The Complexity of Model Checking Mobile Ambients

Witold Charatonik[1,2] Silvano Dal Zilio[3]
Andrew D. Gordon[3] Supratik Mukhopadhyay[1]
Jean-Marc Talbot[1,4]

May 2001

Technical Report
MSR–TR–2001–03

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

[1]Max-Planck-Institut für Informatik, Germany.
[2]University of Wrocław, Poland.
[3]Microsoft Research, United Kingdom.
[4]Laboratoire d'Informatique Fondamentale de Lille, France.

Publication History

A portion of this work appears in the book *Foundations of Software Science and Computation Structures: 4th International Conference, FOSSACS 2001*, F. Honsell, M. Miculan (Eds.), Springer *Lecture Notes in Computer Science* 2030:152–167, 2001.

Abstract

We settle the complexity bounds of the model checking problem for the replication-free ambient calculus with public names against the ambient logic without parallel adjunct. We show that the problem is PSPACE-complete. For the complexity upper-bound, we devise a new representation of processes that remains of polynomial size during process execution; this allows us to keep the model checking procedure in polynomial space. Moreover, we prove PSPACE-hardness of the problem for several quite simple fragments of the calculus and the logic; this suggests that there are no interesting fragments with polynomial-time model checking algorithms.

Contents

1 Introduction

The ambient calculus of Cardelli and Gordon (1999a, 1999b, 2000a) is a formalism for describing the mobility of both software and hardware. An ambient is a named cluster of running processes and nested sub-ambients. Each computation state has a spatial structure, the tree induced by the nesting of ambients. Mobility is abstractly represented by re-arrangement of this tree: an ambient may move inside or outside other ambients.

The ambient logic (Cardelli and Gordon 2000b) is a modal logic designed to specify properties of distributed and mobile computations programmed in the ambient calculus. As well as standard temporal modalities for describing the evolution of ambient processes, the logic includes novel spatial modalities for describing the tree structure of ambient processes. Serendipitously, these spatial modalities can also usefully describe the tree structure of semistructured databases (Cardelli and Ghelli 2001). Other work on the ambient logic includes a study of the process equivalence induced by the satisfaction relation (Sangiorgi 2001) and a study of the logic extended with constructs for describing private names (Cardelli and Gordon 2001).

The *model checking* problem is to decide whether a given object (in our case, an ambient process) satisfies (that is, is a model of) a given formula. Cardelli and Gordon (2000b) show decidability of the model checking problem for a finite-state fragment of the ambient calculus against the fragment of the ambient logic without their parallel adjunct modality. This finite-state ambient calculus omits the constructs for unbounded replication and dynamic name generation of the full calculus. The parallel adjunct modality is omitted because it is defined as an infinite quantification over processes. Cardelli and Gordon give no complexity analysis for their algorithm. Still, given the various possible applications of the logic, it is of interest to analyse the complexity of model checking mobile ambients.

In fact, a naive analysis of the algorithm of Cardelli and Gordon gives only a doubly exponential bound on its use of time and space. A more sophisticated analysis based on results in this paper shows that their algorithm works in single-exponential time on single-exponential space.

In this paper we settle the complexity bounds of the model checking problem for the finite-state ambient calculus (that is, the full calculus apart from replication and name generation) against the logic without parallel adjunct. Our main result (embodied in Theorems 3.11 and 4.2) is that the problem is PSPACE-complete. Hence, this situates model checking the ambient logic in the same complexity class as model checking concurrent programs against CTL and CTL* (Kupferman, Vardi, and Wolper 2000).

As we discuss in Section 2, there are two reasons why Cardelli and Gordon's algorithm uses exponential space. One of them is that a process may grow exponentially during its execution; the other is that there may be exponentially many processes reachable from a given one.

In Section 3, we present a new model checking algorithm that avoids these problems as follows.

- We avoid the first problem by devising a new representation of processes using a form of closure. The main feature of this representation is that substitutions that occur when communications take place within an ambient are not applied directly, but are kept explicit. These explicit substitutions prevent the representation blowing up exponentially in the size of the original process. The idea of using closures comes from DAG representations used in unification for avoiding exponential blow-up. A sequential substitution that we use here can be seen as a DAG representation of the substitution.

- To avoid the second problem, we first devise a non-deterministic algorithm for testing reachability that does not have to store all the reachable processes, but instead tests it on-the-fly, and then remove nondeterminism using Savitch's theorem (Savitch 1970). Hence we prove Theorem 3.11, that the model checking problem is solvable in PSPACE.

We show this upper bound to be tight in Section 4; Theorem 4.2 asserts that the model checking problem is PSPACE-hard. Actually, we give PSPACE-hardness results for various fragments of the logic and of the calculus. For instance, by Theorem 4.4, even for a calculus of purely mobile ambients (that is, a calculus without communication or the capability to dissolve ambients) and the logic without quantifiers, the problem is PSPACE-hard. Moreover, by Theorem 4.6, for a calculus of purely communicative ambients (that is, a calculus without the capabilities to move or to dissolve ambients) and the logic without quantifiers, the problem is also PSPACE-hard. Often in the study of model checking fixing the model or the formula makes the problem easier. Here this is not the case. Even if we fix the process to be the constant $\mathbf{0}$, the model checking problem remains PSPACE-hard. Although we do not prove PSPACE-hardness for fixed arbitrary formulas, our result is not much weaker: Theorem 4.7 asserts that for any level of the polynomial-time hierarchy we can find a fixed formula such that the model checking problem is hard for that level.

We end the main part of the paper with conclusions in Section 5. Appendixes A and B contain proofs of properties stated without proof in Sections 3 and 4, respectively.

2 Review of the Ambient Calculus and Logic

We present a finite-state ambient calculus (that is, the full calculus (Cardelli and Gordon 2000a) apart from replication and name generation) and the ambient logic without parallel adjunct. This is the same calculus and logic for which Cardelli and Gordon present a model checking algorithm (Cardelli and Gordon 2000b).

2.1 The Ambient Calculus with Public Names

The following table describes the expressions and processes of our calculus.

Expressions and Processes:

$M, N ::=$	expressions	$P, Q, R ::=$	processes
n	name	$\mathbf{0}$	inactivity
$\mathit{in}\ M$	can enter M	$P \mid Q$	composition
$\mathit{out}\ M$	can exit M	$M[P]$	ambient
$\mathit{open}\ M$	can open M	$M.P$	action
ϵ	null	$(n).P$	input
$M.M'$	path	$\langle M \rangle$	output

A name n is said to be *bound* in a process P if it occurs within an input prefix (n). A name is said to be *free* in a process P if there is an occurrence of n outside the scope of any input (n). We write $bn(P)$ and $fn(P)$ for respectively the set of bound names and the set of free names in P. We say two processes are α-equivalent if they are identical apart from the choice of bound names. We write $M\{n{\leftarrow}N\}$ and $P\{n{\leftarrow}N\}$ for the outcomes of capture-avoiding substitutions of the expression N for the name n in the expression M and the process P, respectively.

The semantics of the calculus is given by the relations $P \equiv Q$ and $P \to Q$. The *reduction* relation, $P \to Q$, defines the evolution of processes over time. The *structural congruence* relation, $P \equiv Q$, is an auxiliary relation used in the definition of reduction. When we define the satisfaction relation of the modal logic in the next section, we use an auxiliary relation, the *sublocation* relation, $P \downarrow Q$, which defines the spatial distribution of processes and holds when Q is the whole interior of a top-level ambient in P. We write $\to^*$ and $\downarrow^*$ for the reflexive and transitive closure of $\to$ and $\downarrow$, respectively.

Structural Congruence $P \equiv Q$

P, Q are α-equivalent $\Rightarrow P \equiv Q$	(Struct Refl)
$Q \equiv P \Rightarrow P \equiv Q$	(Struct Symm)
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	(Struct Trans)
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	(Struct Par)
$P \equiv Q \Rightarrow M[P] \equiv M[Q]$	(Struct Amb)
$P \equiv Q \Rightarrow M.P \equiv M.Q$	(Struct Action)
$P \equiv Q \Rightarrow (n).P \equiv (n).Q$	(Struct Input)
$P \mid Q \equiv Q \mid P$	(Struct Par Comm)
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	(Struct Par Assoc)
$P \mid \mathbf{0} \equiv P$	(Struct Zero Par)
$\epsilon.P \equiv P$	(Struct ϵ)
$(M.M').P \equiv M.M'.P$	(Struct .)

Reduction $P \to Q$ and Sublocation $P \downarrow Q$:

$n[\mathit{in}\ m.P \mid Q] \mid m[R] \to m[n[P \mid Q] \mid R]$	(Red In)
$m[n[\mathit{out}\ m.P \mid Q] \mid R] \to n[P \mid Q] \mid m[R]$	(Red Out)

$$
\begin{array}{ll}
open\, n.P \mid n[Q] \rightarrow P \mid Q & \text{(Red Open)} \\
\langle M \rangle \mid (n).P \rightarrow P\{n{\leftarrow}M\} & \text{(Red I/O)} \\
\\
P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R & \text{(Red Par)} \\
P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q] & \text{(Red Amb)} \\
P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q' & \text{(Red} \equiv\text{)} \\
\\
P \equiv n[P'] \mid P'' \Rightarrow P \downarrow P' & \text{(Loc)}
\end{array}
$$

The following example shows that the size of reachable processes may be exponential, and that there may be a reduction path of exponential length. The algorithm given in (Cardelli and Gordon 2000b) may use exponential space to check properties of this example.

Consider the family of processes $(P_k)_{k\geq 0}$, recursively defined by the equations $P_0 = (n).(p[n] \mid q[\mathbf{0}])$ and $P_{k+1} = (n_{k+1}).(\langle n_{k+1}.n_{k+1}\rangle \mid P_k)$. Intuitively, the process P_{k+1} inputs a capability, calls it n_{k+1}, doubles it, and outputs the result to the process P_k. We have the following, where $M^1 = M$ and $M^{k+1} = M.M^k$.

$$
\begin{array}{llll}
\langle in\, q.out\, q\rangle \mid P_0 & \rightarrow^1 & p[in\, q.out\, q] & \mid q[\mathbf{0}] \\
\langle in\, q.out\, q\rangle \mid P_1 & \rightarrow^2 & p[(in\, q.out\, q)^2] & \mid q[\mathbf{0}] \\
\langle in\, q.out\, q\rangle \mid P_2 & \rightarrow^3 & p[(in\, q.out\, q)^4] & \mid q[\mathbf{0}] \\
\langle in\, q.out\, q\rangle \mid P_k & \rightarrow^{k+1} & p[(in\, q.out\, q)^{2^k}] & \mid q[\mathbf{0}]
\end{array}
$$

Since $(in\; q.out\; q)^{2^k}$ is a sequence of 2^k copies of $in\; q.out\; q$, the process $p[(in\, q.out\, q)^{2^k}] \mid q[\mathbf{0}]$ reduces in 2^{k+1} steps to $p[\mathbf{0}] \mid q[\mathbf{0}]$. Therefore, we have $\langle in\, q.out\, q\rangle \mid P_k \rightarrow^{(k+1)+2^{k+1}} p[\mathbf{0}] \mid q[\mathbf{0}]$.

This example points out two facts. First, using a simple representation of processes (such as the one proposed in (Cardelli and Gordon 2000b)), it may be that the size of a process considered during model checking grows exponentially bigger than the size of the initial process. Second, during the model checking procedure, there may be an exponential number of reachable processes to consider. Therefore, a direct implementation of the algorithm proposed in (Cardelli and Gordon 2000b) may use space exponential in the size of the input process.

These remarks motivate the approach taken in this paper. First, we devise a new representation for ambient processes that remains of polynomial size with respect to to the input process. Second, we give a non-deterministic algorithm for testing reachability that uses only polynomial space in the combined size of the problem; then by an application of Savitch's theorem (Savitch 1970) we remove nondeterminism and obtain a deterministic version that itself uses only polynomial space.

2.2 The Logic (for Public Names)

We describe the formulas and satisfaction relation of the logic.

