{Exp} & ::= & n \mid \mathsf{in}\ M \mid \mathsf{out}\ M \mid \mathsf{open}\ M \mid \epsilon \mid M_1.M_2 \end{array}$$

which is much as in [CG99], except we annotate each ambient $M[\![P]\!]^{\xi}$ with a unique *tag* ξ, so as to distinguish between ambients with the same name (as is customary in flow logic [NNHJ99]). We shall assume a function *group*, extracting the group of an ambient from its tag. We write $P \equiv_{\mathrm{t}} P'$ if *(i)* P and P' are equal except for the tags; and *(ii)* if P at some position has tag ξ then P' at the same position has a tag χ with $\mathit{group}(\xi) = \mathit{group}(\chi)$.

For the binding constructs, each name being bound is annotated with a type (to be defined in Sect. 4). The set of all names (free or bound) occurring in P is denoted $\mathsf{names}(P)$. We say that a process P is non-conflicting with a set of names X if *(i)* no name is bound more than once in P, and *(ii)* a name bound in P does not occur in X.

3.1 Operational Semantics

The semantics of AC is presented in Fig. 1. We write $P_1 \xrightarrow{\ell} P_2$ if P_1 reduces in one step to P_2 by performing "an action described by ℓ". Note that the definition of this relation deviates from the standard in that in (Red Open) and (Red Comm) we provide the surrounding ambient so as to record in ℓ *where* the opening or communication has taken place. We write $P_1 \equiv P_2$ to denote that P_1 and P_2 are equivalent, modulo "syntactic rearrangement", and modulo consistent renaming of bound names. The latter may be needed to apply (Red Comm) since to avoid name capture this rule has a side condition

$$(n_1 \ldots n_k : \tau_1 \ldots \tau_k).P \text{ is non-conflicting with } \mathsf{names}(M_1, \ldots, M_k)$$

The relation $\equiv$ (defined in Appendix A) is much as in [CG99], containing rules like $P \mid \mathbf{0} \equiv P$ and $P \mid Q \equiv Q \mid P$, except that we omit (as otherwise it appears hard to establish "subject congruence") the rule $!P \equiv P \mid !P$ and instead allow this "unfolding" to take place via the rule (Red Repl) (as also seen in, e.g., [NNS00]) To permit also the reduct to be uniquely tagged, this rule actually allows $!P$ to be unfolded into $P' \mid !P$ where P' is a "copy" of P.

We use evaluation contexts $\mathcal{PC}$ to succinctly describe the place in a process where a subprocess (Red PctxtP) is reduced, and write $\mathcal{PC}\lfloor P \rfloor$ for the process resulting from replacing the hole in $\mathcal{PC}$ by P. Note that $P \xrightarrow{\ell} Q$ does not imply that $M.P \xrightarrow{\ell} M.Q$ since the capability M must be executed before P can be activated.

4 Types and Behaviors

We have hinted at the form of types τ and behaviors b, to be defined mutually recursively in Fig. 2.

Concerning behaviors, we have seen the need for constants like ε, and we clearly also need operators like $b_1 \mid b_2$ (to model parallel composition) and $a.b$ (to model sequential prefixing). Ultimately, a behavior must approximate sets of traces, so why not—also to give the user more freedom in specification—let a behavior *be* a nonempty regular set of finite traces? (This is unlike what we did in [AKPG01], but similar to [CC91] where recursive types are infinite trees, or [RV97] where types are graphs.)

We can then *define* the above operators: ε as $\{\bullet\}$ where $\bullet$ is the empty sequence; $a.b$ as $\{a \diamond tr \mid tr \in b\}$ where $\diamond$ denotes concatenation; and $b_1 \mid b_2$ as $\bigcup_{tr_1 \in b_1, tr_2 \in b_2} tr_1 \parallel tr_2$ where $tr_1 \parallel tr_2$ denotes all traces that can be formed by arbitrarily interleaving tr_1 with tr_2. We write a for $a.\varepsilon = \{a\}$.

Labels ℓ	$::=$	ϵ	only "internal computation" is performed
	$\mid$	$\xi : \mathsf{enter}\ \chi$	the ambient ξ is steered into the ambient χ
	$\mid$	$\xi : \mathsf{exit}\ \chi$	the ambient ξ is steered out of the ambient χ
	$\mid$	$\xi : \mathsf{open}\ \chi$	a process controlling the ambient ξ opens the ambient χ
	$\mid$	$\xi : \mathsf{comm}\ \sigma$	inside the ambient ξ values of type σ are communicated
Contexts $\mathcal{PC}$	$::=$	$\Box \mid (\mathcal{PC} \mid P) \mid (\nu n : \tau).\mathcal{PC} \mid n[\![\mathcal{PC}]\!]^{\xi}$	

$$m[\![\mathsf{in}\ n.P \mid Q]\!]^{\xi} \mid n[\![R]\!]^{\chi} \xrightarrow{\xi:\mathsf{enter}\ \chi} n[\![m[\![P \mid Q]\!]^{\xi} \mid R]\!]^{\chi} \qquad (\mathsf{Red\ In})$$

$$n[\![m[\![\mathsf{out}\ n.P \mid Q]\!]^{\xi} \mid R]\!]^{\chi} \xrightarrow{\xi:\mathsf{exit}\ \chi} m[\![P \mid Q]\!]^{\xi} \mid n[\![R]\!]^{\chi} \qquad (\mathsf{Red\ Out})$$

$$m[\![\mathsf{open}\ n.P \mid n[\![Q]\!]^{\chi} \mid R]\!]^{\xi} \xrightarrow{\xi:\mathsf{open}\ \chi} m[\![P \mid Q \mid R]\!]^{\xi} \qquad (\mathsf{Red\ Open})$$

$$m[\![(n_1 \ldots n_k : \tau_1 \ldots \tau_k).P \mid \langle M_1 \ldots M_k \rangle \mid Q]\!]^{\xi} \xrightarrow{\xi:\mathsf{comm}\ \times(\tau_1,\ldots,\tau_k)} m[\![P[n_i := M_i] \mid Q]\!]^{\xi} \qquad (\mathsf{Red\ Comm})$$

$$!P \xrightarrow{\epsilon} P' \mid !P \quad \text{if } P' \equiv_{\mathrm{t}} P \qquad (\mathsf{Red\ Repl})$$

$$\text{If } P \xrightarrow{\ell} Q \text{ then } \mathcal{PC}\lfloor P \rfloor \xrightarrow{\ell} \mathcal{PC}\lfloor Q \rfloor \qquad (\mathsf{Red\ PctxtP})$$

$$\text{If } P' \equiv P, P \xrightarrow{\ell} Q, Q \equiv Q' \text{ then } P' \xrightarrow{\ell} Q' \qquad (\mathsf{Red}\ \equiv)$$

Figure 1: Operational Semantics.

$a \in \mathsf{Actions}$	$::=$	${}^{H}\mathsf{enter}(g)$	steers an ambient from H to g
	$\mid$	${}^{g}\mathsf{exit}(H)$	steers an ambient from g to H
	$\mid$	${}_{G}\mathsf{open}(g)$	opens g; if executed in G then unleashed behavior correct
	$\mid$	$\mathsf{put}(\sigma)$	output tuple of type σ
	$\mid$	$\mathsf{get}(\sigma)$	input tuple of type σ
$tr \in \mathsf{Traces}$	$::=$	a^*	finite sequence of actions
$b \in \mathsf{Behaviors}$	$\subseteq$	$\mathcal{P}(\mathsf{Traces})$	non-empty regular set of traces
$B \in \mathsf{BehContxt}$	$::=$	$\Box \mid a.B \mid (b \mid B)$	
$br \in \mathsf{BehRows}$	$::=$	$\{g_i : b_i\}_{i \in I}$	behaves as b_i when opened in g_i
$\sigma \in \mathsf{Tuples}$	$::=$	$\times(\tau_1, \ldots, \tau_k)$	we write τ_1 if $k = 1$
$\tau \in \mathsf{Types}$	$::=$	$\mathsf{amb}^{g}_{H}[br]$	type of ambient name
	$\mid$	$\mathsf{cap}[B]$	type of capability
	$\mid$	$\mathsf{int} \mid \mathsf{real} \mid \mathsf{bool}$	base types (optional)

Figure 2: Syntax of Types and Behaviors.

An ambient n has a type of the form $\mathsf{amb}^g_H[\{g_i : b_i\}_{i\in I}]$ which (cf. the Introduction) should be read as follows: it has group g, can be directly enclosed by ambients of groups belonging to H, and after being opened inside g_i it behaves as b_i. For all $i \in I$, we thus require $\mathcal{O}(g, g_i)$.

Concerning types of the form $\mathsf{cap}[B]$, they are for the typing of capabilities (like $\mathsf{open}\ n$). Here B is a *behavior context*, that is a "behavior with a hole inside"; we write $B\lfloor b \rfloor$ for the result of "plugging" b into the hole of B. The notion of behavior contexts was introduced in [AKPG01] and used also in, e.g., [GYY01]); it conveniently expresses the result of prefixing, as illustrated by the situation where n has type $\mathsf{amb}^g_H[g_0 : b_0]$ and P has behavior b. Then the process $\mathsf{open}\ n.P$ will open g and then run a process described by b_0 in parallel with a process described by b. And in fact, referring to Fig. 3, ${}_{g_0}\mathsf{open}(g).(b_0 \mid b)$ is the result of (Proc Action) plugging b into $\mathsf{cap}[{}_{g_0}\mathsf{open}(g).(b_0 \mid \Box)]$ which by (Exp Open) is a possible type of $\mathsf{open}\ n$.

5 Ordering Relations

The type system (Fig. 3) uses subsumption rules, based on an ordering $\tau_1 \leqslant \tau_2$ on types (subtyping), and an ordering $b_1 \leqslant b_2$ on behaviors (subbehaviors). These orderings are defined in a mutually recursive way, together with orderings on actions, traces, etc. To ensure that this is well-defined, we shall employ the notion of *level*: an entity has level i if i is an upper bound of the depth of nested occurrences of $\mathsf{amb}^{\_}_{\_}[\_]$ or $\mathsf{cap}[\_]$ within it. (We use "$\_$" to stand for an arbitrary entity of the appropriate kind.) Example: $\mathsf{put}(\mathsf{cap}[\{\mathsf{put}(\mathsf{amb}^g_H)\} \mid \Box])$ has level two. Then, as detailed in the subsequent paragraphs, a relation on level i types induces a relation on level i tuples which induces a relation on level i actions which induces a relation on level i traces which induces a relation on level i behaviors which induces a relation on level i behavior rows and contexts which in turn induces a relation on level $i + 1$ types.

The relation $a_1 \leqslant a_2$, with the intuitive interpretation that a_2 is more "permissive" than a_1, can be summarized by stating that the constructors have the following polarity:

$$^{\oplus}\mathsf{enter}(=) \qquad ^{=}\mathsf{exit}(\oplus) \qquad {}_{\ominus}\mathsf{open}(=) \qquad \mathsf{put}(\oplus) \qquad \mathsf{get}(\ominus)$$

Concerning the polarity of ${}^{g}\mathsf{exit}(H)$, the intuition is that if the ambient as a result of leaving g will enter an ambient whose group is in H, this group also belongs to any set containing H; similarly for ${}^{H}\mathsf{enter}(g)$. For ${}_{G}\mathsf{open}(g)$, the intuition is that if a sufficient condition for the unleashed behavior to be correct is that the process is in an ambient with group in G, belonging to a subset of G is also a sufficient condition. Concerning the actions for communication, we have that $\mathsf{put}(\mathsf{int}) \leqslant \mathsf{put}(\mathsf{real})$, since a process that sends an integer thereby also sends a real number, and $\mathsf{get}(\mathsf{real}) \leqslant \mathsf{get}(\mathsf{int})$, since a process that accepts a real number also will accept an integer.

Concerning the relation $b_1 \leqslant b_2$, with the intuitive interpretation that b_2 is more "permissive" than b_1, we expect, e.g., $a_1.a_2 \leqslant a_1 \mid a_2$ since the right hand

side does not prescribe which action comes first. We stipulate that $b_1 \leqslant b_2$ iff for all $tr_1 \in b_1$ there exists $tr_2 \in b_2$ such that $tr_1 \leqslant tr_2$. Here the relation $tr_1 \leqslant tr_2$ is the pointwise extension of the relation $a_1 \leqslant a_2$ (if $tr_1 \leqslant tr_2$ then tr_1 and tr_2 have the same length). Note that $b_1 \leqslant b_2$ and $b_2 \leqslant b_1$ does not necessarily imply $b_1 = b_2$ (let $b_1 = \{a_1\}$ and let $b_2 = \{a_1, a_2\}$ with $a_2 \leqslant a_1$).

The relation $\leqslant$ on behavior rows is defined as follows: with $br = \{g_i : b_i\}_{i \in I}$ and $br' = \{g'_j : b'_j\}_{j \in J}$, $br \leqslant br'$ holds iff for all $j \in J$ there exists $i \in I$ such that $g_i = g'_j$ and $b_i \leqslant b'_j$.

The relation $\leqslant$ on behavior contexts is defined by stipulating that $B_1 \leqslant B_2$ holds iff for all (level 0) behaviors b we have $B_1\lfloor b \rfloor \leqslant B_2\lfloor b \rfloor$. For example, for all actions a we have $a.\Box \leqslant a \mid \Box$, since $a.b \leqslant a \mid b$ holds for all b. The restriction to level 0 behaviors is formally needed, in order for the relation on level i behavior contexts to be determined by the relation on level i behaviors, but (much as in [AKPG01]) turns out to be superfluous, thanks to the following result (proved in Appendix C):

Proposition 1 *Given B_1 and B_2, we can construct an action* test *of level zero such that the following conditions are equivalent:*

(a) $B_1 \leqslant B_2$

(b) $B_1\lfloor b \rfloor \leqslant B_2\lfloor b \rfloor$ *for all b (regardless of level)*

(c) $B_1\lfloor \mathsf{test} \rfloor \leqslant B_2\lfloor \mathsf{test} \rfloor$.

The relation $\tau_1 \leqslant \tau_2$, with the intuitive interpretation that an expression of type τ_1 also has type τ_2, can be summarized by stating that $\mathsf{int} \leqslant \mathsf{real}$ and that we have the polarity $\mathsf{cap}[\oplus]$. Concerning the polarity of the type $\mathsf{amb}^g_H[br]$, the proof of subject reduction (Theorem 3) reveals that this type must be both covariant and contravariant, in H, to deal with (Red In) and (Red Out), as well as in br, to deal with (Red Open). We could thus (as in [AKPG01]) play the trick of [Zim00] (akin to the "split types" of [BPG00] for an object-oriented calculus) and split each of these arguments into two, one contravariant (to be used in (Proc Amb)) and the other covariant (to be used in (Exp In), (Exp Out), (Exp Open)). But to keep things simple, we refrain from doing so.

6 Operations on Traces

Given an ambient n, controlled by a process P and residing within an ambient whose group belongs to H, we want to estimate the destination of n after (partial) execution of P. For this purpose, we define the upwards closed sets

$Dest(H, a)$ and $Dest(H, tr)$ as follows:

$$\begin{aligned}
Dest(H,\ ^{H_0}\mathsf{enter}(g)) &= \text{if } H \cap H_0 \neq \emptyset \text{ then } g^{\uparrow} \text{ else } \emptyset \\
Dest(H,\ ^{g}\mathsf{exit}(H_0)) &= \text{if } g \in H \text{ then } H_0 \text{ else } \emptyset \\
Dest(H, a) &= H \text{ otherwise} \\
Dest(H, \bullet) &= H \\
Dest(H, a \diamond tr) &= Dest(Dest(H, a), tr)
\end{aligned}$$

The idea behind the first line is that an ambient residing inside H can only enter an ambient g if they are "siblings", a necessary condition for which is that $H \cap H_0 \neq \emptyset$ where H_0 are the possible surroundings for g. Clearly, $Dest(\emptyset, tr) = \emptyset$ for all tr. On the other hand, it may happen that $Dest(H, tr) = \emptyset$ even if $H \neq \emptyset$; this corresponds to a "jump of location", an example of which is the trace in (5).

We say that a trace tr is *saturated* if all occurrences of $\mathsf{put}(\_)$ and $\mathsf{get}(\_)$ in tr come in pairs, with $\mathsf{put}(\_)$ immediately preceding $\mathsf{get}(\_)$. We shall have particular interest in *feasible* traces, the intuition being that such a trace can execute "on its own", without environment interaction.

Definition 1 *We say that a trace tr is* feasible *from H if* (i) *tr is saturated, and* (ii) *$Dest(H, tr) \neq \emptyset$.*

A non-feasible trace still conveys useful information: it may be interleaved with another trace, resulting in a feasible trace; or a suffix may be unleashed by an opening process.

7 Type System

Our type system is defined in Fig. 3, where E is an environment mapping names to types. For expressions, the judgments are of the form $E \vdash M : \tau$; we have already motivated the rule (Exp Open) and the other rules are similar or simpler. For processes, the judgments are of the form $\Delta, E \vdash_g P : b$, where Δ maps an ambient tag into the behavior of the process inside that ambient, and where g is the group of the enclosing ambient (to be used in the rule (Proc Amb)). We have already motivated the rule (Proc Action). Concerning the rule (Proc Repl), the side condition ensures that the behavior of a replicated process is in fact invariant under replication.

The crux of the type system is the rule (Proc Amb), where the situation is that an ambient tagged ξ and with type $\mathsf{amb}_H^{g_0}[br]$ contains a process P and is enclosed by an ambient of group g. We need to check that H contains (the groups of) all ambients that may possibly contain ξ, and therefore need to know the initial location of ξ which is $g^{\uparrow}$. When typing P, the enclosing ambient is g_0, the group of ξ, and we must be able to assign P the behavior $b = \Delta(\xi)$. Moreover, it must hold that $(g^{\uparrow}, b) \overset{g_0}{\rightsquigarrow} (H, br)$, defined as follows:

Non-structural Rules

$$\text{(Proc Subsumption)}\quad \frac{\Delta, E \vdash_g P : b}{\Delta, E \vdash_g P : b'} \quad (b \leqslant b') \qquad \text{(Exp Subsumption)}\quad \frac{E \vdash M : \tau}{E \vdash M : \tau'} \quad (\tau \leqslant \tau')$$

Expressions

$$\text{(Exp } n\text{)}\quad \frac{E(n) = \tau}{E \vdash n : \tau} \qquad \text{(Exp } \epsilon\text{)}\quad \frac{}{E \vdash \epsilon : \mathsf{cap}[\Box]} \qquad \text{(Exp Action)}\quad \frac{E \vdash M_1 : \mathsf{cap}[B_1] \quad E \vdash M_2 : \mathsf{cap}[B_2]}{E \vdash M_1.M_2 : \mathsf{cap}[B_1 \lfloor B_2 \rfloor]}$$

$$\text{(Exp In)}\quad \frac{E \vdash M : \mathsf{amb}^g_H[br]}{E \vdash \mathsf{in}\ M : \mathsf{cap}[{}^H\mathsf{enter}(g).\Box]} \qquad \text{(Exp Out)}\quad \frac{E \vdash M : \mathsf{amb}^g_H[br]}{E \vdash \mathsf{out}\ M : \mathsf{cap}[{}^g\mathsf{exit}(H).\Box]}$$

$$\text{(Exp Open)}\quad \frac{E \vdash M : \mathsf{amb}^g_H[\{g_i : b_i\}_{i \in I}]}{E \vdash \mathsf{open}\ M : \mathsf{cap}[_G\mathsf{open}(g).(b \mid \Box)]} \quad \text{if } \forall g' \in G : \exists i \in I : g' = g_i \text{ and } b_i \leqslant b$$

Processes

$$\text{(Proc Zero)}\quad \frac{}{\Delta, E \vdash_g \mathbf{0} : \varepsilon} \qquad \text{(Proc Par)}\quad \frac{\Delta, E \vdash_g P_1 : b_1 \qquad \Delta, E \vdash_g P_2 : b_2}{\Delta, E \vdash_g P_1 \mid P_2 : b_1 \mid b_2} \qquad \text{(Proc Repl)}\quad \frac{\Delta, E \vdash_g P : b}{\Delta, E \vdash_g\ !P : b} \quad \text{if } (b \mid b) \leqslant b$$

$$\text{(Proc Res)}\quad \frac{\Delta, E, n : \mathsf{amb}^{g_0}_H[br] \vdash_g P : b}{\Delta, E \vdash_g (\nu n : \mathsf{amb}^{g_0}_H[br]).P : b} \qquad \text{(Proc Action)}\quad \frac{E \vdash M : \mathsf{cap}[B] \qquad \Delta, E \vdash_g P : b}{\Delta, E \vdash_g M.P : B \lfloor b \rfloor}$$

$$\text{(Proc Amb)}\quad \frac{E \vdash M : \mathsf{amb}^{g_0}_H[br] \quad \Delta, E \vdash_{g_0} P : b}{\Delta, E \vdash_g M[\![P]\!]^\xi : \varepsilon} \quad \text{if} \quad \begin{array}{l} (g^\uparrow, b) \overset{g_0}{\rightsquigarrow} (H, br) \\ \mathit{group}(\xi) = g_0 \\ \Delta(\xi) = b \end{array}$$

$$\text{(Proc Input)}\quad \frac{\Delta, E, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash_g P : b}{\Delta, E \vdash_g (n_1 \ldots n_k : \tau_1 \ldots \tau_k).P : \mathsf{get}(\sigma).b} \quad \sigma = \times(\tau_1, \ldots, \tau_k) \qquad \text{(Proc Output)}\quad \frac{\forall i \in \{1 \ldots k\} : E \vdash M_i : \tau_i}{\Delta, E \vdash_g \langle M_1 \ldots M_k \rangle : \mathsf{put}(\sigma)}$$

Figure 3: Typing Rules.

Definition 2 *We write* $(H_0, b) \overset{g_0}{\rightsquigarrow} (H, br)$ *iff whenever* b *contains a trace* tr *of the form* $tr_1 \diamond tr_2$ *with* tr_1 *feasible from* H_0*, then with* $H_1 = Dest(H_0, tr_1)$

1. H_1 *is a subset of* H*;*
2. *If* tr_2 *starts with* $\mathsf{put}(\sigma_1)\,\mathsf{get}(\sigma_2)$ *then* $\sigma_1 \leqslant \sigma_2$*;*
3. *If* tr_2 *starts with* ${}_G\mathsf{open}(\_)$ *then* $g_0 \in G$*;*
4. *With* $br = \{g_i : b_i\}_{i \in I}$*, for all* $i \in I$*: if* $g_i \in H_1$ *then* $\{tr_2\} \leqslant b_i$*.*

Here (1) ensures that H is indeed is an upper bound of where ξ may end up; (2) ensures that there is always agreement on the topic of conversation; (3) ensures that whenever an ambient is opened, it is in an expected place and hence the unleashed behavior is correct; (4) ensures that if ξ after executing tr_1 can be inside g_i, then tr_2 must be in b_i since b_i approximates what happens after ξ is opened within g_i.

Theorem 2 (Type Safety) *Assume we can derive*

$$\Delta, E \vdash_{\_} m[\![n[\![Q]\!]^{\chi} \mid \ldots]\!]^{\xi} : \_$$

with $E(n) = \mathsf{amb}_H^{g_0}[\_]$*. Then* $group(\xi) \in H$*.*

Since a process can be typed only if all subprocesses can be typed, this shows that the type of an ambient does safely estimate its possible surrounding.

Proof: From (Proc Amb), we infer that with $g = group(\xi)$ we have $\Delta, E \vdash_g n[\![Q]\!]^{\chi} \mid \ldots : \_$. Again from (Proc Amb), we next infer that $(g^{\uparrow}, \_) \overset{g_0}{\rightsquigarrow} (H, \_)$ holds. But this implies $g^{\uparrow} \subseteq H$ (as can be seen from taking $tr_1 = \bullet$ in Def. 2), and thus $g \in H$. ∎

8 Semantic Soundness

That our type system is semantically sound will be formulated as a subject reduction result, intuitively stating that "well-typed processes never evolve into ill-typed processes" and also stating that "well-typed processes behave according to their behavior". To express the result in a succinct way, we introduce a relation $\Delta \xrightarrow{\ell} \Delta'$ which captures that the behaviors in Δ evolves into Δ' as predicted by the reduction label ℓ. Formally, we have

Definition 3 *The relation* $\Delta \xrightarrow{\ell} \Delta'$ *holds iff*

- Δ' *agrees with* Δ *on* $\mathrm{dom}(\Delta) \setminus dom(\ell)$*, and*
- *if* $\ell = \xi : \mathsf{enter}\ \chi$ *then* ${}^{\emptyset}\mathsf{enter}(group(\chi)).\Delta'(\xi) \leqslant \Delta(\xi)$*;*
- *if* $\ell = \xi : \mathsf{exit}\ \chi$ *then* ${}^{group(\chi)}\mathsf{exit}(\emptyset).\Delta'(\xi) \leqslant \Delta(\xi)$*;*
- *if* $\ell = \xi : \mathsf{open}\ \chi$ *then* ${}_{\mathsf{Grps}}\mathsf{open}(group(\chi)).\Delta'(\xi) \leqslant \Delta(\xi)$*;*

- *if* $\ell = \xi : \mathsf{comm}\ \sigma$ *then* $\exists \sigma' \leqslant \sigma$*:* $\mathsf{put}(\sigma').\mathsf{get}(\sigma).\Delta'(\xi) \leqslant \Delta(\xi)$.