Logical Formulas:

η	a name n or a variable x
$\mathcal{A}, \mathcal{B} ::=$	formula
$\mathbf{T}$	true
$\neg\mathcal{A}$	negation
$\mathcal{A} \vee \mathcal{B}$	disjunction
$\mathbf{0}$	void
$\eta[\mathcal{A}]$	ambient match
$\mathcal{A} \mid \mathcal{B}$	composition match
$\mathcal{A}@\eta$	location adjunct
$\exists x.\mathcal{A}$	existential quantification
$\Diamond\mathcal{A}$	sometime modality
$\diamondsuit\mathcal{A}$	somewhere modality

We assume that names and variables belong to two disjoint vocabularies. We write $\mathcal{A}\{x{\leftarrow}m\}$ for the outcome of substituting each free occurrence of the variable x in the formula $\mathcal{A}$ with the name m. We say a formula $\mathcal{A}$ is closed if and only if it has no free variables (though it may contain free names).

Intuitively, we interpret closed formulas as follows. The formulas $\mathbf{T}$, $\neg\mathcal{A}$, and $\mathcal{A} \vee \mathcal{B}$ embed propositional logic. The formulas $\mathbf{0}$, $\eta[\mathcal{A}]$, and $\mathcal{A} \mid \mathcal{B}$ are spatial modalities. A process satisfies $\mathbf{0}$ if it is structurally congruent to the empty process $\mathbf{0}$. It satisfies $n[\mathcal{A}]$ if it is structurally congruent to an ambient $n[P]$ where P satisfies $\mathcal{A}$. A process P satisfies $\mathcal{A} \mid \mathcal{B}$ if it can be decomposed into two subprocesses, $P \equiv Q \mid R$, where Q satisfies $\mathcal{A}$, and R satisfies $\mathcal{B}$. The formula $\exists x.\mathcal{A}$ is an existential quantification over names. The formulas $\Diamond\mathcal{A}$ (sometime) and $\diamondsuit\mathcal{A}$ (somewhere) quantify over time and space, respectively. A process satisfies $\Diamond\mathcal{A}$ if it has a temporal successor, that is, a process into which it evolves, that satisfies $\mathcal{A}$. A process satisfies $\diamondsuit\mathcal{A}$ if it has a spatial successor, that is, a sublocation, that satisfies $\mathcal{A}$. Finally, a process P satisfies the formula $\mathcal{A}@n$ if the ambient $n[P]$ satisfies $\mathcal{A}$.

The satisfaction relation $P \models \mathcal{A}$ formalizes these intuitions.

Satisfaction $P \models \mathcal{A}$ (for $\mathcal{A}$ closed):

$P \models \mathbf{T}$		
$P \models \neg\mathcal{A}$	$\triangleq$	$\neg(P \models \mathcal{A})$
$P \models \mathcal{A} \vee \mathcal{B}$	$\triangleq$	$P \models \mathcal{A} \vee P \models \mathcal{B}$
$P \models \mathbf{0}$	$\triangleq$	$P \equiv \mathbf{0}$
$P \models n[\mathcal{A}]$	$\triangleq$	$\exists P'.P \equiv n[P'] \wedge P' \models \mathcal{A}$
$P \models \mathcal{A} \mid \mathcal{B}$	$\triangleq$	$\exists P', P''.P \equiv P' \mid P'' \wedge P' \models \mathcal{A} \wedge P'' \models \mathcal{B}$
$P \models \exists x.\mathcal{A}$	$\triangleq$	$\exists m.P \models \mathcal{A}\{x{\leftarrow}m\}$
$P \models \Diamond\mathcal{A}$	$\triangleq$	$\exists P'.P \rightarrow^* P' \wedge P' \models \mathcal{A}$
$P \models \diamondsuit\mathcal{A}$	$\triangleq$	$\exists P'.P \downarrow^* P' \wedge P' \models \mathcal{A}$
$P \models \mathcal{A}@n$	$\triangleq$	$n[P] \models \mathcal{A}$

We use $\Box\mathcal{A}$ (everytime modality), ❑$\mathcal{A}$ (everywhere modality) and $\forall x.\mathcal{A}$ (universal quantification) as abbreviations for $\neg(\Diamond\neg\mathcal{A})$, $\neg($✧$\neg\mathcal{A})$ and $\neg(\exists x.\neg\mathcal{A})$, respectively.

3 A Model Checking Algorithm

We show that the model checking problem can be decided in polynomial space by devising a new representation of processes (Section 3.1) that remains polynomial in the size of the initial process (Section 3.2). In Section 3.3 we present a new model checking algorithm based on this representation.

Since the reduction relation is defined up to α-equivalence, we may assume for the purposes of computing reachable processes that the free and bound names of every ambient process are distinct, and moreover that the bound names are pairwise distinct.

3.1 A Polynomial-Space Representation

We give in this section a new representation for ambient processes based on *normal closures* (It is different from the *normal form* of processes introduced in (Cardelli and Gordon 2000b)). We also present basic operations on closures and prove that closures indeed simulate the processes they represent. All proofs not in this section (in particular, proofs of Propositions 3.1–3.4) can be found in the appendix.

Annotated Processes, Substitutions, Closures:

$\tilde{P} ::=$	annotated process
$\quad\prod_{i\in I}\pi_i$	multiset of primes
$\pi ::=$	prime
$\quad M[\tilde{P}]$	ambient
$\quad M(o).\tilde{P}$	action, with offset $o \geq 0$
$\quad (n).\tilde{P}$	input
$\quad \langle M\rangle$	output
$\sigma ::= \{n_1\leftarrow M_1\}\cdots\{n_k\leftarrow M_k\}$	sequential substitution, $k \geq 0$
$\langle\tilde{P};\sigma\rangle$	closure

In a sequential substitution $\{n_1\leftarrow M_1\}\cdots\{n_k\leftarrow M_k\}$, the expression M_i lies in the scope of the bindings for the remaining names $n_{i+1}, \ldots, n_k$. We denote by ι the empty sequence of substitutions and treat it as the identity substitution. A sequential substitution σ is said to be *acyclic* if either $\sigma = \iota$ or $\sigma = \{x\leftarrow M\}\sigma'$, where x does not occur in σ' and σ' is an acyclic substitution.

For an annotated process $\tilde{P}$, we define free and bound names in the same way as for ambient processes. Let $names(\sigma)$ be the set of all names occurring in σ.

We define a partial mapping $\mathcal{U}$ from closures to the set of ambient processes. Intuitively, it unfolds a closure to the process it represents by applying the substitution and cutting off the prefix defined by the offset. Roughly speaking, the expression $\mathcal{U}(\tilde{P},\sigma)$ is defined if the offsets within the annotated process do not exceed the length of the expression they are associated with. The unfolding $\mathcal{U}(\tilde{P},\sigma)$ is defined as follows.

The Unfolding $\mathcal{U}(\tilde{P},\sigma)$ of a Closure $\langle\tilde{P};\sigma\rangle$:

$$\mathcal{U}(\textstyle\prod_{i\in I}\pi_i,\sigma) = \begin{cases}\mathcal{U}(\pi_1,\sigma) \mid \ldots \mid \mathcal{U}(\pi_n,\sigma) & \text{if } I=\{1,\ldots,n\}\neq\varnothing \\ \mathbf{0} & \text{otherwise}\end{cases}$$

$$\mathcal{U}(M[\tilde{P}],\sigma) = M\sigma[\mathcal{U}(\tilde{P},\sigma)]$$

$$\mathcal{U}(M(o).\tilde{P},\sigma) = \begin{cases}N_{o+1}.\cdots.N_l.\mathcal{U}(\tilde{P},\sigma) & \text{if } M\sigma = N_1.\cdots.N_l, o<l \text{ and } N_i \\ & \quad\text{being either a name or of the form} \\ & \quad cap\ N' \text{ with } cap\in\{in, out, open\} \\ \text{undefined} & \text{otherwise}\end{cases}$$

$$\mathcal{U}((n).\tilde{P},\sigma) = (n).\mathcal{U}(\tilde{P},\sigma)$$
$$\mathcal{U}(\langle M\rangle,\sigma) = \langle M\sigma\rangle$$

We are only interested in a particular kind of closure, which we refer to as *normal.* Let a closure $\langle\tilde{P};\sigma\rangle$ be normal if $\mathcal{U}(\tilde{P},\sigma)$ is defined and if it meets some technical conditions about free and bound names

Definition 1 *A closure $\langle\tilde{P};\sigma\rangle$ is* normal *if:*

(1) *$\mathcal{U}(\tilde{P},\sigma)$ is defined,*

(2) *$bn(\tilde{P})\cap(fn(\tilde{P})\cup names(\sigma))=\varnothing$,*

(3) *every name n in $\tilde{P}$ occurs at most once within an input,*

(4) *every offset o occurring in the scope of an input in $\tilde{P}$ is equal to* 0*, and*

(5) *σ is acyclic.*

The next proposition says that our representation of ambient processes with normal closures preserves their basic properties. We write $\{\}$ and $+\!\!+$ for the empty multiset and the multiset union operation, respectively.

Proposition 3.1 (Structural Equivalences) *Let $\langle\prod_{i\in I}\pi_i;\sigma\rangle$ be a normal closure. Then*

(1) *$\mathcal{U}(\prod_{i\in I}\pi_i,\sigma)\equiv\mathbf{0}$ iff $I=\varnothing$.*

(2) *$\mathcal{U}(\prod_{i\in I}\pi_i,\sigma)\equiv M[Q]$ iff $\exists M',\tilde{Q}$: I is a singleton $\{i\}$, $\pi_i=M'[\tilde{Q}]$, $M'\sigma=M$, $\mathcal{U}(\tilde{Q},\sigma)\equiv Q$.*

(3) *$\mathcal{U}(\prod_{i\in I}\pi_i,\sigma)\equiv P'\mid P''$ iff $\exists J,K$: $J\cup K=I$, $J\cap K=\varnothing$, $P'\equiv\mathcal{U}(\prod_{j\in J}\pi_j,\sigma)$, $P''\equiv\mathcal{U}(\prod_{k\in K}\pi_k,\sigma)$.*

(4) $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv \langle M\rangle$ *iff* $\exists M' : I$ *is a singleton* $\{i\}$, $\pi_i = \langle M'\rangle$ *and* $M'\sigma = M$.

(5) $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv (n).P$ *iff* $\exists\tilde{P} : I$ *is a singleton* $\{i\}$, $\pi_i = (n).\tilde{P}$ *and* $\mathcal{U}(\tilde{P},\sigma) \equiv P$.

Next, we present how the reduction and sublocation transitions $\rightarrow$, $\downarrow$ can be defined on closures. Due to this particular representation and the fact that some part of the ambient process is contained in the sequential substitution, some auxiliary subroutines are needed.

One can see in the definition of $\mathcal{U}$ that only expressions M in the annotated process are affected by the sequential substitution. For the sublocation transition, it is important to extract the name represented by the expression M under the substitution σ. So, one of those subroutines, $nam(M,\sigma)$, consists in recovering from an expression M the name it effectively represents within the substitution σ.

The reduction transition for a closure $\langle\tilde{P};\sigma\rangle$ requires some other auxiliary subroutines, which are more specifically dedicated to the case where the substitution applied on the expression M leads to a sequence of capabilities $in\ M'$, $out\ M'$, $open\ M'$. Intuitively, the outcome of applying the substitution σ to an expression M contained within $\tilde{P}$ is a finite sequence of either capabilities of the form $in\,M'$, $out\,M'$, $open\,M'$, or names not bound by the substitution. We need a subroutine to compute the length of this sequence in terms of capabilities. To keep the algorithm in polynomial space, we must simply be able to compute this length without applying explicitly σ on M; this is the role of $len(M,\sigma)$.

Now, from the definition of the reduction on ambient processes, one can see that the reduction consumes one capability: once the reduction is done, the involved capability disappears from the resulting process. This is slightly different for the representation we have proposed: a sequence of capabilities can be partially contained in a sequential substitution σ. This substitution remains fixed during the execution of capabilities and the offset attached to this sequence plays the role of a program counter. Therefore, to perform a reduction step one has to extract the first capability to execute from a sequence of capabilities, M, a substitution, σ, and an offset, o. This is computed by $fst(M,o,\sigma)$.

The next subroutine introduced here, $split(M(o).\tilde{P},\sigma)$, computes a pair from a prime, $M(o).\tilde{P}$, and a sequential substitution, σ. The first component of this result is the first capability to be executed in $\langle\{M(o).\tilde{P}\};\sigma\rangle$ (the one in head position). The second component is the remaining annotated process once this first capability has been executed.

The Auxiliary Functions *nam*, *len*, *fst* **and** *split***:**

$$nam(n,\{m{\leftarrow}M\}\sigma) = \begin{cases} nam(M,\sigma) & \text{if } n = m \\ nam(n,\sigma) & \text{otherwise} \end{cases}$$

$$nam(n,\iota) = n$$

$$len(\epsilon,\sigma) = 0$$

$$len(M.N,\sigma) = len(M,\sigma) + len(N,\sigma)$$
$$len(M,\sigma) = 1 \quad \text{if } M \in \{\mathit{in}\ N, \mathit{out}\ N, \mathit{open}\ N\}$$
$$len(n,\{m{\leftarrow}M\}\sigma) = \begin{cases} len(M,\sigma) & \text{if } n = m \\ len(n,\sigma) & \text{otherwise} \end{cases}$$
$$len(n,\iota) = 1$$

$$fst(M.N,o,\sigma) = \begin{cases} fst(M,o,\sigma) & \text{if } len(M,\sigma) > o \\ fst(N,o-len(M,\sigma),\sigma) & \text{otherwise} \end{cases}$$
$$fst(cap\ N,0,\sigma) = cap\ (nam(N,\sigma)) \qquad \text{for } cap \text{ in } \{\mathit{in}, \mathit{out}, \mathit{open}\}$$
$$fst(n,o,\{m{\leftarrow}M\}\sigma) = \begin{cases} fst(M,o,\sigma) & \text{if } n = m \\ fst(n,o,\sigma) & \text{otherwise} \end{cases}$$

$$split(M(o).\tilde{P},\sigma) = \begin{cases} (fst(M,o,\sigma),\{M(o+1).\tilde{P}\}) & \text{if } len(M,\sigma) > o+1 \\ (fst(M,o,\sigma),\tilde{P}) & \text{otherwise} \end{cases}$$

Notice that $nam(M,\sigma)$ is undefined if M is of the form ϵ, $N.N'$, $\mathit{in}\ N$, $\mathit{out}\ N$, or $\mathit{open}\ N$. Therefore, the expression $nam(M,\sigma)$ is either undefined or is evaluated to a name. Moreover, we can compute the name returned by $nam(M,\sigma)$, or whether it is undefined, in linear time. The number returned by $len(M,\sigma)$ can be computed in polynomial space[1]. We can compute the capability returned by $fst(M,o,\sigma)$ and the pair returned by $split(M(o).\tilde{P},\sigma)$, or whether they are undefined, in polynomial space.

Suppose $\langle\tilde{P};\sigma\rangle$ is a normal closure containing an action $M(o).\tilde{Q}$. From the definition of a normal closure, $len(M,\sigma) > o$, and if the action occurs under an input variable n, then the offset $o = 0$. If n occurs in M and gets bound to ϵ by an I/O step, it may be that $len(M,\{n{\leftarrow}\epsilon\}\sigma) = 0$. So, in the transition rule for I/O, we need to re-normalize the closure representing the outcome of the transition. We do so using the following subroutines, $norm(\tilde{P},\sigma)$ and $norm(\pi,\sigma)$, that return the annotated process obtained by removing from $\tilde{P}$ and π, respectively, any prefix $M(o)$ such that $len(M,\sigma) = 0$.

The Auxiliary Functions *norm***:**

$$norm(\textstyle\prod_{i\in 1..k}\pi_i,\sigma) = \begin{cases} \{\} & \text{if } k = 0 \\ norm(\pi_1,\sigma) +\!\!+ \cdots +\!\!+ norm(\pi_k,\sigma) & \text{otherwise} \end{cases}$$
$$norm(M[\tilde{P}],\sigma) = \{M[norm(\tilde{P},\sigma)]\}$$
$$norm(M(o).\tilde{P},\sigma) = \begin{cases} norm(\tilde{P},\sigma) & \text{if } len(M,\sigma) = 0 \\ \{M(o).norm(\tilde{P},\sigma)\} & \text{otherwise} \end{cases}$$
$$norm((n).\tilde{P},\sigma) = \{(n).norm(\tilde{P},\sigma)\}$$
$$norm(\langle M\rangle,\sigma) = \{\langle M\rangle\}$$

Next, we define a transition relation, $\langle\tilde{P};\sigma\rangle \rightarrow \langle\tilde{P}';\sigma'\rangle$, and a sublocation

[1] We are not concerned here with time complexity; a naive algorithm for computing $len(M,\sigma)$, as presented here, runs in exponential time in the worst case. However, it is quite easy to provide a version of this function that runs in polynomial time.

relation, $\langle \tilde{P}; \sigma \rangle \downarrow \langle \tilde{P}'; \sigma \rangle$, on closures. These relations simulate the reduction and the sublocation relations on processes defined in Section 2.1.

Transitions and Sublocations of Closures:

(Trans In)

$$\frac{split(\pi,\sigma) = (in\, m, \tilde{P}) \quad nam(M,\sigma) = m \quad nam(N,\sigma) = n}{\langle \{N[\{\pi\} +\!\!+ \tilde{Q}], M[\tilde{R}]\}; \sigma\rangle \to \langle \{M[\{N[\tilde{P} +\!\!+ \tilde{Q}]\} +\!\!+ \tilde{R}]\}; \sigma\rangle}$$

(Trans Out)

$$\frac{split(\pi,\sigma) = (out\, m, \tilde{P}) \quad nam(M,\sigma) = m \quad nam(N,\sigma) = n}{\langle \{M[\{N[\{\pi\} +\!\!+ \tilde{Q}]\} +\!\!+ \tilde{R}]\}; \sigma\rangle \to \langle \{N[\tilde{P} +\!\!+ \tilde{Q}], M[\tilde{R}]\}; \sigma\rangle}$$

(Trans Open)

$$\frac{split(\pi,\sigma) = (open\, n, \tilde{P}) \quad nam(M,\sigma) = n}{\langle \pi, \{M[\tilde{Q}]\}; \sigma\rangle \to \langle \tilde{P} +\!\!+ \tilde{Q}; \sigma\rangle}$$

(Trans I/O)

$$\frac{\tilde{P}' = norm(\tilde{P}, \{n \leftarrow M\}\sigma)}{\langle \{(n).\tilde{P}, \langle M\rangle\}; \sigma\rangle \to \langle \tilde{P}'; \{n \leftarrow M\}\sigma\rangle}$$

(Trans Par)

$$\frac{\langle \tilde{P}; \sigma\rangle \to \langle \tilde{P}'; \sigma'\rangle}{\langle \tilde{P} +\!\!+ \tilde{Q}; \sigma\rangle \to \langle \tilde{P}' +\!\!+ \tilde{Q}; \sigma'\rangle}$$

(Trans Amb)

$$\frac{\langle \tilde{P}; \sigma\rangle \to \langle \tilde{P}'; \sigma'\rangle \quad nam(M,\sigma) = n}{\langle \{M[\tilde{P}]\}; \sigma\rangle \to \langle \{M[\tilde{P}']\}; \sigma'\rangle}$$

(Loc)

$$\frac{nam(M,\sigma) = m}{\langle \tilde{Q} +\!\!+ \{M[\tilde{P}]\}; \sigma\rangle \downarrow \langle \tilde{P}; \sigma\rangle}$$

The condition for (Loc) ensures simply that the expression M together with σ is a name. For two normal closures $\langle P; \sigma\rangle$, $\langle P'; \sigma'\rangle$, deciding whether $\langle P; \sigma\rangle \downarrow \langle P'; \sigma'\rangle$ can be achieved in polynomial space. There is no rule corresponding to (Red $\equiv$) since we always keep closures in normal form. The two rules (Trans Par) and (Trans Amb) correspond to the congruence rules (Red Par) and (Red Amb) for reduction.

In the same way as for ambient processes, we define the relations $\to^*$ and $\downarrow^*$ (on closures) as the reflexive and transitive closures of $\to$ and $\downarrow$, respectively.

Proposition 3.2

(1) *If $\langle \tilde{P}; \sigma\rangle$ is normal and $\langle \tilde{P}; \sigma\rangle \downarrow^* \langle \tilde{P}'; \sigma\rangle$ then $\langle \tilde{P}'; \sigma\rangle$ is normal.*

(2) *If $\langle \tilde{P}; \sigma\rangle$ is normal and $\langle \tilde{P}; \sigma\rangle \to^* \langle \tilde{P}'; \sigma'\rangle$ then $\langle \tilde{P}'; \sigma'\rangle$ is normal.*

The next proposition says that the representation of processes as closures preserves sublocations and reductions.

Proposition 3.3 (Sublocation Equivalences) *Assume $\langle \tilde{P}; \sigma\rangle$ is a normal closure. If $\langle \tilde{P}; \sigma\rangle \downarrow \langle \tilde{Q}; \sigma\rangle$ then $\mathcal{U}(\tilde{P},\sigma) \downarrow \mathcal{U}(\tilde{Q},\sigma)$. If $\mathcal{U}(\tilde{P},\sigma) \downarrow Q$ then there exists $\tilde{Q}$ such that $\langle \tilde{P}; \sigma\rangle \downarrow \langle \tilde{Q}; \sigma\rangle$ and $\mathcal{U}(\tilde{Q},\sigma) \equiv Q$.*

The following proposition is a counterpart of Proposition 3.3. It refers to time in the same way as Proposition 3.3 refers to space.

Proposition 3.4 (Reduction Equivalences) *Assume $\langle \tilde{P}; \sigma \rangle$ is a normal closure. If $\langle \tilde{P}; \sigma \rangle \to \langle \tilde{P}'; \sigma' \rangle$ then $\mathcal{U}(\tilde{P}, \sigma) \to \mathcal{U}(\tilde{P}', \sigma')$. If $\mathcal{U}(\tilde{P}, \sigma) \to P'$ then there exists $\langle \tilde{P}'; \sigma' \rangle$ such that $\langle \tilde{P}; \sigma \rangle \to \langle \tilde{P}'; \sigma' \rangle$ and $\mathcal{U}(\tilde{P}', \sigma') \equiv P'$.*

Propositions 3.1–3.4 are enough to prove that normal closures indeed simulate the processes they represent.

3.2 Size of the Representation

We show that closures indeed give a polynomial representation of processes. To do this, we have to bound the size of offsets that occur in closures.

For a given object (a closure or a process) O, by $|O|$ we mean the length of its string representation and by $\|O\|$ the number of nodes in its tree representation. We assume that an offset is represented by a single node in the tree representation.

Lemma 3.5 *Suppose that $\langle \tilde{P}; \sigma \rangle \to \langle \tilde{P}'; \sigma' \rangle$. Then $\|\langle \tilde{P}'; \sigma' \rangle\| \le \|\langle \tilde{P}; \sigma \rangle\|$.*

Proof By a simple case analysis on the derivation of $\langle \tilde{P}; \sigma \rangle \to \langle \tilde{P}'; \sigma' \rangle$. In cases (Trans In), (Trans Out) and (Trans Open), the transition either does not change or decreases the representation's size. In case (Trans I/O), the three nodes representing input, output and process composition $((), \langle\rangle, .)$ together with the representation of x and M are replaced with two nodes representing assignment and substitution composition $(\leftarrow, \{\})$ together with the representation of x and M. Thus the tree decreases by one node. ∎

Proposition 3.6 *Assume $\langle \tilde{P}; \sigma \rangle$ is normal and $\langle \tilde{P}; \sigma \rangle \to \langle \tilde{P}'; \sigma' \rangle$. Then all offsets used in $\tilde{P}$ and $\tilde{P}'$ can be represented by the same number of bits, polynomial in $|\langle \tilde{P}; \sigma \rangle|$ and, with such a representation, $|\langle \tilde{P}'; \sigma' \rangle| \le |\langle \tilde{P}; \sigma \rangle|$.*

Proof A simple induction on the length of the substitution σ' proves that the offsets in $\tilde{P}'$ are bounded by the value $\|\langle \tilde{P}'; \sigma' \rangle\|^{\|\langle \tilde{P}'; \sigma' \rangle\|}$. By Lemma 3.5, they are also bounded by $\|\langle \tilde{P}; \sigma \rangle\|^{\|\langle \tilde{P}; \sigma \rangle\|}$ and then all offsets used in $\tilde{P}$ and $\tilde{P}'$ are bounded by this value, which can be represented on $\|\langle \tilde{P}; \sigma \rangle\| \cdot (\lfloor \log(\|\langle \tilde{P}; \sigma \rangle\|) \rfloor + 1)$ bits. With this representation of offsets, incrementing an offset does not increase the size of its string representation. Thus no transitions can increase the length of the string representations of closures. ∎

The following proposition is a key fact in the proof that our model checking algorithm and also the algorithm of Cardelli and Gordon (2000b) terminate in exponential time. It implies that the computation tree of a given process might be very deep and very narrow (as in our example in Section 2) or not so deep and wider; in any case the number of nodes in the tree remains exponentially bounded. A naive argument (without using closures) gives only a doubly exponential bound on the number of reachable processes: one can prove that the computation tree of a given process is at most exponentially deep (as our example in Section 2 shows, this bound is tight) and that the number of successors for every node is at most polynomial. For example, the closure $\langle\{n[inn(0).\tilde{P}_0],\ldots,n[inn(0).\tilde{P}_k]\};\sigma\rangle$ has at most k^2 different successors. These two facts do not give, however, the exponential bound on the number of nodes in the tree, which is given by the following proposition.