Here the function $dom(\ell)$ *is given by stipulating* $dom(\epsilon) = \emptyset$*;* $dom(\xi : \mathsf{enter}\ \chi) = dom(\xi : \mathsf{exit}\ \chi) = dom(\xi : \mathsf{open}\ \chi) = dom(\xi : \mathsf{comm}\ \sigma) = \{\xi\}$.

For example, if $\ell = \xi : \mathsf{enter}\ \chi$ and $\Delta(\xi) = {}^{A}\mathsf{enter}(B).\,{}^{B}\mathsf{exit}(A)$, then $\Delta \xrightarrow{\ell} \Delta'$ holds if $group(\chi) = B$ (the ambient tagged ξ is in fact steered into an ambient with group B) and $\Delta' = \Delta[\xi \mapsto {}^{B}\mathsf{exit}(A)]$ (the remaining action).

Theorem 3 (Subject reduction) *Suppose that*

$$P \xrightarrow{\ell} Q$$

where P *and* Q *are uniquely tagged, and suppose that*

$$\Delta, E \vdash_g P : b$$

where $\mathrm{dom}(\Delta)$ *contains no tags occurring in* Q *but not in* P*. Then there exists* Δ' *such that*

$$\Delta \xrightarrow{\ell} \Delta'$$

$$\Delta', E \vdash_g Q : b$$

*and additionally (***Safety of opening***): if* $\ell = \xi : \mathsf{open}\ \chi$ *then* $\mathcal{O}(group(\chi), group(\xi))$.

The proof can be found in Appendix D.

9 Type Checking

Given a complete type derivation for some process P, we can check its validity according to the rules from Fig. 3. To see this, first observe that behaviors can be represented as finite automata (cf. the regular types of [Nie93]), with transitions labeled by actions.

Next observe that the relations $\leqslant$ defined in Sect. 5 are decidable, with decision procedures defined mutually recursively; the only non-trivial issues are

1. given a procedure for deciding $\leqslant$ on level i actions, construct a procedure for deciding $\leqslant$ on level i behaviors;
2. given a procedure for deciding $\leqslant$ on level i behaviors, construct a procedure for deciding $\leqslant$ on level i behavior contexts.

We first address 1. Given b_1 and b_2, recognized by ϵ-transition-free automata A_1 and A_2, we must decide whether $b_1 \leqslant b_2$. For that purpose, we construct[4] a "difference automaton" $A_1 \setminus A_2$, and checks whether $A_1 \setminus A_2$ rejects all inputs.

[4]The states of $A_1 \setminus A_2$ are of the form (q_1, Q_2) with q_1 a state in A_1 and Q_2 a set of states in A_2, with $(\iota_1, \{\iota_2\})$ being initial if ι_1 and ι_2 are initial, with (q_1, Q_2) being final if q_1 is final and Q_2 does not contain any final states, and with an a-transition from (q_1, Q_2) to (q_1', Q_2') if A_1 has an a-transition from q_1 to q_1' and Q_2' is the set of states q_2' in A_2 for which there exists $q_2 \in Q_2$ and an a^+ transition from q_2 to q_2' where $a \leqslant a^+$.

We next address 2. Given B_1 and B_2, we must decide whether $B_1 \leqslant B_2$, that is whether $B_1\lfloor b\rfloor \leqslant B_2\lfloor b\rfloor$ for all behaviors b. This might seem infeasible to check, but Prop. 1 comes to our rescue: it is sufficient to check whether $B_1\lfloor \mathsf{test}\rfloor \leqslant B_2\lfloor \mathsf{test}\rfloor$, where test is suitably chosen.

Finally, to check whether $(g^{\uparrow}, b) \overset{g_0}{\rightsquigarrow} (H, br)$ holds, we annotate each state in the automaton for b with the sets of groups that may enclose g_0 at the given point. The construction is rather involved; we refer to [Amt02] for the details.

The decision procedures are thus potentially very expensive for entities of high level, but such entities are probably rare in practice; for instance, in the communication-free ambient calculus, types are of level 1 and all other entities are of nesting 0. Another concern is that the size of the automaton for a behavior is exponential in the number of parallel constructs; in practice, however, there may never be more than a few processes running in parallel. In general, even though we do not yet have much empirical evidence, hope for efficiency in practice is supported by other similar situations. For example, ML type-inference shows an extreme disparity between worst-case performance in theory (exponential time) and actual performance in practice (very fast).

10 Conclusion

We have demonstrated that it is possible to use *type-based* methods to develop a precise analysis of the *classical* ambient calculus, by presenting a *flow-sensitive* type system where *behaviors* summarize the actions of ambients. Behaviors are regular sets of traces; therefore finite automata can be used for *type checking*. We encourage implementing the type checking algorithm and measuring its actual performance, as well as taking steps towards type reconstruction.

Acknowledgments. Most of the research behind this work was carried out in 2001 while the author was at Boston University, supported by NSF EIA grant 9806745/9806746/9806747/9806835 and partially supported by Sun grant EDUD-7826-990410-US; also some part was done in 2002 when at Heriot-Watt University, supported by the DART project (EC grant IST-2001-33477). Currently, the author is supported by an AFOSR grant, and by NSF grant CCR-0296182.

The author would like to thank Michele Bugliesi, Assaf Kfoury, and Santiago Pericas-Geertsen for inspiring discussions, and the latter two for commenting upon an early draft of [Amt02].

References

[AKPG01] Torben Amtoft, Assaf J. Kfoury, and Santiago M. Pericas-Geertsen. What are polymorphically-typed ambients? In David Sands, editor, *ESOP 2001, Genova*, volume 2028 of *LNCS*, pages 206–220. Springer-Verlag, April 2001.

[Amt02] Torben Amtoft. Causal type system for ambient movements. Technical Report 2002-04, Department of Computing and Information Sciences, Kansas State University, 2002.

[AMW04] Torben Amtoft, Henning Makholm, and J. B. Wells. PolyA: True type polymorphism for mobile ambients. In J.-J. Levy, E. W. Mayr, and J. C. Mitchell, editors, *TCS 2004 (3rd IFIP International Conference on Theoretical Computer Science), Toulouse, France, August 2004*, pages 591–604. Kluwer Academic Publishers, 2004.

[BCC01] Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Boxed ambients. In *4th International Conference on Theoretical Aspects of Computer Science (TACS'01)*, volume 2215 of *LNCS*, pages 38–63. Springer-Verlag, 2001.

[BPG00] Michele Bugliesi and Santiago Pericas-Geertsen. Depth subtyping and type inference for object calculi. In *Proc. of the 7th Int. Workshop on Foundations of Object Oriented Languages (FOOL'7)*, 2000. Electronic Proceedings: `http://www.cis.upenn.edu/~bcpierce/FOOL//FOOL7.html`.

[CC91] Felice Cardone and Mario Coppo. Type inference with recursive types: Syntax and semantics. *Information and Computation*, 92:48–80, 1991.

[CDC02] M. Coppo and M. Dezani-Ciancaglini. A fully abstract model for higher-order mobile ambients. In *VMCAI'02*, volume 2294 of *LNCS*, pages 255–271. Springer-Verlag, 2002.

[CG98] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In Maurice Nivat, editor, *Foundations of Software Science and Computational Structures (FOSSACS'98)*, volume 1378 of *LNCS*, pages 140–155. Springer-Verlag, 1998.

[CG99] Luca Cardelli and Andrew D. Gordon. Types for mobile ambients. In *POPL'99, San Antonio, Texas*, pages 79–92. ACM Press, 1999.

[CGG99] Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Mobility types for mobile ambients. In Jiri Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP'99*, volume 1644 of *LNCS*, pages 230–239. Springer-Verlag, 1999. Extended version appears as Microsoft Research Technical Report MSR-TR-99-32, 1999.

[CGG00] Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Ambient groups and mobility types. In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000), Tohoku University, Sendai, Japan*, volume 1872 of *LNCS*, pages 333–347. Springer-Verlag, August 2000.

[GH99] Simon Gay and Malcolm Hole. Types and subtypes for client-server interactions. In *ESOP'99*, volume 1576 of *LNCS*, pages 74–90. Springer-Verlag, 1999.

[GYY01] Xudong Guan, Yiling Yang, and Jinyuan You. Typing evolving ambients. *Information Processing Letters*, 80(5):265–270, November 2001.

[HJNN99] Rene Rydhof Hansen, Jacob Grydholt Jensen, Flemming Nielson, and Hanne Riis Nielson. Abstract interpretation of mobile ambients. In *SAS'99*, volume 1694 of *LNCS*, pages 134–148. Springer-Verlag, 1999.

[LM04] Francesca Levi and Sergio Maffeis. On abstract interpretation of mobile ambients. *Information and Computation*, 188(2):179–240, January 2004.

[LS00] Francesca Levi and Davide Sangiorgi. Controlling interference in ambients. In *POPL'00, Boston, Massachusetts*, pages 352–364. ACM Press, 2000.

[MS02] Massimo Merro and Vladimiro Sassone. Typing and subtyping mobility in boxed ambients. In *CONCUR'02*, volume 2421 of *LNCS*, pages 304–320. Springer-Verlag, 2002.

[MW05] Henning Makholm and J.B. Wells. Instant polymorphic type systems for mobile process calculi: Just add reduction rules and close. In Mooly Sagiv, editor, *ESOP 2005, Edinburgh*, volume 3444 of *LNCS*, pages 389–407. Springer-Verlag, April 2005.

[Nie93] Oscar Nierstrasz. Regular types for active objects. In *OOPSLA '93*, volume 28 of *ACM SIGPLAN Notices*, pages 1–15, 1993.

[NN01] Hanne Riis Nielson and Flemming Nielson. Shape analysis for mobile ambients. *Nordic Journal of Computing*, 8:233–275, 2001. A preliminary version appeared at POPL'00.

[NNHJ99] Flemming Nielson, Hanne Riis Nielson, Rene Rydhof Hansen, and Jacob Grydholt Jensen. Validating firewalls in mobile ambients. In *Proc. CONCUR'99*, volume 1664 of *LNCS*, pages 463–477. Springer-Verlag, 1999.

[NNP04] Hanne Riis Nielson, Flemming Nielson, and Henrik Pilegaard. Spatial analysis of BioAmbients. In R. Giacobazzi, editor, *SAS 2004 (11th Static Analysis Symposium)*, volume 3148 of *LNCS*, pages 69–83. Springer-Verlag, 2004.

[NNS00] Flemming Nielson, Hanne Riis Nielson, and Mooly Sagiv. A Kleene analysis of mobile ambients. In *ESOP 2000, Berlin*, volume 1782 of *LNCS*, pages 305–319. Springer-Verlag, 2000.