Proposition 3.7 *Let $\langle\tilde{P};\sigma\rangle$ be a normal closure. Then there exist at most exponentially many $\langle\tilde{P}';\sigma'\rangle$ such that $\langle\tilde{P};\sigma\rangle \to^* \langle\tilde{P}';\sigma'\rangle$.*

Proof This is a direct consequence of Proposition 3.6 and the observation that there are only exponentially many strings of polynomial length. ■

Proposition 3.8 *The reachability problem for normal closures is decidable in PSPACE.*

Proof Take any instance $\langle\tilde{P};\sigma\rangle$, $\langle\tilde{P}';\sigma'\rangle$ of the reachability problem. To decide whether $\langle\tilde{P};\sigma\rangle \to^* \langle\tilde{P}';\sigma'\rangle$, we first define a nondeterministic algorithm that starting from $\langle\tilde{P};\sigma\rangle$ guesses an immediate successor of the current closure until it reaches $\langle\tilde{P}';\sigma'\rangle$ or there are no further successors. By Proposition 3.6 the algorithm requires only polynomial space (we have to store only the current closure and its one immediate successor); Proposition 3.7 implies termination. Finally, using the general statement of Savitch's theorem (Savitch 1970) (NPSPACE$(S(n)) \subseteq$ PSPACE$(S(n)^2)$), this non-deterministic algorithm can be turned into a deterministic one. ■

3.3 A New Algorithm

We propose a new algorithm, $\mathit{Check}(\tilde{P},\sigma,\mathcal{A})$, to check whether the ambient process simulated by $\langle\tilde{P};\sigma\rangle$ satisfies the closed formula $\mathcal{A}$. For each ambient process, P, we only consider the closure, $\mathcal{F}(P)$, obtained using the *folding* function defined as follows. We prove (Proposition 3.10), that $P \models \mathcal{A}$ if and only if $\mathit{Check}(\mathcal{F}(P),\iota,\mathcal{A})$ returns the Boolean value **T**.

The Folding $\mathcal{F}(P)$ of a Process P:

$$\mathcal{F}(\mathbf{0}) = \{\}$$
$$\mathcal{F}(P \mid Q) = \mathcal{F}(P) +\!\!+ \mathcal{F}(Q)$$
$$\mathcal{F}(M[P]) = \{M[\mathcal{F}(P)]\}$$
$$\mathcal{F}((n).P) = \{(n).\mathcal{F}(P)\}$$
$$\mathcal{F}(\langle M\rangle) = \{\langle M\rangle\}$$
$$\mathcal{F}(M.P) = \begin{cases}\mathcal{F}(P) & \text{if } \mathit{len}(M,\iota)=0\\ \{M(0).\mathcal{F}(P)\} & \text{otherwise}\end{cases}$$

For any process P, the closure $\langle\mathcal{F}(P);\iota\rangle$ is normal and $\mathcal{U}(\mathcal{F}(P),\iota)$ is structurally congruent to P. Furthermore, $\mathcal{F}(P)$ can be computed in linear time in the size of P.

For the model checking problem, $P \models \mathcal{A}$, we may assume without loss of generality that the free names of $\mathcal{A}$ are disjoint from the bound names of P. We denote by $fn(\tilde{P}, \sigma)$ the set $(fn(\tilde{P}) \cup names(\sigma)) \setminus dom(\sigma)$.

Computing Whether a Process Satisfies a Closed Formula:

$Check(\tilde{P}, \sigma, \mathbf{T}) = \mathbf{T}$
$Check(\tilde{P}, \sigma, \neg\mathcal{A}) = \neg Check(\tilde{P}, \sigma, \mathcal{A})$
$Check(\tilde{P}, \sigma, \mathcal{A} \vee \mathcal{B}) = Check(\tilde{P}, \sigma, \mathcal{A}) \vee Check(\tilde{P}, \sigma, \mathcal{B})$
$Check(\prod_{i\in I} \pi_i, \sigma, \mathbf{0}) = \begin{cases} \mathbf{T} & \text{if } I = \varnothing \\ \mathbf{F} & \text{otherwise} \end{cases}$
$Check(\prod_{i\in I} \pi_i, \sigma, n[\mathcal{A}]) =$
$\quad \begin{cases} Check(\tilde{Q}, \sigma, \mathcal{A}) & \text{if } I = \{i\},\ \pi_i = M[\tilde{Q}],\ nam(M, \sigma) = n \\ \mathbf{F} & \text{otherwise} \end{cases}$
$Check(\prod_{i\in I} \pi_i, \sigma, \mathcal{A} \mid \mathcal{B}) = \bigvee_{J\subseteq I}(Check(\prod_{j\in J} \pi_j, \sigma, \mathcal{A}) \wedge$
$\qquad Check(\prod_{k\in I-J} \pi_k, \sigma, \mathcal{B}))$
$Check(\tilde{P}, \sigma, \exists x.\mathcal{A}) =$ let $\{m_1, \ldots, m_k\} = fn(\tilde{P}, \sigma) \cup fn(\mathcal{A})$ in
$\qquad$ let $m_0 \notin \{m_1, \ldots, m_k\} \cup bn(\tilde{P}) \cup dom(\sigma)$ be fresh in
$\qquad \bigvee_{i\in 0..k} Check(\tilde{P}, \sigma, \mathcal{A}\{x\leftarrow m_i\})$
$Check(\tilde{P}, \sigma, \Diamond\mathcal{A}) = \bigvee_{\langle\tilde{P};\sigma\rangle \rightarrow^* \langle\tilde{P}';\sigma'\rangle} Check(\tilde{P}', \sigma', \mathcal{A})$
$Check(\tilde{P}, \sigma, \diamondsuit\mathcal{A}) = \bigvee_{\langle\tilde{P};\sigma\rangle \downarrow^* \langle\tilde{P}';\sigma\rangle} Check(\tilde{P}', \sigma, \mathcal{A})$
$Check(\tilde{P}, \sigma, \mathcal{A}@n) = Check(n[\tilde{P}], \sigma, \mathcal{A})$

An expression $Check(\tilde{P}, \sigma, \mathcal{A})$ is said to be *normal* if and only if the closure $\langle\tilde{P}; \sigma\rangle$ is normal, $\mathcal{A}$ is a closed formula, and $fn(\mathcal{A}) \cap (bn(\tilde{P}) \cup dom(\sigma)) = \varnothing$. Hence, for the model checking problem $P \models \mathcal{A}$ where $\mathcal{A}$ is a closed formula, the expression $Check(\mathcal{F}(P), \iota, \mathcal{A})$ is normal and moreover we have:

Proposition 3.9 *The model checking algorithm described above preserves the normality of* $Check(\tilde{P}, \sigma, \mathcal{A})$.

Proposition 3.10 *For all processes P and closed formulas $\mathcal{A}$, we have $P \models \mathcal{A}$ if and only if* $Check(\mathcal{F}(P), \iota, \mathcal{A}) = \mathbf{T}$.

Theorem 3.11 *Model checking the ambient calculus and logic of this paper is decidable in PSPACE.*

Proof To test for a given process P and formula $\mathcal{A}$ whether $P \models \mathcal{A}$ we simply compute the value of $Check(\mathcal{F}(P), \iota, \mathcal{A})$. The only problem is to implement *Check* in such a way that it works in polynomial space.

In the case of $\mathbf{T}, \mathbf{0}, n[\mathcal{A}], \mathcal{A}@n, \neg\mathcal{A}$, the algorithm can directly check whether the respective conditions hold. In the case of $\mathcal{A} \vee \mathcal{B}, \mathcal{A} \mid \mathcal{B}, \exists x.\mathcal{A}, \Diamond\mathcal{A}, \diamondsuit\mathcal{A}$, we have to be more careful about the space used to compute the value of disjunctions. In a loop we iteratively compute the value of each disjunct, reusing the same space in every iteration. In the case of $\Diamond\mathcal{A}$ the subroutine computing $\bigvee_{\langle\tilde{P};\sigma\rangle\to^*\langle\tilde{P}';\sigma'\rangle} \mathit{Check}(\tilde{P}',\sigma',\mathcal{A})$ could look as follows.

$\mathit{result} \leftarrow \mathbf{F}$
for all $\langle\tilde{P}';\sigma'\rangle$ such that $\langle\tilde{P};\sigma\rangle \to^* \langle\tilde{P}';\sigma'\rangle$
  if $\mathit{Check}(\tilde{P}',\sigma',\mathcal{A}) = \mathbf{T}$ then $\mathit{result} \leftarrow \mathbf{T}$
return(result)

By Propositions 3.6 and 3.8, every iteration requires only polynomial space. The cases of $\mathcal{A}\vee\mathcal{B}, \mathcal{A} \mid \mathcal{B}, \exists x.\mathcal{A}, \diamondsuit\mathcal{A}$ are similar. Thus, the space $S(k,|\tilde{P}|+|\sigma|)$ used by the algorithm to compute $\mathit{Check}(\tilde{P},\sigma,\mathcal{A})$ for formulas $\mathcal{A}$ of depth not exceeding k satisfies the inequality

$$S(k+1,|\tilde{P}|+|\sigma|) \leq S(k,|\tilde{P}|+c+|\sigma|) + p(|\tilde{P}|+|\sigma|)$$

for some constant c and some polynomial p (the constant c comes from the fact that in the case of $\mathcal{A} = \mathcal{B}@n$ the size of $n[\tilde{P}]$ is greater than the size of $\tilde{P}$; the polynomial p estimates the space needed for testing reachability etc). Therefore, $S(k,|\tilde{P}|+|\sigma|) \leq k\cdot p(|\tilde{P}|+k\cdot c+|\sigma|)$.

Finally, the fact that $\mathcal{F}(P)$ is polynomial in the size of P and the statement of Proposition 3.10 complete the proof. ■

4 Complexity Lower Bounds

Below we present lower bounds on the space complexity of model checking our process calculus against our modal logic, and also for two significant fragments.

The results given here are based on known results about the complexity of decision problems for Quantified Boolean Formulas (QBF). We can assume without loss of generality that these Boolean formulas are in prenex and conjunctive normal form. The alternation depth of a formula is the number of alternations between existential and universal quantifiers in its prenex quantification.

Those known results are: (1) deciding the validity problem for a closed quantified Boolean formula φ is PSPACE-complete; (2) deciding the validity problem for a closed quantified Boolean formula φ of alternation depth k whose outermost quantifier is $\exists$ is Σ_k^P-complete (Stockmeyer 1976), where Σ_k^P denotes the k-th level of the polynomial-time hierarchy. In particular, Σ_0^P = P and Σ_1^P = NP.

We will use the following formula as a running example of a valid closed QBF formula:

$$\forall v_1.\exists v_2.\exists v_3.(v_1 \vee \overline{v_2} \vee v_3) \wedge (\overline{v_1} \vee v_2 \vee v_3) \wedge \overline{v_3}$$

4.1 The Full Calculus and Logic

We define an encoding of QBF formulas into ambient formulas. This encoding is then used to prove Theorem 4.2, that the complexity of model checking the ambient logic is PSPACE-hard.

In our encoding, we assume that the truth values tt and ff used in the definition of QBF satisfaction are distinct ambient calculus names.