$P \equiv P$	(Struct Refl)
If $P \equiv Q$ then $Q \equiv P$	(Struct Symm)
If $P \equiv Q$ and $Q \equiv R$ then $P \equiv R$	(Struct Trans)
If $P \equiv Q$ then $(\nu n : \tau).P \equiv (\nu n : \tau).Q$	(Struct Res)
If $P \equiv Q$ then $P \mid R \equiv Q \mid R$	(Struct Par)
If $P \equiv Q$ then $!P \equiv !Q$	(Struct Repl)
If $P \equiv Q$ then $M[\![P]\!]^{\xi} \equiv M[\![Q]\!]^{\xi}$	(Struct Amb)
If $P \equiv Q$ then $M.P \equiv M.Q$	(Struct Action)
If $P \equiv Q$ then $(n_1 \ldots n_k : \tau_1 \ldots \tau_k).P \equiv (n_1 \ldots n_k : \tau_1 \ldots \tau_k).Q$	(Struct Input)
$P \mid Q \equiv Q \mid P$	(Struct ParComm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct ParAssoc)
$(\nu n_1 : \tau_1).(\nu n_2 : \tau_2).P \equiv (\nu n_2 : \tau_2).(\nu n_1 : \tau_1).P$ if $n_1 \neq n_2$	(Struct ResRes)
$(\nu n : \tau).(P \mid Q) \equiv P \mid (\nu n : \tau).Q$ if $n \notin \mathsf{fn}(P)$	(Struct ResPar)
$(\nu n : \tau).m[\![P]\!]^{\xi} \equiv m[\![(\nu n : \tau).P]\!]^{\xi}$ if $n \neq m$	(Struct ResAmb)
$P \mid \mathbf{0} \equiv P$	(Struct ZeroPar)
$(\nu n : \tau).\mathbf{0} \equiv \mathbf{0}$	(Struct ZeroRes)
$!\mathbf{0} \equiv \mathbf{0}$	(Struct ZeroRepl)
$\epsilon.P \equiv P$	(Struct ε)
$(M.M').P \equiv M.(M'.P)$	(Struct .)
$P \equiv Q$ if P equals Q modulo consistent renaming of bound names	(Struct α-rename)

Figure 4: Structural Congruence.

[RV97] António Ravara and Vasco T. Vasconcelos. Behavioural types for a calculus of concurrent objects. In *Euro-Par'97*, volume 1300 of *LNCS*, pages 554–561. Springer-Verlag, 1997. Extended version in Technical Report 97-6, Dep. of Mathematics, Technical University of Lisbon, 1997.

[Yos96] Nobuko Yoshida. Graph types for monadic mobile processes. In *16th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'96)*, volume 1180 of *LNCS*, pages 371–386. Springer-Verlag, 1996.

[Zim00] Pascal Zimmer. Subtyping and typing algorithms for mobile ambients. In *FOSSACS 2000, Berlin*, volume 1784 of *LNCS*, pages 375–390. Springer-Verlag, 2000.

A Process Congruence

We define the relation $\equiv$ as the least one satisfying the clauses presented in Fig. 4; here $\mathsf{fn}(P)$ denotes the set of names occurring free in P.

B Basic Properties

Lemma 4 *Given behavior contexts B_1, B_2, and behavior b. Then $B_1\lfloor B_2\lfloor b\rfloor\rfloor = (B_1\lfloor B_2\rfloor)\lfloor b\rfloor$.*

Lemma 5 *If $b_1 \leqslant b_2$ then for all behavior contexts B it holds that $B\lfloor b_1\rfloor \leqslant B\lfloor b_2\rfloor$.*

Lemma 6 *Given action a and behaviors b_1 and b_2. Then $a.(b_1 \mid b_2) \leqslant (a.b_1) \mid b_2$.*

Lemma 7 *$Dest(H, tr)$ is monotone in H as well as in tr.*

Lemma 8 *If $(H_0, b) \overset{g}{\rightsquigarrow} (H, \{g_i : b_i\}_{i\in I})$ then $H_0 \subseteq H$, and for all $i \in I$ with $g_i \in H_0$ we have $b \leqslant b_i$.*

Proof: If $H_0 = \emptyset$, the claim is trivial. Otherwise, $\bullet$ is feasible from H_0, with $Dest(H_0, \bullet) = H_0$. Let tr_2 be a trace in b (by our non-emptiness condition, such a trace exists). Thus the conditions mentioned in Definition 2 are satisfied (with $\bullet$ playing the role of tr_1), and we infer that $H_0 \subseteq H$ and that $g_i \in H_0$ implies $\{tr_2\} \leqslant b_i$. As tr_2 was an arbitrary member of b, this shows $b \leqslant b_i$ as desired. ∎

Lemma 9 *The predicate $(H_0, b) \overset{g}{\rightsquigarrow} (H, br)$ is anti-monotonic in H_0 and b, and monotonic in H and br.*

Proof: Assume that $(H_0, b) \overset{g}{\rightsquigarrow} (H, br)$. Let $H_0^- \subseteq H_0$, $b^- \leqslant b$, $H \subseteq H^+$, and $br \leqslant br^+$; our task is to show $(H_0^-, b^-) \overset{g}{\rightsquigarrow} (H^+, br^+)$.

So let $tr_1^- \diamond tr_2^- \in b^-$ with tr_1^- feasible from H_0^-, and let $H_1^- = Dest(H_0^-, tr_1^-)$.

It is easy to see that there exists $tr_1 \diamond tr_2 \in b$ with $tr_1^- \leqslant tr_1$ and $tr_2^- \leqslant tr_2$. And with $H_1 = Dest(H_0, tr_1)$, we have $H_1^- \subseteq H_1$. We infer that tr_1 is feasible from H_0.

With $br = \{g_i : b_i\}_{i\in I}$, our assumptions thus entail that

1. $H_1 \subseteq H$, and
2. if $tr_2 = \mathsf{put}(\sigma_1)\,\mathsf{get}(\sigma_2) \diamond tr_3$ then $\sigma_1 \leqslant \sigma_2$, and
3. if $tr_2 = {}_G\mathsf{open}(\_) \diamond tr_3$ then $g \in G$, and
4. for all $i \in I$: $g_i \in H_1$ implies $\{tr_2\} \leqslant b_i$.

With $br^+ = \{g'_j : b'_j\}_{j\in J}$, the claim now follows from the following observations:

1. $H_1^- \subseteq H_1 \subseteq H \subseteq H^+$.
2. If tr_2^- takes the form $\mathsf{put}(\sigma_1^-)\,\mathsf{get}(\sigma_2^+) \diamond tr_3^-$, then tr_2 takes the form $\mathsf{put}(\sigma_1)\,\mathsf{get}(\sigma_2) \diamond tr_3$ with $\sigma_1^- \leqslant \sigma_1$ and $\sigma_2 \leqslant \sigma_2^+$ (and $tr_3^- \leqslant tr_3$). Using (2) above we infer that $\sigma_1^- \leqslant \sigma_1 \leqslant \sigma_2 \leqslant \sigma_2^+$.

3. if tr_2^- takes the form ${}_{G^+}\mathsf{open}(\_)\diamond tr_3^-$, then tr_2 takes the form ${}_G\mathsf{open}(\_)\diamond tr_3$ with $G \subseteq G^+$ (and $tr_3^- \leqslant tr_3$). Using (3) above we thus infer that $g \in G$ and therefore $g \in G^+$.

4. If $j \in J$ is such that $g'_j \in H_1^-$, then by the definition of the ordering relation on behavior rows there exists $i \in I$ with $g_i = g'_j$ and $b_i \leqslant b'_j$. Since $g_i \in H_1^- \subseteq H_1$, we from (4) above infer that $\{tr_2\} \leqslant b_i$, and therefore $\{tr_2^-\} \leqslant \{tr_2\} \leqslant b_i \leqslant b'_j$.

(The polarities of $\mathsf{put}(\_)$, $\mathsf{get}(\_)$, and $\_\mathsf{open}(\_)$ have been put to good use.) ∎

Lemma 10 *Let tr be of the form either ${}^-\mathsf{enter}(\_)$, ${}^-\mathsf{exit}(\_)$, $\_\mathsf{open}(\_)$, or $\mathsf{put}(\_)\mathsf{get}(\_)$. Assume that*

$$(H_0, tr.b) \overset{g}{\rightsquigarrow} (H, br)$$

Then with $H_1 = Dest(H_0, tr)$ we also have

$$(H_1, b) \overset{g}{\rightsquigarrow} (H, br).$$

Proof: Let $tr_1 \diamond tr_2$ belong to b, with tr_1 feasible from H_1, and let $H_2 = Dest(H_1, tr_1)$. As $H_2 = Dest(H_0, tr \diamond tr_1)$ we infer (also employing the conditions on the form of tr) that $tr \diamond tr_1$ is feasible from H_0. Since $tr \diamond tr_1 \diamond tr_2 \in tr.b$, the premise of the lemma thus assures that

1. $H_2 \leqslant H$, and

2. if tr_2 takes the form $\mathsf{put}(\sigma_1)\,\mathsf{get}(\sigma_2) \diamond tr_3$ then $\sigma_1 \leqslant \sigma_2$, and

3. if tr_2 takes the form ${}_G\mathsf{open}(\_) \diamond tr_3$ then $g \in G$, and

4. for all $i \in I$: $g_i \in H_2$ implies $\{tr_2\} \leqslant b_i$, where $br = \{g_i : b_i\}_{i \in I}$.

But this is exactly what is needed to establish the conclusion of the lemma. ∎

Definition 4 *We say that Δ agrees with Δ' on X, to be written $\Delta \underset{X}{=} \Delta'$, if $\Delta(\xi) = \Delta'(\xi)$ for all tags ξ occurring in the entity X.*

Lemma 11 *Suppose that $\Delta, E \vdash_g P : b$, and that $\Delta \underset{P}{=} \Delta'$. Then also $\Delta', E \vdash_g P : b$.*

Lemma 12 (Swapping) *Assume that $E_1, n_1 : \sigma_1, n_2 : \sigma_2, E_2 \vdash M : \tau$ with $n_1 \neq n_2$. Then it follows that $E_1, n_2 : \sigma_2, n_1 : \sigma_1, E_2 \vdash M : \tau$, with a derivation of the same shape. Similarly for $\Delta, E_1, n_1 : \sigma_1, n_2 : \sigma_2, E_2 \vdash_g P : b$.*

Proof: By induction on derivations. ∎

Lemma 13 (Weakening) *Assume that $\Delta, E \vdash_g P : b$, and that n is a name not in $\mathsf{names}(P)$. Then for all τ it follows that $\Delta, E, n : \tau \vdash_g P : b$, with a derivation of the same shape. Similarly for a judgment $E \vdash M : \tau'$ (with $n \notin \mathsf{names}(M)$).*

Proof: The proof is by induction on derivations, applying Lemma 12 to deal with the cases (Proc Res) and (Proc Input). ∎

Lemma 14 (Strengthening) *Assume that* $\Delta, E, n : \tau \vdash_g P : b$*, with* n *not in* $\mathsf{names}(P)$*. Then also* $\Delta, E \vdash_g P : b$ *holds, and with a derivation of the same shape. Similarly for* $E, n : \tau \vdash M : \tau'$ *with* $n \notin \mathsf{names}(M)$*.*

Proof: Similar to the proof of Lemma 13. ∎

Lemma 15 (Closed under open) *Suppose that* $\Delta, E \vdash_g P : b$*. If* $\mathcal{O}(g, g')$ *then also* $\Delta, E \vdash_{g'} P : b$*.*

Proof: Structural induction on the derivation, where the only non-trivial case is (Proc Amb). Noting that $g' \in g^{\uparrow}$ and therefore ${g'}^{\uparrow} \subseteq g^{\uparrow}$, Lemma 9 will ensure that the side condition still holds. ∎

Lemma 16 (Substitution) *Assume that* $E, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash M : \tau$ *(with* $n_1 \ldots n_k$ *distinct), and that there exists* $M_1 \ldots M_k$ *such that for all* $i \in \{1 \ldots k\}$ *it holds that* $E \vdash M_i : \tau_i$ *and that* M *is non-conflicting with* $\{n_i\} \cup \mathsf{names}(M_i)$*. Then* $E \vdash M[n_i := M_i] : \tau$*. Similarly, if* $\Delta, E, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash_g P : b$ *and for all* i *it holds that* $E \vdash M_i : \tau_i$ *and that* P *is non-conflicting with* $\{n_i\} \cup \mathsf{names}(M_i)$ *then* $\Delta, E \vdash_g P[n_i := M_i] : b$*.*

Proof: By induction in the size of the typing derivation for M (resp. P). We do a case analysis on the inference rule applied, but only list two of the cases; the remaining follow by straightforward applications of the induction hypothesis, except for (Proc Input) which is handled as (Proc Res) below.

$\dfrac{}{\tau = \tau_i}$ (Exp n). Here M is a name n. If $n = n_i$ for some $i \in \{1 \ldots k\}$, we infer that $\tau = \tau_i$. The claim is then $E \vdash M_i : \tau_i$, which is among our assumptions. If $n \neq n_i$ for all $i \in \{1 \ldots k\}$, we infer that $\tau = E(n)$. But then we have $E \vdash n : \tau$, as desired.

$\dfrac{}{E, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash P : \tau}$ (Proc Res). Here P takes the form $(\nu n : \tau_0).P_0$, and $E, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash P : \tau$ holds because

$$E, n_1 : \tau_1, \cdots, n_k : \tau_k, n : \tau_0 \vdash P_0 : \tau. \tag{6}$$

Note that for all $i \in \{1 \ldots k\}$ it holds (since P is non-conflicting with $\{n_i\} \cup \mathsf{names}(M_i)$) that $n_i \neq n$ and $n \notin \mathsf{names}(M_i)$ and that P_0 is non-conflicting with $\{n_i\} \cup \mathsf{names}(M_i)$. We can thus apply Lemma 13 to infer that $E, n : \tau_0 \vdash M_i : \tau_i$, and (repeatedly) apply Lemma 12 to infer that

$$E, n : \tau_0, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash P_0 : \tau \tag{7}$$

by a derivation of the same shape as the one for (6). We can thus apply the induction hypothesis on (7) to infer that $E, n : \tau_0 \vdash P_0[n_i := M_i] : \tau$. But then an application of (Proc Res) yields $E \vdash (\nu n : \tau_0).P_0[n_i := M_i] : \tau$, which is as desired since $P[n_i := M_i] = (\nu n : \tau_0).P_0[n_i := M_i]$. ∎

Lemma 17 (Reduction of subprocess) *Assume that*

$$\Delta, E \vdash_g \mathcal{PC}\lfloor P \rfloor : b.$$

Then there exists E_0, g_0, *and* b_0 *such that*

$$\Delta, E_0 \vdash_{g_0} P : b_0$$

and with the property that if there also exists a judgment

$$\Delta', E_0 \vdash_{g_0} Q : b_0$$

where $\Delta \underset{\overline{\mathcal{PC}}}{=} \Delta'$, *then also*

$$\Delta', E \vdash_g \mathcal{PC}\lfloor Q \rfloor : b.$$

Proof: The proof is by induction in $\mathcal{PC}$. We do a case analysis on $\mathcal{PC}$, where only four cases are possible:

$\underline{\mathcal{PC} = \Box}$. Choose $E_0 = E$, $g_0 = g$, and $b_0 = b$. Then the claim clearly holds.

$\underline{\mathcal{PC} = \mathcal{PC}_0 \mid R}$. We assume $\Delta, E \vdash_g \mathcal{PC}_0\lfloor P \rfloor \mid R : b$, so there exists b_1 and b_2 with $b_1 \mid b_2 \leqslant b$ such that

$$\Delta, E \vdash_g \mathcal{PC}_0\lfloor P \rfloor : b_1$$

and $\Delta, E \vdash_g R : b_2$. We can thus apply the induction hypothesis to find E_0, g_0, and b_0 such that $\Delta, E_0 \vdash_{g_0} P : b_0$.

Now assume that $\Delta', E_0 \vdash_{g_0} Q : b_0$ where $\Delta \underset{\overline{\mathcal{PC}}}{=} \Delta'$. Thus $\Delta \underset{\overline{\mathcal{PC}_0}}{=} \Delta'$ so we can further apply the induction hypothesis to infer

$$\Delta', E \vdash_g \mathcal{PC}_0\lfloor Q \rfloor : b_1$$

and also $\Delta \underset{\overline{R}}{=} \Delta'$ so we can apply Lemma 11 to infer $\Delta', E \vdash_g R : b_2$. But this enables us to arrive at the judgment

$$\Delta', E \vdash_g \mathcal{PC}_0\lfloor Q \rfloor \mid R : b$$

which is as desired since $\mathcal{PC}\lfloor Q \rfloor = \mathcal{PC}_0\lfloor Q \rfloor \mid R$.

$\underline{\mathcal{PC} = (\nu n : \tau).\mathcal{PC}_0}$. We assume $\Delta, E \vdash_g (\nu n : \tau).\mathcal{PC}_0\lfloor P \rfloor : b$, so we also have

$$\Delta, E, n : \tau \vdash_g \mathcal{PC}_0\lfloor P \rfloor : b.$$

Inductively there thus exists E_0, g_0, and b_0 such that $\Delta, E_0 \vdash_{g_0} P : b_0$.

Now assume that $\Delta', E_0 \vdash_{g_0} Q : b_0$ where $\Delta \underset{\overline{\mathcal{PC}}}{=} \Delta'$. Thus $\Delta \underset{\overline{\mathcal{PC}_0}}{=} \Delta'$ so a further application of the induction hypothesis tells us that

$$\Delta', E, n : \tau \vdash_g \mathcal{PC}_0\lfloor Q \rfloor : b$$

from which we arrive at $\Delta', E \vdash_g (\nu n : \tau).\mathcal{PC}_0\lfloor Q \rfloor : b$ which is as desired, since $\mathcal{PC}\lfloor Q \rfloor = (\nu n : \tau).\mathcal{PC}_0\lfloor Q \rfloor$.

$\underline{\mathcal{PC} = n[\![\mathcal{PC}_0]\!]^{\xi}}$. Our assumption is that $\Delta, E \vdash_g n[\![\mathcal{PC}_0\lfloor P \rfloor]\!]^{\xi} : b$ from which we infer that $\varepsilon \leqslant b$ and that there exists judgments $E \vdash n : \mathsf{amb}_H^{g_n}[br]$ and

$$\Delta, E \vdash_{g_n} \mathcal{PC}_0\lfloor P \rfloor : b_n$$

where

$$\Delta(\xi) = b_n \text{ and } group(\xi) = g_n \text{ and } (g^{\uparrow}, b_n) \overset{g_n}{\rightsquigarrow} (H, br)$$

Inductively there exists E_0, g_0, and b_0 such that $\Delta, E_0 \vdash_{g_0} P : b_0$.

Now assume that $\Delta', E_0 \vdash_{g_0} Q : b_0$ where $\Delta \overline{\overline{\mathcal{PC}}} \Delta'$. Thus $\Delta \overline{\overline{\mathcal{PC}_0}} \Delta'$ so we can further apply the induction hypothesis to infer

$$\Delta', E \vdash_{g_n} \mathcal{PC}_0\lfloor Q \rfloor : b_n$$

and also $\Delta(\xi) = \Delta'(\xi)$ so we have $\Delta'(\xi) = b_n$. This enables us to arrive at

$$\Delta', E \vdash_{g} n[\![\mathcal{PC}_0\lfloor Q \rfloor]\!]^{\xi} : b$$

which is as desired since $\mathcal{PC}\lfloor Q \rfloor = n[\![\mathcal{PC}_0\lfloor Q \rfloor]\!]^{\xi}$. ∎

C Testing Behavior Contexts

Definition 5 *Given a behavior context B. We say that an action a tests B if a is incomparable with all actions occurring as part of B. To be precise:*

- *a tests $\Box$*
- *a tests $a'.B$ provided a tests B and neither $a \leqslant a'$ nor $a' \leqslant a$ does hold*
- *a tests $b \mid B$ provided a tests B and for all $tr \in b$ and all a' occurring in tr neither $a \leqslant a'$ nor $a' \leqslant a$ does hold.*

Lemma 18 *Given behavior context B, and an action* test *such that* test *tests B. For all behaviors b and traces tr, $tr \in B\lfloor b \rfloor$ holds if and only if there exists tr_1, tr_2, tr_0 such that $tr_1 \diamond \mathsf{test} \diamond tr_2 \in B\lfloor\{\mathsf{test}\}\rfloor$ and $tr_0 \in \{tr_2\} \mid b$ and $tr = tr_1 \diamond tr_0$.*

Proof: The proof is by induction in B, where we perform a case analysis on the form of B.

$\underline{B = \Box}$. For "if", we from $tr_1 \diamond \mathsf{test} \diamond tr_2 \in \{\mathsf{test}\}$ infer that $tr_1 = tr_2 = \bullet$ and therefore $tr = tr_0 \in b$, as desired.

For "only if", we can pick $tr_1 = tr_2 = \bullet$ and $tr_0 = tr$.

$\underline{B = a.B'.}$ For "if", we infer from $tr_1 \diamond \mathsf{test} \diamond tr_2 \in a.B'\lfloor\{\mathsf{test}\}\rfloor$ that (since $a \neq \mathsf{test}$) we can write $tr_1 = a \diamond tr_1'$ with $tr_1' \diamond \mathsf{test} \diamond tr_2 \in B'\lfloor\{\mathsf{test}\}\rfloor$. We can therefore write $tr = a \diamond tr'$ with $tr' = tr_1' \diamond tr_0$. The induction hypothesis therefore tells us that $tr' \in B'\lfloor b \rfloor$, showing that $tr \in B\lfloor b \rfloor$ as desired.

For "only if", we see that we can write $tr = a \diamond tr'$ with $tr' \in B'\lfloor b \rfloor$. By applying the induction hypothesis, we find $tr_1' \diamond \mathsf{test} \diamond tr_2 \in B'\lfloor\{\mathsf{test}\}\rfloor$ and $tr_0 \in \{tr_2\} \mid b$ such that $tr' = tr_1' \diamond tr_0$. With $tr_1 = a \diamond tr_1'$, we therefore get $tr = tr_1 \diamond tr_0$ and $tr_1 \diamond \mathsf{test} \diamond tr_2 \in B\lfloor\{\mathsf{test}\}\rfloor$, as desired.

$\underline{B = b' \mid B'}$. For "if", we infer from $tr_1 \diamond \mathsf{test} \diamond tr_2 \in b' \mid B'\lfloor\{\mathsf{test}\}\rfloor$ that (since test does not occur in b') there exists tr_1', tr_2', tr_1'', tr_2'' such that $tr_1 \in tr_1' \parallel tr_1''$ and $tr_2 \in tr_2' \parallel tr_2''$ and $tr_1' \diamond tr_2' \in b'$ and $tr_1'' \diamond \mathsf{test} \diamond tr_2''$ belongs to $B'\lfloor\mathsf{test}\rfloor$. By associativity of interleaving, we infer from $tr_0 \in \{tr_2\} \mid b$ that there exists $tr_0' \in \{tr_2''\} \mid b$ such that $tr_0 \in tr_2' \parallel tr_0'$. Using the induction hypothesis on B', we infer that $tr_1'' \diamond tr_0' \in B'\lfloor b\rfloor$. But then clearly $tr_1 \diamond tr_0 \in b' \mid B'\lfloor b\rfloor = B\lfloor b\rfloor$, as desired.

For "only if", our assumptions entail that there exists $tr' \in b'$ and $tr'' \in B'\lfloor b\rfloor$ such that $tr \in tr' \parallel tr''$. By applying the induction hypothesis, we find $tr_1'' \diamond \mathsf{test} \diamond tr_2'' \in B'\lfloor\mathsf{test}\rfloor$ such that $tr'' = tr_1'' \diamond tr_0''$ with $tr_0'' \in \{tr_2''\} \mid b$. Since $tr \in tr' \parallel (tr_1'' \diamond tr_0'')$ there clearly exists tr_1', tr_1, tr_0', tr_0 such that $tr' = tr_1' \diamond tr_0'$, $tr = tr_1 \diamond tr_0$, $tr_1 \in tr_1' \parallel tr_1''$, $tr_0 \in tr_0' \parallel tr_0''$. From associativity of interleaving, we infer that there exists $tr_2 \in tr_0' \parallel tr_2''$ such that $tr_0 \in \{tr_2\} \mid b$. This establishes the desired relation $tr_1 \diamond \mathsf{test} \diamond tr_2 \in (tr_1' \diamond tr_0') \parallel (tr_1'' \diamond \mathsf{test} \diamond tr_2'') \subseteq b' \mid B'\lfloor test\rfloor = B\lfloor\mathsf{test}\rfloor$. ∎

C.1 Proof of Proposition 1

Proof: First construct an action test such that test tests B_1 and B_2. (To see that this is possible, observe that a behavior b, being a regular set of traces, can contain only a finite number of different actions. Similarly for a behavior context B. Then choose, for instance, $\mathsf{test} = {}^{\emptyset}\mathsf{enter}(g^*)$ where g^* is a group not occurring in any of these actions.)

We now show that (c) implies (b), the only non-trivial part of the lemma.