We also use a derived operator for name equality in the ambient logic first defined by Cardelli and Gordon (2000b):

$$\eta = \mu \triangleq \eta[\mathbf{T}]@\mu$$

Then $\mathbf{0} \models m = n$ if and only if the names m and n are equal. We encode the $\forall$ and $\exists$ quantifiers over truth values as follows.

$$\begin{aligned} \forall x \in \{ff, tt\}.\mathcal{A} &\triangleq \forall x.(x = ff \vee x = tt) \Rightarrow \mathcal{A} \\ \exists x \in \{ff, tt\}.\mathcal{A} &\triangleq \exists x.(x = ff \vee x = tt) \wedge \mathcal{A} \end{aligned}$$

Encoding QBF Formulas as Ambient Logic Formulas:

$[\![v]\!] \triangleq (v = tt)$
$[\![\overline{v}]\!] \triangleq (v = ff)$
$[\![\ell_1 \vee \cdots \vee \ell_k]\!] \triangleq [\![\ell_1]\!] \vee \cdots \vee [\![\ell_k]\!]$
$[\![C_1 \wedge \cdots \wedge C_k]\!] \triangleq [\![C_1]\!] \wedge \cdots \wedge [\![C_k]\!]$
$[\![\forall v.\varphi]\!] \triangleq \forall v \in \{ff, tt\}.[\![\varphi]\!]$
$[\![\exists v.\varphi]\!] \triangleq \exists v \in \{ff, tt\}.[\![\varphi]\!]$

The following properties are proved in the appendix. The proof of Lemma 4.1 is by induction on the number of variables quantified in φ.

Lemma 4.1 *Consider a closed quantified boolean formula φ and its encoding $[\![\varphi]\!]$ in the ambient logic. The formula φ is valid if and only if the model checking problem $\mathbf{0} \models [\![\varphi]\!]$ holds.*

Theorem 4.2 *The complexity of model checking the full logic (including name quantification) is PSPACE-hard.*

Proof Straightforward from Lemma 4.1 since for the fixed ambient process $\mathbf{0}$ solving the model checking problem $\mathbf{0} \models \varphi$ is PSPACE-hard. So in fact the expression complexity, that is, the complexity of checking formulas against a fixed process, is PSPACE-hard. ■

The theorem above holds for any fragment of the logic including boolean connectives, name quantification, and the location and location adjunct modalities, and for any fragment of the calculus including ambients. This might suggest that the complexity of the model checking problem comes from the quantification in the logic. Below we show that it is not the case: the problem remains so complex even if we remove quantification from the logic and communication or mobility from the calculus. This suggests there is little chance of finding interesting fragments of the calculus and the logic that would admit a faster model checking algorithm.

4.2 Mobile Ambients Without I/O, No Quantifiers

In this section, we study the complexity of the model checking problem for the fragment of the ambient calculus without I/O and the fragment of the logic without quantification.

For every QBF variable, v, we assume that v, v' and v'' are distinct ambient calculus names.

Encoding QBF Formulas as Ambient Processes and Formulas:

$$[\![v]\!] = v[pos[\mathbf{0}] \mid v'[\mathbf{0}]] \mid \mathbf{T}$$
$$[\![\overline{v}]\!] = v[neg[\mathbf{0}] \mid v'[\mathbf{0}]] \mid \mathbf{T}$$
$$[\![\ell_1 \vee \cdots \vee \ell_k]\!] = [\![\ell_1]\!] \vee \cdots \vee [\![\ell_k]\!]$$

$$[\![C_1 \wedge \cdots \wedge C_k]\!] = (end[\mathbf{0}], [\![C_1]\!] \wedge \cdots \wedge [\![C_k]\!])$$

$$[\![\forall v.\varphi]\!] = (v'[in\, v.n[out\, v'.out\, v.P]], \Box((n[\mathbf{T}] \mid \mathbf{T}) \Rightarrow \mathcal{A})) \quad \text{where } (n[P], \mathcal{A}) = [\![\varphi]\!]$$
$$[\![\exists v.\varphi]\!] = (v'[in\, v.n[out\, v'.out\, v.P]], \Diamond((n[\mathbf{T}] \mid \mathbf{T}) \wedge \mathcal{A})) \quad \text{where } (n[P], \mathcal{A}) = [\![\varphi]\!]$$

$$enc(\varphi) = (v_1[pos[\mathbf{0}]] \mid v_1[neg[\mathbf{0}]] \mid \cdots \mid v_n[pos[\mathbf{0}]] \mid v_n[neg[\mathbf{0}]] \mid P, \mathcal{A})$$
$$\text{where } (P, \mathcal{A}) = [\![\varphi]\!] \text{ and } \varphi = Q_1 v_1. \ldots .Q_n v_n.C_1 \wedge \cdots \wedge C_k$$
$$\text{where each } Q_i \in \{\exists, \forall\}.$$

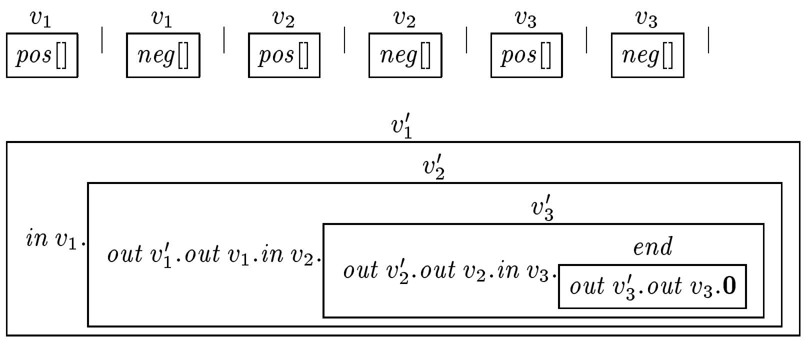
Brief explanation. In the encoding $enc(\varphi)$ above, the parallel composition $v_1[pos[\mathbf{0}]] \mid \ldots \mid v_n[neg[\mathbf{0}]]$ represents the sequence $v_1, \ldots v_n$ of (uninstantiated) boolean variables and P is a process that instantiates them. An instantiated variable v_i is represented by a subprocess $v_i[pos[\mathbf{0}] \mid v_i'[\mathbf{0}]] \mid v_i[neg[\mathbf{0}]]$ (if its value is tt) or $v_i[pos[\mathbf{0}]] \mid v_i[neg[\mathbf{0}] \mid v_i'[\mathbf{0}]]$ (if its value is $f\!f$). The process P first instantiates v_1 by choosing one of the ambients $v_1[pos[\mathbf{0}]]$ or $v_1[neg[\mathbf{0}]]$ nondeterministically, going inside it, leaving the token $v_1'[\mathbf{0}]$ inside the chosen ambient and then returning to the top level. It then iteratively instantiates the variables $v_2, \ldots, v_n$ in the same way. The formula $n[\mathbf{T}] \mid \mathbf{T}$ in the context of the encoding for a quantified variable v_i above (where n is v_{i+1} or end for $i = n$) expresses that the instantiation of v_i has finished but that the instantiation of n has yet to start; thus $\Box(n[\mathbf{T}] \mid \mathbf{T} \ldots)$ and $\Diamond(n[\mathbf{T}] \mid \mathbf{T} \ldots)$ express, respectively, universal and existential quantifications over instantiations of v_i.

In the case where φ is the formula defined previously as an example, one would obtain $enc(\varphi) = (P, \mathcal{A})$, where P is the process depicted in Figure 1(a) and where the formula $\mathcal{A}$ is of the form:

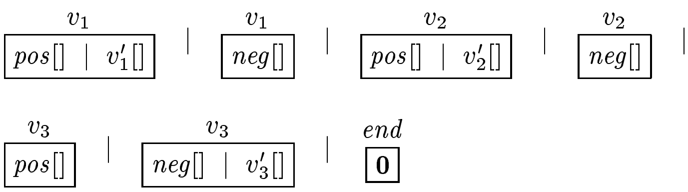
$$\Box((v_2'[\mathbf{T}] \mid \mathbf{T}) \Rightarrow \Diamond((v_3'[\mathbf{T}] \mid \mathbf{T}) \wedge \Diamond((end[\mathbf{T}] \mid \mathbf{T}) \wedge \mathcal{B})))$$

where $\mathcal{B}$ is the formula given by $[\![v_1 \vee \overline{v_2} \vee v_3]\!] \wedge [\![\overline{v_1} \vee v_2 \vee v_3]\!] \wedge [\![\overline{v_3}]\!]$.

More detailed explanation. We explain this encoding with reference to the ambient process depicted in Figure 1(a). The ambients whose names range over v_i describe an interpretation for the Boolean variables v_i whereas the ambients



(a) The process P in $enc(\varphi) = (P, \mathcal{A})$



(b) The irreducible process for the interpretation $v_1 \mapsto tt, v_2 \mapsto tt, v_3 \mapsto \mathit{ff}$

Figure 1: Encoding for mobile ambients without I/O, no quantifiers

named v_i' are the "material" to extend this interpretation. In the initial ambient, the ambients v_i encode the empty interpretation and the material is in an ambient named v_1' marking the fact that v_1 is the first variable to treat. The first step of reduction will move the ambient v_1' non-deterministically either inside $v_1[pos[]]$ (the Boolean variable v_1 takes the value tt) or inside $v_1[neg[]]$ (the Boolean variable v_1 takes the value ff). The next two steps of reduction are deterministic. They aim to leave a mark in one of the ambients v_1 according to the first non-deterministic choice and to reach a situation in which the Boolean variable v_2 is considered. For instance, if the first choice was to instantiate v_1 with tt then, one would obtain a parallel composition of $v_1[pos[] \mid v_1'[]]$ and $v_1[neg[]]$. The ambients named v_2, v_3 are kept unchanged and the ambient containing the rest of the interpretation would be of the form $v_2'[in\, v_2.v_3'[Q]]$ where Q is the internal of v_3' in the initial process. This computation, consisting of one non-deterministic step followed by two deterministic ones, can be carried on for the variables v_2 and v_3. Then, when no more reduction step is possible, the resulting process is a parallel composition of the empty ambient $end[]$ and, for each i, of $v_i[n[] \mid v_i'[]]$ and $v_i[n'[]]$ where n, n' are distinct elements from $\{pos, neg\}$. For instance, the irreducible process given in Figure 1(b) represents the interpretation $v_1 \mapsto tt, v_2 \mapsto tt\ , v_3 \mapsto f\!f$.

We said that the ambient processes encode interpretations. The Boolean formula itself is encoded in the ambient formula $\mathcal{A}$. Once no more reduction step is possible on the ambient process, this latter represents an interpretation whose domain is the set of all variables in φ: this interpretation is given by the places where the marks v_i' have been put. It is easy with an ambient formula to test whether this interpretation renders true the quantifier-free part of φ. This role is played by the ambient formula $\mathcal{B}$ whereas the remaining part of $\mathcal{A}$ aims to encode the quantifiers of φ.

Let us first consider the outermost quantifier $\forall v_1$ in φ: this quantification stands for "for all possible interpretations of the variable v_1". We have described above the mechanism for the instantiation of the Boolean variable v_1 in the ambient process. It consists of first a non-deterministic step, then two deterministic steps. Whatever the first step is, those three steps lead to a situation where the ambient process is of the form $R \mid v_2'[R']$. It should be noticed that those two processes (one for each possibility of the first step) are the only processes of this form reachable from the initial process. Therefore, the statement "for all possible interpretations of the variable v_1" can be translated as "for all processes of the form $R \mid v_2'[R']$ reachable from the initial process". This rephrased statement can be expressed in the ambient logic as $\Box((v_2'[\mathbf{T}] \mid \mathbf{T}) \Rightarrow \dots)$.

A dual reasoning can be applied then for $\exists v_2$, the following quantification of the formula φ. In that case, the statement "there exists an interpretation for the variable v_2" is translated into "there exists an ambient process of the form $T \mid v_3'[T']$ reachable from the current process". This current process is one of the two processes after the instantiation of the variable v_2, that is of the form $S \mid v_3'[S']$. This statement can be expressed by means of the ambient logic by the formula $\Diamond((v_3'[\mathbf{T}] \mid \mathbf{T}) \wedge \dots)$. Finally, the quantification $\exists v_3$ is expressed by $\Diamond((end[\mathbf{T}] \mid \mathbf{T}) \wedge \dots)$.