For this purpose, assume that b has been given. Let tr belong to $B_1\lfloor b\rfloor$, then our task is to find $tr^+ \in B_2\lfloor b\rfloor$ with $tr^+ \geqslant tr$. By Lemma 18, there exists tr_1, tr_2, tr_0 such that $tr_1 \diamond \mathsf{test} \diamond tr_2 \in B_1\lfloor\mathsf{test}\rfloor$ and $tr_0 \in \{tr_2\} \mid b$ and $tr = tr_1 \diamond tr_0$. Using our assumption (c), there exists a trace $tr' \in B_2\lfloor\mathsf{test}\rfloor$ with $tr_1 \diamond \mathsf{test} \diamond tr_2 \leqslant tr'$. Since test occurs in all traces in $B_2\lfloor\mathsf{test}\rfloor$, and since test is incomparable with all actions in tr_1 and in tr_2, we can write $tr' = tr_1^+ \diamond \mathsf{test} \diamond tr_2^+$ with $tr_1 \leqslant tr_1^+$ and $tr_2 \leqslant tr_2^+$. Clearly, there exists $tr_0^+ \in \{tr_2^+\} \mid b$ with $tr_0 \leqslant tr_0^+$. Define $tr^+ = tr_1^+ \diamond tr_0^+$. By Lemma 18 we infer that $tr^+ \in B_2\lfloor b\rfloor$. As clearly $tr \leqslant tr^+$, this is as desired. ∎

D Subject Reduction

Lemma 19 *If $P \equiv Q$ then P and Q contain the same tags, and P is uniquely tagged iff Q is.*

Lemma 20 *If $P \xrightarrow{\ell} Q$ then all tags in $dom(\ell)$ occur in P.*

Lemma 21 (Subject congruence) *Assume that $P \equiv Q$.*
Then $\Delta, E \vdash_g P : b$ if and only if $\Delta, E \vdash_g Q : b$.

Proof: The proof is by induction on the derivation of $P \equiv Q$, much as the similar result in [CG99]. Below we list the most interesting cases.

(Struct ParAssoc). Assume that $\Delta, E \vdash_g (P \mid Q) \mid R : b$ (the other direction is similar). Then there exists b_{12} and b_3 with $b_{12} \mid b_3 \leqslant b$ such that $\Delta, E \vdash_g P \mid Q : b_{12}$ and $\Delta, E \vdash_g R : b_3$. Therefore there exists b_1 and b_2 with $b_1 \mid b_2 \leqslant b_{12}$ such that $\Delta, E \vdash_g P : b_1$ and $\Delta, E \vdash_g Q : b_2$. Now $\Delta, E \vdash_g P \mid (Q \mid R) : b_1 \mid (b_2 \mid b_3)$, and by Lemma 5 we infer that $b_1 \mid (b_2 \mid b_3) \leqslant (b_1 \mid b_2) \mid b_3 \leqslant b_{12} \mid b_3 \leqslant b$. This implies the desired judgment $\Delta, E \vdash_g P \mid (Q \mid R) : b$.

(Struct ResRes). Follows easily from Lemma 12.

(Struct ResPar). Assume that $\Delta, E \vdash_g (\nu n : \tau).(P \mid Q) : b$. There exists b_1 and b_2 with $b_1 \mid b_2 \leqslant b$ such that $\Delta, E, n : \tau \vdash_g P : b_1$ and $\Delta, E, n : \tau \vdash_g Q : b_2$. Since $n \notin \mathsf{fn}(P)$, we can α-rename P into a P' such that $n \notin \mathsf{names}(P')$. Clearly $\Delta, E, n : \tau \vdash_g P' : b_1$ (this claim amounts to the case (Struct α-rename)), so by Lemma 14 we have $\Delta, E \vdash_g P' : b_1$ and therefore also $\Delta, E \vdash_g P : b_1$. We can thus infer first $\Delta, E \vdash_g (\nu n : \tau).Q : b_2$ and then (since $b_1 \mid b_2 \leqslant b$) the desired judgment $\Delta, E \vdash_g P \mid (\nu n : \tau).Q : b$.

For the other direction, assume that $\Delta, E \vdash_g P \mid (\nu n : \tau).Q : b$. There exists b_1 and b_2 with $b_1 \mid b_2 \leqslant b$ such that $\Delta, E \vdash_g P : b_1$ and $\Delta, E, n : \tau \vdash_g Q : b_2$. Since $n \notin \mathsf{fn}(P)$, we can α-rename P into a P' such that $n \notin \mathsf{names}(P')$. Then $\Delta, E \vdash_g P' : b_1$ so by Lemma 13 we have $\Delta, E, n : \tau \vdash_g P' : b_1$ and therefore also $\Delta, E, n : \tau \vdash_g P : b_1$. We can thus infer first $\Delta, E, n : \tau \vdash_g P \mid Q : b_1 \mid b_2$ and then the desired judgment $\Delta, E \vdash_g (\nu n : \tau).(P \mid Q) : b$.

(Struct ResAmb). Let $E(m) = (E, n : \tau)(m) = \mathsf{amb}_H^{g_0}[br]$. Assume that $\Delta, E \vdash_g (\nu n : \tau).m[\![P]\!]^{\xi} : b$ (the other direction is similar). We infer that $\varepsilon \leqslant b$, and that there exists b_0 such that $\Delta, E, n : \tau \vdash_{g_0} P : b_0$ and $(g^{\uparrow}, b_0) \overset{g_0}{\rightsquigarrow} (H, br)$ with $group(\xi) = g_0$ and $\Delta(\xi) = b_0$. But then $\Delta, E \vdash_{g_0} (\nu n : \tau).P : b_0$, clearly enabling us to infer the desired judgment $\Delta, E \vdash_g m[\![(\nu n : \tau).P]\!]^{\xi} : b$.

(Struct ZeroPar). Assume that $\Delta, E \vdash_g P \mid \mathbf{0} : b$. There exists b_1 and b_2 with $b_1 \mid b_2 \leqslant b$ such that $\Delta, E \vdash_g P : b_1$ and $\Delta, E \vdash_g \mathbf{0} : b_2$. We infer that $\varepsilon \leqslant b_2$ and by Lemma 5 therefore $b_1 \leqslant b_1 \mid \varepsilon \leqslant b_1 \mid b_2 \leqslant b$. This implies the desired judgment $\Delta, E \vdash_g P : b$.

The other direction is trivial.

(Struct ZeroRepl). First assume that $\Delta, E \vdash_g !\mathbf{0} : b$. We infer that there exists $b_0 \leqslant b$ with $b_0 \mid b_0 \leqslant b_0$ such that $\Delta, E \vdash_g \mathbf{0} : b_0$, from which we further infer that $\varepsilon \leqslant b_0$. But then also $\Delta, E \vdash_g \mathbf{0} : b$, as desired, since $\varepsilon \leqslant \varepsilon \mid \varepsilon \leqslant b_0 \mid b_0 \leqslant b_0 \leqslant b$.

Conversely, assume that $\Delta, E \vdash_g \mathbf{0} : b$ from which we infer that $\varepsilon \leqslant b$. Since $\varepsilon \mid \varepsilon \leqslant \varepsilon$ we can apply (Proc Zero) and (Proc Repl) to derive $\Delta, E \vdash_g !\mathbf{0} : \varepsilon$, and therefore also the desired $\Delta, E \vdash_g !\mathbf{0} : b$.

(Struct ε). Assume that $\Delta, E \vdash_g P : b$. Since $\Delta, E \vdash_g \epsilon : \mathsf{cap}[\Box]$, we can by (Proc Action) infer $\Delta, E \vdash_g \epsilon.P : b$.

Conversely, assume that $\Delta, E \vdash_g \epsilon.P : b$. Then there exists B and b_0 with $B\lfloor b_0 \rfloor \leqslant b$ such that $\Delta, E \vdash_g \epsilon : \mathsf{cap}[B]$ and $\Delta, E \vdash_g P : b_0$. As $\mathsf{cap}[\Box] \leqslant \mathsf{cap}[B]$ we deduce that $\Box \leqslant B$, implying (by Prop. 1) that $b_0 = \Box\lfloor b_0 \rfloor \leqslant$

$B\lfloor b_0 \rfloor \leqslant b$. Thus we can derive the desired judgment $\Delta, E \vdash_g P : b$.

(Struct .). Assume that $\Delta, E \vdash_g (M_1.M_2).P : b$. There exists B_0 and b_0 with $B_0\lfloor b_0 \rfloor \leqslant b$ such that $E \vdash M_1.M_2 : \mathsf{cap}[B_0]$ and $\Delta, E \vdash_g P : b_0$. There thus exists B_1 and B_2 with $\mathsf{cap}[B_1\lfloor B_2 \rfloor] \leqslant \mathsf{cap}[B_0]$, implying $B_1\lfloor B_2 \rfloor \leqslant B_0$, such that $E \vdash M_1 : \mathsf{cap}[B_1]$ and $E \vdash M_2 : \mathsf{cap}[B_2]$. We can thus infer first $\Delta, E \vdash_g M_2.P : B_2\lfloor b_0 \rfloor$ and then $\Delta, E \vdash_g M_1.(M_2.P) : B_1\lfloor B_2\lfloor b_0 \rfloor\rfloor$. But by employing Lemma 4 and Prop. 1 we deduce that $B_1\lfloor B_2\lfloor b_0 \rfloor\rfloor = (B_1\lfloor B_2 \rfloor)\lfloor b_0 \rfloor \leqslant B_0\lfloor b_0 \rfloor \leqslant b$. This shows that we can derive the desired judgment $\Delta, E \vdash_g M_1.(M_2.P) : b$.

Conversely, assume that $\Delta, E \vdash_g M_1.(M_2.P) : b$. There exists B_1 and b_1 with $B_1\lfloor b_1 \rfloor \leqslant b$ such that $E \vdash M_1 : \mathsf{cap}[B_1]$ and $\Delta, E \vdash_g M_2.P : b_1$, and therefore there exists B_2 and b_0 with $B_2\lfloor b_0 \rfloor \leqslant b_1$ such that $E \vdash M_2 : \mathsf{cap}[B_2]$ and $\Delta, E \vdash_g P : b_0$. We can thus infer first $E \vdash M_1.M_2 : \mathsf{cap}[B_1\lfloor B_2 \rfloor]$ and then $\Delta, E \vdash_g (M_1.M_2).P : (B_1\lfloor B_2 \rfloor)\lfloor b_0 \rfloor$. But by employing Lemmas 4 and 5, we deduce that $(B_1\lfloor B_2 \rfloor)\lfloor b_0 \rfloor = B_1\lfloor B_2\lfloor b_0 \rfloor\rfloor \leqslant B_1\lfloor b_1 \rfloor \leqslant b$. This shows that we can derive the desired judgment $\Delta, E \vdash_g (M_1.M_2).P : b$. ∎

D.1 Proof of Theorem 3

Proof: The proof is by induction on the derivation of $P \xrightarrow{\ell} Q$, with a case analysis on the last rule used. In the type derivations show below, we shall "inline" the applications of the subtyping rules (Proc Subsumption) and (Exp Subsumption).