Lemma 4.3 *Assume φ is a closed quantified Boolean formula, and $(P, \mathcal{A}) = enc(\varphi)$. Then $P \models \mathcal{A}$ if and only if φ is valid.*

Theorem 4.4 *The complexity of model checking mobile ambients without I/O against the quantifier-free logic is PSPACE-hard.*

Proof Straightforward from the PSPACE-completeness of the validity for QBF and from Lemma 4.3, taking into account that for $enc(\varphi) = (P, \mathcal{A})$, both P and $\mathcal{A}$ are of polynomial size with respect to φ. ■

4.3 Immobile Ambients With I/O, No Quantifiers

In this section, we study the complexity of the model checking problem for the fragment of the ambient calculus without action prefix.

We consider fixed names *end*, C, and D. For any QBF variable ambient name v'_i, let

$$Inst(v'_i) \triangleq v'_i[\mathbf{T}] \mid \mathbf{T} \qquad Inst^+(v'_i) \triangleq v'_i[v''_i[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$$

and for the name *end*,

$$Inst(end) \triangleq end[\mathbf{T}] \mid \mathbf{T} \qquad Inst^+(end) \triangleq end[end'[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$$

Encoding QBF Formulas as Ambient Processes and Formulas:

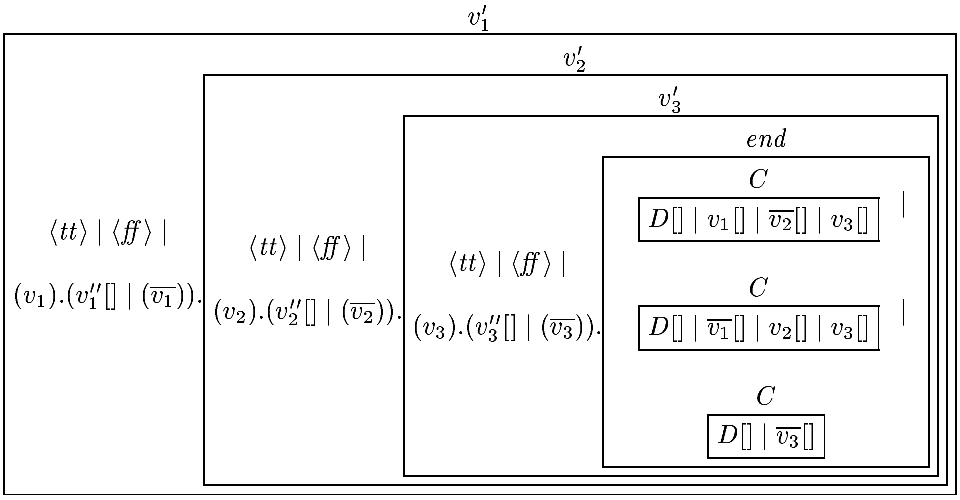
$[\![v]\!] = v[\mathbf{0}]$
$[\![\overline{v}]\!] = \overline{v}[\mathbf{0}]$
$[\![\ell_1 \vee \ldots \vee \ell_k]\!] = D[\mathbf{0}] \mid [\![\ell_1]\!] \mid \ldots \mid [\![\ell_k]\!]$

$$\begin{aligned}
enc(C_1 \wedge \ldots \wedge C_k) = {} & (end[C[\,[\![C_1]\!]\,] \mid \ldots \mid C[\,[\![C_k]\!]\,]], \\
& \quad ❏((D[\mathbf{0}] \mid \mathbf{T}) \Rightarrow (tt[\mathbf{0}] \mid \mathbf{T}))) \\
enc(\exists v.\varphi) = {} & (v'[\langle tt\rangle \mid \langle f\!f\rangle \mid (v).(v''[] \mid (\overline{v}).n[P])], \\
& \mathbf{T} \mid v'[\Diamond(\ (Inst(n) \wedge \neg Inst^+(n))\ \wedge \mathcal{A}\ )]) \\
& \quad \text{where } enc(\varphi) = (n[P], \mathcal{A}) \\
enc(\forall v.\varphi) = {} & (v'[\langle tt\rangle \mid \langle f\!f\rangle \mid (v).(v''[] \mid (\overline{v}).n[P])], \\
& \mathbf{T} \mid v'[\Box(\ (Inst(n) \wedge \neg Inst^+(n))\ \Rightarrow \mathcal{A}\ )]) \\
& \quad \text{where } enc(\varphi) = (n[P], \mathcal{A})
\end{aligned}$$

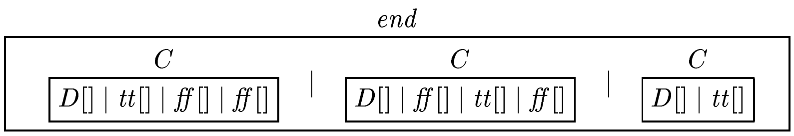
Brief explanation. The idea of the encoding here is quite similar to that from the previous section. A boolean variable v is represented here by two ambients $v[]$ and $\overline{v}[]$, which after the instantiation are named $tt[]$ and $f\!f[]$. We exploit here the nondeterminism of communication: the variable v reads either the message $\langle tt\rangle$ or $\langle f\!f\rangle$; then its dual $\overline{v}$ has to read the other one. The names v'_i and v''_i (similar to v'_i in the previous section) are used for distinguishing the moment when the variable v_i is already instantiated but v_{i+1} is not. The formula ❏$((D[\mathbf{0}] \mid \mathbf{T}) \Rightarrow (tt[\mathbf{0}] \mid \mathbf{T}))$ requires that in the final state, each ambient representing a clause (that is, an ambient containing $D[\mathbf{0}]$) contains at least one true literal (that is, an ambient $tt[\mathbf{0}]$).

For the formula φ used in our example, one would have $enc(\varphi) = (P, \mathcal{A})$, where P is depicted in Figure 2(a).

More detailed explanation. The key idea of this encoding is to use (reductions of) communications for performing the instantiation of the quantifier-free part of φ with respect to some interpretation. Therefore, the quantifier-free formula $C_1 \wedge \ldots \wedge C_k$ is encoded in the ambient process itself, inside an ambient named *end*. For instance, in Figure 2(a) for our example, the ambient $end[C[D[] \mid v_1[] \mid \overline{v_2}[] \mid v_3[]] \mid C[D[] \mid \overline{v_1}[] \mid v_2[] \mid v_3[]] \mid C[D[] \mid \overline{v_3}[]]]$ encodes the quantifier-free part of φ: the ambient *end* contains a sub-ambient called C for each clause C_i in φ and the ambient corresponding to C_i contains an ambient $\ell_j[]$ for each literal ℓ_j from C_i.



(a) The process P in $enc(\varphi) = (P, \mathcal{A})$



(b) The process representing the instantiation of $C_1 \wedge C_2 \wedge C_3$ by $v_1 \mapsto tt, v_2 \mapsto tt, v_3 \mapsto ff$

Figure 2: Encoding for immobile ambients with I/O, no quantifiers

Starting from P described in Figure 2(a), let us inspect the behaviour of processes through reductions. Two reductions can be performed on P: one establishes a communication between $\langle tt \rangle$ and (v_1) and the other one between $\langle ff \rangle$ and (v_1). Once this reduction step is performed the name v_1 has been replaced by either tt or ff uniformly at every position and in particular in the ambient named end. Hence, the first step of computation is non-deterministic and instantiates the literal v_1. It has also a side-effect: it reveals an ambient process $v_1''[]$ within the ambient v_1'; this process is a marker for the control of computations. Its precise role will be explained later on. The second step is deterministic: for each first step, only one second step is possible. This second step aims to instantiate the literal $\overline{v_1}$ according to the instantiation of v_1. Indeed, if the first communication has consumed the output $\langle tt \rangle$ then for the second one only the output $\langle ff \rangle$ remains and vice-versa. So, after the second step, the name $\overline{v_1}$ is globally replaced by a Boolean value. Moreover, at this point there are no more actions prefixing the ambient named v_2' and so this ambient can be now reduced using the rules (Red Par) and (Red Amb). The next reduction steps are performed in a similar way: a non-deterministic step follows by a deterministic one. This leads finally to replace in the ambient *end* all the names corresponding to literals by Boolean values *tt* and *ff*. As an example, in Figure 2(b), we have depicted the ambient *end* once the reductions corresponding to the interpretation $\mathcal{M} = v_1 \mapsto tt, v_2 \mapsto tt, v_3 \mapsto ff$ have been performed.

Now, using an ambient formula it is not difficult to test whether the interpretation induced from the process in Figure 2(b) is a model for $C_1 \wedge C_2 \wedge C_3$: as $C_1 \wedge C_2 \wedge C_3$ is in conjunctive normal form, $\mathcal{M}$ is a model for it if and only if $\mathcal{M}$ renders at least one literal true in every clause C_i. According to the way reductions are performed and correspond to instantiations, this is equivalent to the claim that in the process from Figure 2(b), every ambient named C contains a sub-ambient $tt[]$. This can be tested with the formula $\mathcal{B} = ❑((D[\mathbf{0}] \mid \mathbf{T}) \Rightarrow (tt[\mathbf{0}] \mid \mathbf{T}))$, which is exactly the formula given by $enc(C_1 \wedge C_2 \wedge C_3)$.

In the encoding $enc(\varphi) = (P, \mathcal{A})$, one part of $\mathcal{A}$ aims to test whether the interpretation corresponding to the reductions is a model of φ. The other part of $\mathcal{A}$ is used to encode the quantification of φ. Let us illustrate on our example the ideas of this encoding: for the formula φ from our example, the formula $\mathcal{A}$ is equal to

$$\begin{array}{l}\mathbf{T} \mid v_1'[\Box(\ (Inst(v_2') \wedge \neg Inst^+(v_2'))\ \Rightarrow \\ \quad (\mathbf{T} \mid v_2'[\Diamond(\ (Inst(v_3') \wedge \neg Inst^+(v_3'))\ \wedge \\ \qquad (\mathbf{T} \mid v_3'[\Diamond(Inst(end) \wedge \neg Inst^+(end)\ \ \wedge \mathcal{B})])\ )])\ )]\end{array}$$

where $\mathcal{B}$ is the result of the encoding of the quantifier-free part of φ. For the variable v_i, the intuitive reading of $Inst(v_i')$ is "the next variable to consider is v_i", that is, the instantiation of the variable v_{i-1} has been completed. The reading of $Inst^+(v_i')$ is "the variable v_i has been partially treated", that is, the instantiation has been performed for the positive literal v_i. For the ambient name *end*, $Inst(end)$ refers to the completion of the instantiation of the variable v_n.

The first quantification $\forall v_1$ stands for "for all possible interpretations of the variable v_1" and the part of φ related with this quantification is

$$\mathbf{T} \mid v_1'[\Box(\ (Inst(v_2') \wedge \neg Inst^+(v_2'))\ \Rightarrow \ldots)]$$

This formula is model checked against the process P given in Figure 2(a). As $P \equiv \mathbf{0} \mid P$, the model checking problem is reduced to checking the interior of v_1' against the sub-formula of the form $\Box\mathcal{A}_1$: all processes reachable from the interior of v_1' must satisfy $\mathcal{A}_1$. Let us have a look at the form of those reachable processes: the interior of v_1' is itself reachable as well as the two processes corresponding to the instantiation of the literal v_1 (reachable in one step). In those processes v_1 has been replaced by a Boolean value but none of them satisfies $v_2'[\mathbf{T}] \mid \mathbf{T}$, that is, $Inst(v_2')$. Now, the processes reachable in two steps or more indeed satisfy the formula $Inst(v_2')$; but the ones reachable in exactly two steps can be distinguished from the others since these former are the only ones which do not satisfy $v_2'[v_2''[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$, that is, $Inst^+(v_2')$. Indeed, steps beyond the second one reveal the marker $v_2''[]$ inside the ambient v_2'. We have already mentioned the fact that the two steps of computation correspond exactly to the complete treatment of the variable v_1 which is the intended meaning of $Inst(v_2') \wedge \neg Inst^+(v_2')$. Therefore, model checking continues by checking the two processes (the second step of computation being deterministic), defined as the interior of v_1' in which the literals v_1 and $\overline{v_1}$ have been replaced by Boolean values, against the formula

$$\mathbf{T} \mid v_2'[\Diamond(\ (Inst(v_3') \wedge \neg Inst^+(v_3'))\ \ \wedge ...)]$$

from the encoding of the quantification $\exists v_2$. It stands for "there exists an interpretation for v_2". The process that is checked against this formula is of the form $v_1''[] \mid v_2'[R]$. Therefore, it amounts to check whether the process R, which is the interior of v_2' in which names $v_1, \overline{v_1}$ have been replaced with Boolean values, is a model for the sub-formula of the form $\Diamond \mathcal{A}_2$. Equivalently, there must exist a process reachable from R which satisfies $\mathcal{A}_2$. Let us inspect the processes reachable from R. Of course, R itself is reachable as well as the two processes reachable in one step of computation performing the instantiation for the literal v_2. None of these processes satisfies the formula $v_3'[\mathbf{T}] \mid \mathbf{T}$, that is, $Inst(v_3')$. Processes that are obtained with two steps or more from R do satisfy $Inst(v_3')$ but only those obtained by strictly more than two steps reveal the marker $v_3''[]$ inside v_3' and thus, satisfy $v_3'[v_3''[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$, that is $Inst^+(v_3')$. Those computations from R of exactly two steps correspond to the complete treatment of the variable v_2 and satisfy $Inst(v_3') \wedge \neg(Inst^+(v_3'))$. So, model checking carries on by checking that one of these two processes reachable from R in two steps and defined as the interior of v_2 in which the literals v_1, $\overline{v_1}$, v_2, $\overline{v_2}$ have been replaced by Boolean values, is a model for the remaining part of the encoding of the formula.

Finally, the quantification $\exists v_3$ is encoded as

$$\mathbf{T} \mid v_3'[\Diamond(\ ((\mathbf{T} \mid end[\mathbf{T}]) \wedge \neg(\mathbf{T} \mid end[end'[\mathbf{T}] \mid \mathbf{T}]))\ \ \wedge ...)]$$

and its treatment is similar to that of $\exists v_2$. It leads to model checking the process named *end* given in Figure 2(b) against the formula $\mathcal{B}$.

Lemma 4.5 *Assume φ is a closed quantified Boolean formula, and $(P, \mathcal{A}) = enc(\varphi)$. Then $P \models \mathcal{A}$ if and only if φ is valid.*

Theorem 4.6 *The complexity of model checking immobile ambients with I/O against the quantifier-free logic is PSPACE-hard.*

Proof This follows from the PSPACE-completeness of validity for QBF, from Lemma 4.5 taking into account that for $enc(\varphi) = (P, \mathcal{A})$, both P and $\mathcal{A}$ are of polynomial size with respect to φ. ■

We can strengthen this result by slightly modifying our encoding. Our previous encoding is based on an individual treatment for the variables in the quantification. The improved encoding will be based on the alternation of quantifiers: roughly, $\exists v_2 \exists v_3$ can be grouped together by saying that "there exists an interpretation for v_2 and v_3". As far as the previous encoding is concerned, the ambient formula resulting from the encoding of $\exists v_2 \exists v_3$ will perform two successive tests for reachability; this can be modified in such a way that only one test of reachability is performed. This will imply for the new encoding that the markers used to control the model checking (namely, the ambients v') will no longer be associated with the variables but with the alternation of quantifiers. Those ambient names will range over a_i where i is an integer. We define for those a_i's:

$$Inst(a_i) \triangleq a_i[\mathbf{T}] \mid \mathbf{T} \qquad\qquad Inst^+(a_i) \triangleq a_i[a_i[] \mid \mathbf{T}] \mid \mathbf{T}$$

The Revised Encoding:

$enc(\forall v.\varphi) = enc(\forall v.\varphi, 1)$
$enc(\exists v.\varphi) = enc(\exists v.\varphi, 1)$

$$\begin{aligned}
enc(\forall v.\varphi, i) = (&a_i[\langle tt\rangle \mid \langle f\!f\rangle \mid (v).(a_i[] \mid (\overline{v})).P, \\
&\mathbf{T} \mid a_i[\Box(\; Inst(a_{i+1}) \wedge Inst^+(a_{i+1}) \;\Rightarrow \mathcal{A}\;)]) \\
&\quad \text{where } enc_\forall(\varphi, i) = (P, \mathcal{A}) \\
enc(\exists v.\varphi, i) = (&a_i[\langle tt\rangle \mid \langle f\!f\rangle \mid (v).(a_i[] \mid (\overline{v})).P, \\
&\mathbf{T} \mid a_i[\Diamond(\; Inst(a_{i+1}) \wedge Inst^+(a_{i+1}) \;\wedge \mathcal{A}\;)]) \\
&\quad \text{where } enc_\exists(\varphi, i) = (P, \mathcal{A})
\end{aligned}$$

$enc_\forall(\exists v.\varphi, i) = enc(\exists v.\varphi, i+1)$
$enc_\forall(\forall v.\varphi, i) = (\langle tt\rangle \mid \langle f\!f\rangle \mid (v).(\overline{v}).P, \mathcal{A}) \quad$ where $enc_\forall(\varphi, i) = (P, A)$

$enc_\exists(\forall v.\varphi) = enc(\forall v.\varphi, i+1)$
$enc_\exists(\forall v.\varphi, i) = (\langle tt\rangle \mid \langle f\!f\rangle \mid (v).(\overline{v}).P, \mathcal{A}) \quad$ where $enc_\exists(\varphi, i) = (P, A)$

$enc(C_1 \wedge \ldots \wedge C_k, i) = (a_i[C[\,[\![C_1]\!]\,] \mid \ldots \mid C[\,[\![C_k]\!]\,]], ❏((D[\mathbf{0}] \mid \mathbf{T}) \Rightarrow tt[\mathbf{0}] \mid \mathbf{T}))$

$[\![\ell_1 \vee \ldots \vee \ell_k]\!] = D[\mathbf{0}] \mid [\![\ell_1]\!] \mid \ldots \mid [\![\ell_k]\!]$
$[\![v]\!] = v[]$
$[\![\overline{v}]\!] = \overline{v}[]$

The statement of Lemma 4.5 still holds for this new encoding. Furthermore, in the encoding $(P, \mathcal{A})$ of the Boolean formula φ, the ambient logic formula $\mathcal{A}$ depends only on the alternation depth and the outermost quantifier of φ; for any two Boolean formulas φ, φ' having the same alternation depth k and the same outermost quantifier Q, if $enc(\varphi) = (P, \mathcal{A})$ and $enc(\varphi') = (P', \mathcal{A}')$ then $\mathcal{A} = \mathcal{A}'$.

Theorem 4.7 *For every integer k there exists a formula $\mathcal{A}_k^\exists$ such that the complexity of model checking processes against $\mathcal{A}_k^\exists$ is Σ_k^P-hard.*

Proof Let $\mathcal{A}_k^{\exists}$ be the formula such that for any closed quantified Boolean formula φ of alternation depth k whose outermost quantifier is $\exists$, $enc(\varphi) = (P_\varphi, \mathcal{A}_k^{\exists})$. Due to the remark above, we know that this formula exists and furthermore, is of size polynomial in k.

Now, by Lemma 4.5, every instance of the validity problem for a closed quantified Boolean formula φ of alternation depth k whose outermost quantifier is $\exists$ can be reduced to the model checking problem $P_\varphi \models \mathcal{A}_k^{\exists}$ for $enc(\varphi) = (P_\varphi, \mathcal{A}_k^{\exists})$. Thus, since the size of P_φ is polynomial in the size of φ, the theorem follows. ∎

5 Conclusion

We show in this paper that the model checking problem of the replication-free ambient calculus with public names against the ambient logic without composition-adjunct is PSPACE-complete. In order to prove this complexity bound, we have proposed a new representation for processes, called closures, that prevents the exponential blow-up of the size. We use this representation together with a new algorithm to prove the PSPACE upper bound.

We also have shown that there is little chance to find polynomial algorithms for interesting subproblems: model checking remains PSPACE-hard even for quite simple fragments of the calculus and the logic.

Possible directions for future work include investigations of the model checking problem for extensions of the logic and the calculus. Recently, Cardelli and Gordon (2001) have presented an extended version of the logic that allows reasoning about restricted names; it seems that there is no difficulty in extending our algorithm to deal with name restriction.

References

Cardelli, L. and G. Ghelli (2001). A query language based on the ambient logic. In *Proceedings ESOP'01*, Volume 2028 of *Lecture Notes in Computer Science*, pp. 1–22. Springer.

Cardelli, L. and A. D. Gordon (1999a). Types for mobile ambients. In *Proceedings POPL'99*, pp. 79–92. ACM.

Cardelli, L. and A. D. Gordon (1999b). Equational properties of mobile ambients. In *Proceedings FoSSaCS'99*, Volume 1578 of *Lecture Notes in Computer Science*, pp. 212–226. Springer. An extended version appears as Microsoft Research Technical Report MSR–TR–99–11, April 1999.

Cardelli, L. and A. D. Gordon (2000a). Mobile ambients. *Theoretical Computer Science 240*, 177–213.

Cardelli, L. and A. D. Gordon (2000b). Anytime, anywhere: Modal logics for mobile ambients. In *Proceedings POPL'00*, pp. 365–377. ACM.

Cardelli, L. and A. Gordon (2001). Logical properties of name restriction. In *Proceedings TLCA'01*. Springer.

Kupferman, O., M. Y. Vardi, and P. Wolper (2000). An automata-theoretic approach to branching-time model checking. *Journal of the ACM 47*(2), 312–360.

Lloyd, J. (1987). Foundations of Logic Programming. *Springer-Verlag.*

Sangiorgi, D. (2001). Extensionality and intensionality of the ambient logics. In *Proceedings POPL'01*, pp. 4–13. ACM.

Savitch, W. (1970). Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences 4*(2), 177–192.

Stockmeyer, L. J. (1976). The polynomial-time hierarchy. *Theoretical Computer Science 3*(1), 1–22.

A Correctness Proofs

This appendix contains proofs of results stated in Section 3.

A.1 Proof of Proposition 3.1

Proposition 3.1 concerns the relationship between normal closures and structural congruence. In this appendix we develop enough facts about closures and structural congruence to prove it.

We begin with a proposition that normality is preserved by decomposition with ambient or parallel composition.

Proposition A.1

- *$\langle \tilde{P}; \sigma \rangle$ and $\langle \tilde{Q}; \sigma \rangle$ are normal and $fn(\tilde{P}) \cap bn(\tilde{Q}) = bn(\tilde{P}) \cap fn(\tilde{Q}) = bn(\tilde{P}) \cap bn(\tilde{Q}) = \varnothing$ iff $\langle \tilde{P} +\!\!+ \tilde{Q}; \sigma \rangle$ is normal.*

- *for all expressions M such that M does not contain names from $bn(\tilde{P})$, $\langle \{M[\tilde{P}]\}; \sigma \rangle$ is normal iff $\langle \tilde{P}; \sigma \rangle$ is normal.*