(Red In). The situation is that

$$m[\![\mathsf{in}\ n.P \mid Q]\!]^{\xi} \mid n[\![R]\!]^{\chi} \xrightarrow{\xi:\mathsf{enter}\ \chi} n[\![m[\![P \mid Q]\!]^{\xi} \mid R]\!]^{\chi}$$

With $\Delta(\xi) = b_m$ and $\Delta(\chi) = b_n$, and with $group(\xi) = g_m$ and $group(\chi) = g_n$ (note that $\xi \neq \chi$ by assumption on unique tags), the typing of the left hand side takes the form

$$\dfrac{\dfrac{E \vdash m : \mathsf{amb}^{g_m}_{H_m}[br_m] \qquad \dfrac{\dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H'_n}[br'_n]}{E \vdash \mathsf{in}\ n : \mathsf{cap}[B]} \quad \Delta, E \vdash_{g_m} P : b_0}{\Delta, E \vdash_{g_m} \mathsf{in}\ n.P : b_1} \qquad \Delta, E \vdash_{g_m} Q : b_2}{\Delta, E \vdash_{g_m} \mathsf{in}\ n.P \mid Q : b_m}}{\Delta, E \vdash_g m[\![\mathsf{in}\ n.P \mid Q]\!]^{\xi} : \varepsilon} \qquad \dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H_n}[br_n] \quad \Delta, E \vdash_{g_n} R : b_n}{\Delta, E \vdash_g n[\![R]\!]^{\chi} : \varepsilon}}{\Delta, E \vdash_g m[\![\mathsf{in}\ n.P \mid Q]\!]^{\xi} \mid n[\![R]\!]^{\chi} : b}$$

where we have the relations

$$^{H'_n}\mathsf{enter}(g_n).\Box \leqslant B \qquad B\lfloor b_0 \rfloor \leqslant b_1 \qquad b_1 \mid b_2 \leqslant b_m \qquad \varepsilon \leqslant b$$

$$(g^{\uparrow}, b_m) \overset{g_m}{\rightsquigarrow} (H_m, br_m) \tag{8}$$

$$(g^{\uparrow}, b_n) \overset{g_n}{\rightsquigarrow} (H_n, br_n) \tag{9}$$

We now define

$$\Delta' = \Delta[\xi \mapsto (b_0 \mid b_2)]$$

By Prop. 1, it holds that

$$^{H'_n}\mathsf{enter}(g_n).b_0 \leqslant B\lfloor b_0 \rfloor \leqslant b_1.$$

From (9) we see by Lemma 8 that $g^{\uparrow} \subseteq H_n$, which since $\mathsf{amb}^{-}_{H}[\_]$ is *invariant* in H and therefore $H_n \subseteq H'_n$ amounts to

$$g^{\uparrow} \subseteq H'_n.$$

By Lemma 6 we therefore infer that

$$^{g^{\uparrow}}\mathsf{enter}(g_n).(b_0 \mid b_2) \leqslant (\,^{H'_n}\mathsf{enter}(g_n).b_0) \mid b_2 \leqslant b_1 \mid b_2 \leqslant b_m. \tag{10}$$

By Lemma 9 applied to (8), this shows

$$(g^{\uparrow},\, ^{g^{\uparrow}}\mathsf{enter}(g_n).(b_0 \mid b_2)) \overset{g_m}{\rightsquigarrow} (H_m, br_m)$$

which by Lemma 10 implies that

$$(g_n{}^{\uparrow}, b_0 \mid b_2) \overset{g_m}{\rightsquigarrow} (H_m, br_m). \tag{11}$$

Since the left hand side of the reduction is uniquely tagged, ξ does not occur in either of P, Q, or R. Thus $\Delta \overset{}{\underset{P,Q,R}{=\!=}} \Delta'$, so by Lemma 11 it holds that

$$\Delta', E \vdash_{g_m} P : b_0 \text{ and } \Delta', E \vdash_{g_m} Q : b_2 \text{ and } \Delta', E \vdash_{g_n} R : b_n.$$

Using the above observations, in particular (11), we can indeed construct a typing for the right hand side of the reduction:

$$\dfrac{\dfrac{E \vdash m : \mathsf{amb}^{g_m}_{H_m}[br_m] \qquad \dfrac{\Delta', E \vdash_{g_m} P : b_0 \qquad \Delta', E \vdash_{g_m} Q : b_2}{\Delta', E \vdash_{g_m} P \mid Q : b_0 \mid b_2}}{\Delta', E \vdash_{g_n} m[\![P \mid Q]\!]^{\xi} : \varepsilon} \qquad \Delta', E \vdash_{g_n} R : b_n}{\dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H_n}[br_n] \qquad \Delta', E \vdash_{g_n} m[\![P \mid Q]\!]^{\xi} \mid R : b_n}{\Delta', E \vdash_{g} n[\![m[\![P \mid Q]\!]^{\xi} \mid R]\!]^{\chi} : b}}$$

Moreover, by (10) we also have the desired relation

$$^{\emptyset}\mathsf{enter}(\mathit{group}(\chi)).\Delta'(\xi) = {}^{\emptyset}\mathsf{enter}(g_n).(b_0 \mid b_2) \leqslant b_m = \Delta(\xi).$$

(Red Out). The situation is that

$$n[\![m[\![\mathsf{out}\ n.P \mid Q]\!]^{\xi} \mid R]\!]^{\chi} \overset{\xi:\mathsf{exit}\,\chi}{\longrightarrow} m[\![P \mid Q]\!]^{\xi} \mid n[\![R]\!]^{\chi}$$

With $\Delta(\xi) = b_m$ and $\Delta(\chi) = b_n$, and with $\mathit{group}(\xi) = g_m$ and $\mathit{group}(\chi) = g_n$ (note that $\xi \neq \chi$ by assumption on unique tags), the typing of the left hand side takes the form

$$\dfrac{\dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H'_n}[br'_n]}{E \vdash \mathsf{out}\ n : \mathsf{cap}[B] \quad \Delta, E \vdash_{g_m} P : b_0} \qquad \dfrac{\Delta, E \vdash_{g_m} \mathsf{out}\ n.P : b_1 \qquad \Delta, E \vdash_{g_m} Q : b_2}{\Delta, E \vdash_{g_m} \mathsf{out}\ n.P \mid Q : b_m}}{\Delta, E \vdash_{g_n} m[\mathsf{out}\ n.P \mid Q]^{\xi} : \varepsilon} \qquad \Delta, E \vdash_{g_n} R : b_n$$

$$\dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H_n}[br_n] \qquad \Delta, E \vdash_{g_n} m[\mathsf{out}\ n.P \mid Q]^{\xi} \mid R : b_n}{\Delta, E \vdash_{g} n[m[\mathsf{out}\ n.P \mid Q]^{\xi} \mid R]^{\chi} : b}$$

where we have the relations

$$^{g_n}\mathsf{exit}(H'_n).\square \leqslant B \qquad B\lfloor b_0 \rfloor \leqslant b_1 \qquad b_1 \mid b_2 \leqslant b_m \qquad \varepsilon \leqslant b$$

$$(g_n{}^{\uparrow}, b_m) \overset{g_m}{\rightsquigarrow} (H_m, br_m) \tag{12}$$

$$(g^{\uparrow}, b_n) \overset{g_n}{\rightsquigarrow} (H_n, br_n) \tag{13}$$

We now define

$$\Delta' = \Delta[\xi \mapsto (b_0 \mid b_2)]$$

By Prop. 1, it holds that

$$^{g_n}\mathsf{exit}(H'_n).b_0 \leqslant B\lfloor b_0 \rfloor \leqslant b_1.$$

From (13) we see by Lemma 8 that $g^{\uparrow} \subseteq H_n$, which since $\mathsf{amb}^{-}_{H}[\_]$ is *invariant* in H and therefore $H_n \subseteq H'_n$ amounts to

$$g^{\uparrow} \subseteq H'_n.$$

By Lemma 6 we therefore infer that

$$^{g_n}\mathsf{exit}(g^{\uparrow}).(b_0 \mid b_2) \leqslant (\,^{g_n}\mathsf{exit}(H'_n).b_0) \mid b_2 \leqslant b_1 \mid b_2 \leqslant b_m. \tag{14}$$

By Lemma 9 applied to (12), this shows

$$(g_n{}^{\uparrow},\ ^{g_n}\mathsf{exit}(g^{\uparrow}).(b_0 \mid b_2)) \overset{g_m}{\rightsquigarrow} (H_m, br_m)$$

which by Lemma 10 implies that

$$(g^{\uparrow}, b_0 \mid b_2) \overset{g_m}{\rightsquigarrow} (H_m, br_m). \tag{15}$$

Since the left hand side of the reduction is uniquely tagged, ξ does not occur in neither of P, Q, or R. Therefore (by Lemma 11) it holds that

$$\Delta', E \vdash_{g_m} P : b_0 \text{ and } \Delta', E \vdash_{g_m} Q : b_2 \text{ and } \Delta', E \vdash_{g_n} R : b_n$$

Using the above observations, in particular (15), we can indeed construct a typing for the right hand side of the reduction:

$$\dfrac{E \vdash m : \mathsf{amb}^{g_m}_{H_m}[br_m] \qquad \dfrac{\Delta', E \vdash_{g_m} P : b_0 \qquad \Delta', E \vdash_{g_m} Q : b_2}{\Delta', E \vdash_{g_m} P \mid Q : b_0 \mid b_2}}{\Delta', E \vdash_g m[\![P \mid Q]\!]^{\xi} : \varepsilon} \qquad \dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H_n}[br_n] \qquad \Delta', E \vdash_{g_n} R : b_n}{\Delta', E \vdash_g n[\![R]\!]^{\chi} : \varepsilon}$$

$$\Delta', E \vdash_g m[\![P \mid Q]\!]^{\xi} \mid n[\![R]\!]^{\chi} : b$$

Moreover, by (14) we also have the desired relation

$${}^{group(\chi)}\mathsf{exit}(\emptyset).\Delta'(\xi) = {}^{g_n}\mathsf{exit}(\emptyset).(b_0 \mid b_2) \leqslant b_m = \Delta(\xi).$$

(Red Open). The situation is that

$$m[\![\mathsf{open}\ n.P \mid n[\![Q]\!]^{\chi} \mid R]\!]^{\xi} \xrightarrow{\xi:\mathsf{open}\ \chi} m[\![P \mid Q \mid R]\!]^{\xi}$$

With $\Delta(\xi) = b_m$ and $\Delta(\chi) = b_n$, and with $group(\xi) = g_m$ and $group(\chi) = g_n$ (note that $\xi \neq \chi$ by assumption on unique tags), the typing of the left hand side takes the form

$$\dfrac{E \vdash m : \mathsf{amb}^{g_m}_{H_m}[br_m] \qquad \dfrac{\dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H'_n}[br'_n]}{E \vdash \mathsf{open}\ n : \mathsf{cap}[B]} \quad \Delta, E \vdash_{g_m} P : b_0}{\Delta, E \vdash_{g_m} \mathsf{open}\ n.P : b_1} \qquad \dfrac{E \vdash n : \mathsf{amb}^{g_n}_{H_n}[br_n] \qquad \Delta, E \vdash_{g_n} Q : b_n}{\Delta, E \vdash_{g_m} n[\![Q]\!]^{\chi} : \varepsilon} \qquad \Delta, E \vdash_{g_m} R : b_2}{\Delta, E \vdash_{g_m} \mathsf{open}\ n.P \mid n[\![Q]\!]^{\chi} \mid R : b_m}$$

$$\Delta, E \vdash_g m[\![\mathsf{open}\ n.P \mid n[\![Q]\!]^{\chi} \mid R]\!]^{\xi} : b$$

where we have the relations

$${}_G\mathsf{open}(g_n).(b' \mid \square) \leqslant B \qquad B\lfloor b_0 \rfloor \leqslant b_1 \qquad b_1 \mid b_2 \leqslant b_m \qquad \varepsilon \leqslant b$$

$$(g_m{}^{\uparrow}, b_n) \overset{g_n}{\rightsquigarrow} (H_n, br_n) \tag{16}$$

$$(g^{\uparrow}, b_m) \overset{g_m}{\rightsquigarrow} (H_m, br_m) \tag{17}$$

With $br'_n = \{g'_i : b'_i\}_{i \in I}$ we also have

$$\forall g' \in G : \exists i \in I : g' = g'_i \text{ and } b'_i \leqslant b' \tag{18}$$

Since $\mathsf{amb}^{\text{-}}_{\text{-}}[br]$ is *invariant* in br, we have $br_n \leqslant br'_n$, so (16) implies

$$(g_m{}^{\uparrow}, b_n) \overset{g_n}{\rightsquigarrow} (H_n, br'_n) \tag{19}$$

We now define

$$\Delta' = \Delta[\xi \mapsto (b_0 \mid b_n \mid b_2)]$$

By Prop. 1, it holds that

$${}_G\mathsf{open}(g_n).(b' \mid b_0) \leqslant B\lfloor b_0 \rfloor \leqslant b_1.$$

By Lemma 6 we therefore infer that

$$ {}_G\mathsf{open}(g_n).(b' \mid b_0 \mid b_2) \leqslant ({}_G\mathsf{open}(g_n).(b' \mid b_0)) \mid b_2 \leqslant b_1 \mid b_2 \leqslant b_m. \tag{20} $$

By Lemma 9 applied to (17), this shows

$$ (g^{\uparrow}, {}_G\mathsf{open}(g_n).(b' \mid b_0 \mid b_2)) \overset{g_m}{\rightsquigarrow} (H_m, br_m) $$

As ${}_G\mathsf{open}(g_n).(b' \mid b_0 \mid b_2)$ contains a trace (we exploit that all behaviors are non-empty) of the form $\bullet \diamond {}_G\mathsf{open}(g_n) \diamond tr_3$ with $\bullet$ feasible from $g^{\uparrow}$, Definition 2 tells us that $g_m \in G$. By (18) there thus exists $i \in I$ such that

$$ g_m = g'_i \text{ and } b'_i \leqslant b'. $$

From this we infer, since $\mathsf{amb}^{g_n}_{H'_n}[br'_n]$ is a (well-formed) type, that

$$ \mathcal{O}(g_n, g_m) \tag{21} $$

and also infer that $g'_i \in {g_m}^{\uparrow}$. Lemma 8 applied to (19) then tells us that $b_n \leqslant b'_i$ and therefore

$$ \Delta'(\xi) = b_0 \mid b_n \mid b_2 \leqslant b' \mid b_0 \mid b_2. $$

From (20) we thus infer

$$ {}_G\mathsf{open}(g_n).\Delta'(\xi) \leqslant b_m \tag{22} $$

which by Lemma 9 applied to (17) shows $(g^{\uparrow}, {}_G\mathsf{open}(g_n).\Delta'(\xi)) \overset{g_m}{\rightsquigarrow} (H_m, br_m)$. By Lemma 10 this implies that

$$ (g^{\uparrow}, \Delta'(\xi)) \overset{g_m}{\rightsquigarrow} (H_m, br_m). \tag{23} $$

Since the left hand side of the reduction is uniquely tagged, ξ does not occur in neither of P, Q, or R. Therefore (by Lemma 11) it holds that

$$ \Delta', E \vdash_{g_m} P : b_0 \text{ and } \Delta', E \vdash_{g_m} R : b_2 $$

as well as $\Delta', E \vdash_{g_n} Q : b_n$ which by Lemma 15 using (21) implies

$$ \Delta', E \vdash_{g_m} Q : b_n. $$

Using the above observations, in particular (23), we can indeed construct a typing for the right hand side of the reduction:

$$ \dfrac{E \vdash m : \mathsf{amb}^{g_m}_{H_m}[br_m] \qquad \dfrac{\Delta', E \vdash_{g_m} P : b_0 \qquad \Delta', E \vdash_{g_m} Q : b_n \qquad \Delta', E \vdash_{g_m} R : b_2}{\Delta', E \vdash_{g_m} P \mid Q \mid R : \Delta'(\xi)}}{\Delta', E \vdash_{g} m[P \mid Q \mid R]^{\xi} : b} $$

Moreover, by (22) and the polarity of $\_\mathsf{open}(\_)$, we have the desired relation

$$ \mathsf{Grps}\mathsf{open}(\mathit{group}(\chi)).\Delta'(\xi) \leqslant {}_G\mathsf{open}(g_n).\Delta'(\xi) \leqslant b_m = \Delta(\xi) $$

and by (21) also the desired safety predicate

$\mathcal{O}(group(\chi), group(\xi))$.

(Red Comm). The situation is that

$$m[\![(n_1 \dots n_k : \tau_1 \dots \tau_k).P \mid \langle M_1 \dots M_k \rangle \mid Q]\!]^\xi \xrightarrow{\xi:\mathsf{comm} \times(\tau_1,\dots,\tau_k)} m[\![P[n_i := M_i] \mid Q]\!]^\xi$$

With $\Delta(\xi) = b_m$ and $group(\xi) = g_m$, the typing of the left hand side takes the form

$$\dfrac{E \vdash m : \mathsf{amb}^{g_m}_{H_m}[br_m] \qquad \dfrac{\dfrac{\Delta, E, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash_{g_m} P : b_0}{\Delta, E \vdash_{g_m} (n_1 \dots n_k : \tau_1 \dots \tau_k).P : b_1} \quad \dfrac{\cdots E \vdash M_i : \tau_i' \cdots}{\Delta, E \vdash_{g_m} \langle M_1 \dots M_k \rangle : b_2} \quad \Delta, E \vdash_{g_m} Q : b_3}{\Delta, E \vdash_{g_m} (n_1 \dots n_k : \tau_1 \dots \tau_k).P \mid \langle M_1 \dots M_k \rangle \mid Q : b_m}}{\Delta, E \vdash_g m[\![(n_1 \dots n_k : \tau_1 \dots \tau_k).P \mid \langle M_1 \dots M_k \rangle \mid Q]\!]^\xi : b}$$

where with $\sigma = \times(\tau_1, \dots, \tau_k)$ and $\sigma' = \times(\tau_1', \dots, \tau_k')$ we have the relations

$$\mathsf{get}(\sigma).b_0 \leqslant b_1 \qquad \mathsf{put}(\sigma') \leqslant b_2 \qquad b_1 \mid b_2 \mid b_3 \leqslant b_m \qquad \varepsilon \leqslant b$$
$$(g^\uparrow, b_m) \overset{g_m}{\rightsquigarrow} (H_m, br_m) \tag{24}$$

We now define

$$\Delta' = \Delta[\xi \mapsto (b_0 \mid b_3)]$$

By Lemma 6 we infer that

$$\mathsf{put}(\sigma').\mathsf{get}(\sigma).\Delta'(\xi) \leqslant \mathsf{put}(\sigma').(\mathsf{get}(\sigma).b_0 \mid b_3) \leqslant \mathsf{put}(\sigma').(\varepsilon \mid (b_1 \mid b_3)) \leqslant (\mathsf{put}(\sigma').\varepsilon) \mid (b_1 \mid b_3) \leqslant b_2 \mid b_1 \mid b_3 \leqslant b_m \tag{25}$$

which by Lemma 9 applied to (24) shows

$$(g^\uparrow, \mathsf{put}(\sigma').\mathsf{get}(\sigma).\Delta'(\xi)) \overset{g_m}{\rightsquigarrow} (H_m, br_m) \tag{26}$$

As $\mathsf{put}(\sigma').\mathsf{get}(\sigma).\Delta'(\xi)$ contains a trace (we exploit that all behaviors are non-empty) of the form $\bullet \diamond \mathsf{put}(\sigma') \diamond \mathsf{get}(\sigma) \diamond tr_3$ with $\bullet$ feasible from $g^\uparrow$, Definition 2 tells us that

$$\sigma' \leqslant \sigma \text{ and thus } \forall i \in \{1 \dots k\} : \tau_i' \leqslant \tau_i. \tag{27}$$

Lemma 10 applied to (26) tells us that

$$(g^\uparrow, \Delta'(\xi)) \overset{g_m}{\rightsquigarrow} (H_m, br_m). \tag{28}$$

For all $i \in \{1 \dots k\}$ we from (27) and $E \vdash M_i : \tau_i'$ infer that

$$E \vdash M_i : \tau_i.$$

By assumption, $(n_1 \dots n_k : \sigma_1 \dots \sigma_k).P$ is non-conflicting with $\mathsf{names}(M_i)$, and therefore P is non-conflicting with $\{n_i\} \cup \mathsf{names}(M_i)$. We can thus apply Lemma 16 to the judgment

$$\Delta, E, n_1 : \tau_1, \cdots, n_k : \tau_k \vdash_{g_m} P : b_0$$

to infer that

$$\Delta, E \vdash_{g_m} P[n_i := M_i] : b_0.$$

Since the left hand side of the reduction is uniquely tagged, ξ does not occur in neither of P or Q (and certainly not in any of the M_i's). Therefore (by Lemma 11) it holds that

$$\Delta', E \vdash_{g_m} P[n_i := M_i] : b_0 \text{ and } \Delta', E \vdash_{g_m} Q : b_3.$$

Using the above observations, in particular (28), we can indeed construct a typing for the right hand side of the reduction:

$$\dfrac{E \vdash m : \mathsf{amb}^{g_m}_{H_m}[br_m] \qquad \dfrac{\Delta', E \vdash_{g_m} P[n_i := M_i] : b_0 \qquad \Delta', E \vdash_{g_m} Q : b_3}{\Delta', E \vdash_{g_m} P[n_i := M_i] \mid Q : \Delta'(\xi)}}{\Delta', E \vdash_g m[\![P[n_i := M_i] \mid Q]\!]^{\xi} : b}$$

Moreover, by (25) and (27), we also have the desired relation

$$\mathsf{put}(\sigma').\mathsf{get}(\sigma).\Delta'(\xi) \leqslant b_m = \Delta(\xi) \text{ where } \sigma' \leqslant \sigma.$$

(Red Repl). The situation is that $!P \xrightarrow{\epsilon} P' \mid !P$ where $P' \equiv_{\mathrm{t}} P$. The typing of the left hand side takes the form

$$\dfrac{\dfrac{\Delta, E \vdash_g P : b_0}{\Delta, E \vdash_g !P : b_0}}{\Delta, E \vdash_g !P : b}$$

where $b_0 \mid b_0 \leqslant b_0$ and $b_0 \leqslant b$. We now define Δ' as follows:

- $\Delta'(\xi) = \Delta(\xi)$ if ξ occurs in $\mathrm{dom}(\Delta)$;
- $\Delta'(\xi) = \Delta(\chi)$ if ξ occurs in P' with χ the tag of the corresponding position in P.

Using our assumptions, it is easy to see that this is well-defined. Moreover, since $\Delta, E \vdash_g P : b_0$ it clearly holds that

$$\Delta', E \vdash_g P' : b_0$$

and (by Lemma 11) also

$$\Delta', E \vdash_g P : b_0.$$

Using the above observations, in particular that $b_0 \mid b_0 \leqslant b_0$, we can indeed construct a typing for the right hand side of the reduction:

$$\dfrac{\dfrac{\dfrac{\Delta', E \vdash_g P' : b_0 \quad \dfrac{\Delta', E \vdash_g P : b_0}{\Delta', E \vdash_g\ !P : b_0}}{\Delta', E \vdash_g P' \mid !P : b_0}}{\Delta', E \vdash_g P' \mid !P : b}}$$

(Red PctxtP). The situation is that

$$\mathcal{PC}\lfloor P \rfloor \xrightarrow{\ell} \mathcal{PC}\lfloor Q \rfloor$$

because $P \xrightarrow{\ell} Q$, and that

$$\Delta, E \vdash_g \mathcal{PC}\lfloor P \rfloor : b. \tag{29}$$

By Lemma 17 there exists E_0, g_0, and b_0 such that

$$\Delta, E_0 \vdash_{g_0} P : b_0.$$

We can thus apply the induction hypothesis on $P \xrightarrow{\ell} Q$ to find Δ' such that

$$\Delta', E_0 \vdash_{g_0} Q : b_0$$

and such that Δ' has certain properties dependent on ℓ, in particular

$$\Delta' \text{ agrees with } \Delta \text{ on } \mathrm{dom}(\Delta) \setminus dom(\ell). \tag{30}$$

We can thus apply Lemma 17 to arrive at the desired judgment

$$\Delta', E \vdash_g \mathcal{PC}\lfloor Q \rfloor : b$$

provided we can show that $\Delta \underset{\mathcal{PC}}{\equiv} \Delta'$.

So let ξ be a tag occurring in $\mathcal{PC}$; we must show $\Delta(\xi) = \Delta'(\xi)$. Thus ξ occurs in $\mathcal{PC}\lfloor P \rfloor$ and therefore (from (29)) clearly $\xi \in \mathrm{dom}(\Delta)$, and (since $\mathcal{PC}\lfloor P \rfloor$ is uniquely tagged by assumption) ξ does not occur in P which by Lemma 20 (applied to $P \xrightarrow{\ell} Q$) shows that ξ does not occur in $dom(\ell)$. Thus $\xi \in \mathrm{dom}(\Delta) \setminus dom(\ell)$. But by (30) this shows that $\Delta'(\xi) = \Delta(\xi)$, as desired.

(Red $\equiv$). The situation is that

$$P' \xrightarrow{\ell} Q'$$

because $P' \equiv P$ and $P \xrightarrow{\ell} Q$ and $Q \equiv Q'$, and that

$$\Delta, E \vdash_g P' : b.$$

By Lemma 21 we infer that

$$\Delta, E \vdash_g P : b.$$

We can therefore apply the induction hypothesis (thanks to Lemma 19) to find Δ' such that

$$\Delta', E \vdash_g Q : b$$

and such that Δ' has certain properties dependent on ℓ. This is as desired, since by one more application of Lemma 21 we arrive at

$$\Delta', E \vdash_g Q' : b.$$