Proof For the first point: from right to left, it is straightforward from the definition of $\mathcal{U}$ that if $\mathcal{U}(\tilde{P} +\!\!+ \tilde{Q}, \sigma)$ is defined then both $\mathcal{U}(\tilde{P}, \sigma)$ and $\mathcal{U}(\tilde{Q}, \sigma)$ are so. As $fn(\tilde{P} +\!\!+ \tilde{Q}) = fn(\tilde{P}) \cup fn(\tilde{Q})$ and $bn(\tilde{P} +\!\!+ \tilde{Q}) = bn(\tilde{P}) \cup bn(\tilde{Q})$, if $bn(\tilde{P} +\!\!+ \tilde{Q}) \cap (fn(\tilde{P} +\!\!+ \tilde{Q}) \cup names(\sigma)) = \varnothing$ then $bn(\tilde{P}) \cap (fn(\tilde{P}) \cup names(\sigma)) = bn(\tilde{Q}) \cap (fn(\tilde{Q}) \cup names(\sigma)) = \varnothing$. If for $\tilde{P} +\!\!+ \tilde{Q}$ bound variables occur at most once within an input and offsets in the scope of an input are equal to 0, then it is so for $\tilde{P}$ and $\tilde{Q}$. The last condition for normality on sequential substitution is obvious. The three other conditions follow directly from the normality of $\langle \tilde{P} +\!\!+ \tilde{Q}; \sigma \rangle$. From left to right, the definition of $\mathcal{U}$ implies that if $\langle \tilde{P}; \sigma \rangle$ and $\langle \tilde{Q}; \sigma \rangle$ are defined then $\langle \tilde{P} +\!\!+ \tilde{Q}; \sigma \rangle$ is defined. Now, $fn(\tilde{P} +\!\!+ \tilde{Q}) \cap bn(\tilde{P} +\!\!+ \tilde{Q}) = (fn(\tilde{P}) \cup fn(\tilde{Q})) \cap (bn(\tilde{P}) \cup bn(\tilde{Q}))$. We have $fn(\tilde{P}) \cap bn(\tilde{Q}) = bn(\tilde{P}) \cap fn(\tilde{Q}) = \varnothing$ by assumption and $fn(\tilde{P}) \cap bn(\tilde{P}) = fn(\tilde{Q}) \cap bn(\tilde{Q}) = \varnothing$ as $\langle \tilde{P}; \sigma \rangle$ and $\langle \tilde{Q}; \sigma \rangle$ are normal. So, $fn(\tilde{P} +\!\!+ \tilde{Q}) \cap bn(\tilde{P} +\!\!+ \tilde{Q}) = \varnothing$. By normality of $\langle \tilde{P}; \sigma \rangle$ and $\langle \tilde{Q}; \sigma \rangle$, $names(\sigma) \cap bn(\tilde{R}) = \varnothing$ for $\tilde{R} \in \{\tilde{P}, \tilde{Q}\}$. So, $names(\sigma) \cap bn(\tilde{P} +\!\!+ \tilde{Q}) = \varnothing$. $\langle \tilde{P}; \sigma \rangle$ and $\langle \tilde{Q}; \sigma \rangle$ being normal and as by assumption $bn(\tilde{P}) \cap bn(\tilde{Q}) = \varnothing$, every input variable occurs at most once within an input in $\tilde{P} +\!\!+ \tilde{Q}$. The last conditions on offsets in the scope of an input and on sequential substitution is obvious.

For the second point: It is easy to see that $\mathcal{U}(\{M[\tilde{P}]\}, \sigma)$ is defined iff $\mathcal{U}(\tilde{P}, \sigma)$ is so. The set of names occurring free in M is exactly the set $fn(\{M[\mathbf{0}]\})$. Now, as $bn(\{M[\tilde{P}]\}) = bn(\tilde{P})$ and $fn(\{M[\tilde{P}]\}) = fn(\tilde{P}) \cup fn(\{M[\mathbf{0}]\})$, $fn(\{M[\tilde{P}]\}) \cap bn(\{M[\tilde{P}]\})$ is empty iff $fn(\tilde{P}) \cap bn(\tilde{P})$ is empty (taking into account the assumption that $bn(\tilde{P}) \cap fn(\{M[\mathbf{0}]\}) = \varnothing$) and $bn(\{M[\tilde{P}]\}) \cap names(\sigma) = bn(\tilde{P}) \cap names(\sigma) = \varnothing$. Finally, the last three statements are obvious to check. ■

In the proof of Proposition 3.1 we will have to show that some processes are equivalent if and only if some conditions hold. In particular, we will have to show that if these conditions do not hold, the processes are not equivalent.

Although it is relatively easy to prove equivalence of processes, it is not so easy to prove their inequivalence (which requires showing that no equivalence proof exists). We use Theorem A.2 and Propositions A.3–A.5 below as tools for proving inequivalences needed in Proposition 3.1.

Let us consider Σ the signature used to build processes from the ambient calculus with public names. The signature Σ contains an infinite number of constants used as names. It contains moreover $\mathbf{0}$ and ϵ as constant symbols, the capabilities *in*, *out*, *open* and $\langle\rangle$ as unary function symbols. Finally, the binary function symbols $|, [], ., ()$ belong to Σ.

Let us denote T_Σ the set of all terms over Σ. Any ambient process from the ambient calculus with public names can be written as a term over this vocabulary. And of course, some terms from T_Σ are not ambient process, as for instance, $\langle \mathbf{0} \mid \mathbf{0} \rangle$.

The set T_Σ induces a canonical algebra that we denote $\mathcal{T}_\Sigma$: the algebra $\mathcal{T}_\Sigma$ has for carrier the set T_Σ and each function symbols from Σ is interpreted syntactically in $\mathcal{T}_\Sigma$.

The structural congruence relation $\equiv$ defined in Section 2.1 over pairs of ambient processes can be viewed as a relation defined over $T_\Sigma \times T_\Sigma$. One should notice that the set of axioms defining $\equiv$ is a set of definite Horn clauses, and thus, $(\mathcal{T}_\Sigma, \equiv)$ is a Herbrand model for this set of axioms. Moreover, as we consider the least relation satisfying these axioms, the structure $(\mathcal{T}_\Sigma, \equiv)$ is the least Herbrand model for this set of axioms. This implies that two processes P, Q are structurally equivalent if and only if $P \equiv Q$ belongs to the least Herbrand model of these axioms.

Note that if $\equiv$ is not assumed to be the least relation satisfying the axioms but for instance the greatest one, then one would have $P \equiv Q$ whatever P, Q are.

The following theorem is a direct consequence of two well-known facts (Lloyd 1987), that (1) every model of a set of Horn clauses can be translated to a Herbrand model, and (2) that every Herbrand model contains the least Herbrand model. Essentially, the theorem says that anything that does not belong to some model cannot belong to the least model.

Theorem A.2 *Let $\mathcal{S}$ be a set of definite Horn clauses defining a relation symbol $\equiv$. Then for all algebras A, for all structures R defined over A and giving an interpretation for $\equiv$ such that $\mathsf{R} \models \mathcal{S}$,*

$$\mathsf{R} \models s \equiv t \textit{ if } (\mathcal{T}_\Sigma, \equiv) \models s \equiv t$$

That is, if there exists a structure R such that $\mathsf{R} \models \mathcal{S}$ and $\mathsf{R} \models s \not\equiv t$, then $(\mathcal{T}_\Sigma, \equiv) \models s \not\equiv t$.

Let us consider now the algebra $\hat{\mathsf{A}}$ defined over Σ; the carrier $D_{\hat{\mathsf{A}}}$ is the least set such that

- the constants from Σ except ϵ and $\mathbf{0}$ belong to $D_{\hat{\mathsf{A}}}$,
- the empty string and the empty multiset belong to $D_{\hat{\mathsf{A}}}$,

- for any $d_1, d_2 \in D_{\hat{\mathsf{A}}}$, the items $in\ d_1$, $out\ d_1$, $open\ d_1$, $\langle d_1 \rangle$, $(d_1)d_2$ and $d_1[d_2]$ belong to $D_{\hat{\mathsf{A}}}$,
- for any $d_1, \ldots, d_n \in D_{\hat{\mathsf{A}}}$, the string $d_1 \ldots d_n$ and the multiset $\{d_1, \ldots, d_n\}$ belong to $D_{\hat{\mathsf{A}}}$.

The function symbols from Σ are interpreted in $\hat{\mathsf{A}}$ as follows.

- The constants from Σ except ϵ and $\mathbf{0}$ are interpreted syntactically.
- The constants ϵ and $\mathbf{0}$ are interpreted respectively as the empty string and as the empty multiset.
- The function symbols in, out, $open$, $\langle\rangle$, $()$ and $[]$ are interpreted syntactically.
- For the function symbol $.$: $d_1.d_2$ is the string obtained by concatenation of d_1 and d_2 if both d_1 and d_2 are strings. Otherwise, elements from $\{d_1, d_2\}$ that are not strings are transformed into a string of length one and then, the concatenation is performed.
- For the function symbol $|$: $d_1 \mid d_2$ is the multiset obtained by union of d_1 and d_2 if both d_1 and d_2 are multisets. Otherwise, elements from $\{d_1, d_2\}$ that are not multisets are transformed into a singleton multiset and then, the union is performed.

The algebra $\hat{\mathsf{A}}$ is extended into a structure $\hat{\mathsf{R}}$ in which $\equiv$ is interpreted as the binary relation $\doteq$ over $D_{\hat{\mathsf{A}}} \times D_{\hat{\mathsf{A}}}$. The relation $\doteq$ is defined recursively as follows: $d \doteq d'$ iff

- d and d' are both the empty string.
- d and d' are both composed strings such that d_h and d'_h, the first two elements of d, d' satisfy $d_h \doteq d'_h$ and d_t and d'_t the two strings obtained by removing the first element in respectively d and d' satisfy $d_t \doteq d'_t$.
- d and d' are both the empty multiset.
- d and d' are both non-empty multiset and there exists d_e and d'_e respectively in d and d' such that $d_e \doteq d'_e$ and $d \setminus d_e \doteq d' \setminus d'_e$.
- d and d' are respectively of the form $\langle d_1 \rangle$ and $\langle d'_1 \rangle$ and $d_1 \doteq d'_1$.
- d and d' are respectively of the form $cap\ d_1$ and $cap\ d'_1$ and $d_1 \doteq d'_1$ where cap belongs to $\{in, out, open\}$.
- d and d' are respectively of the form $d_1[d_2]$ and $d'_1[d'_2]$ and $d_1 \doteq d'_1$, $d_2 \doteq d'_2$.
- d and d' are respectively of the form $(d_1)d_2$ and $(d'_1)d'_2$ and $d_1 \doteq d'_1$, $d_2 \doteq d'_2$.

Proposition A.3 $\hat{\mathsf{R}}$ *is a model of the axioms for* $\equiv$*.*

Proof By case inspection. ■

Proposition A.4 *For any process P, for any M, for any name n, for any $cap \in \{in, out, open\}$,*

- *for any process Q, we have $\mathbf{0} \not\equiv M[P]$, $\mathbf{0} \not\equiv (n).P$, $\mathbf{0} \not\equiv \langle M \rangle$, $\mathbf{0} \not\equiv cap\ M.P$ and $\mathbf{0} \not\equiv P \mid Q$ if $P \not\equiv \mathbf{0}$.*
- *if $P \not\equiv \mathbf{0}$, then for any processes Q, P' such that $Q \not\equiv \mathbf{0}$, we have $P \mid Q \not\equiv M[P']$, $P \mid Q \not\equiv (n).P'$, $P \mid Q \not\equiv \langle M \rangle$, $P \mid Q \not\equiv cap\ M.P'$.*
- *for any processes Q, P' and for any M', we have $M[P] \not\equiv (n).Q$, $M[P] \not\equiv \langle M' \rangle$, $M[P] \not\equiv cap\ M'.P'$ and $M[P] \not\equiv M'[P']$ if M, M' are two different sequences or if $P \not\equiv P'$.*
- *for any M', we have $\langle M \rangle \not\equiv cap\ M'.P$, $\langle M \rangle \not\equiv (n).P$ and $\langle M \rangle \not\equiv \langle M' \rangle$ if M, M' are two different sequences.*
- *for any process Q, for any names n, m, we have $(n).P \not\equiv cap\ M.Q$ and $(n).P \not\equiv (m).Q$ if n, m are two different names or if $P \not\equiv Q$.*
- *for any process Q, for any M' and for any capability $cap' \in \{in, out, open\}$, we have $cap\ M.P \not\equiv cap'\ M'.Q$ if either $cap \neq cap'$ or M, M' are two different sequences or if $Q \not\equiv Q$.*

Proof It is easy to check that all the statements above holds for $\hat{\mathsf{R}}$. Using Proposition A.3 with Theorem A.2, those statements hold for ambient processes and $\equiv$. ■

Proposition A.5 *For any sequential substitution σ, for any prime π such that $\langle \{\pi\}; \sigma \rangle$ is normal, $\mathcal{U}(\pi, \sigma) \not\equiv \mathbf{0}$.*

Proof Straightforward from the definition of $\mathcal{U}$ and Proposition A.3 ■

Restatement of Proposition 3.1 *Let $\langle \prod_{i \in I} \pi_i; \sigma \rangle$ be a normal closure. Then*

(1) *$\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv \mathbf{0}$ iff $I = \varnothing$.*

(2) *$\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv M[Q]$ iff $\exists M', \tilde{Q}$: I is a singleton $\{i\}$, $\pi_i = M'[\tilde{Q}]$, $M'\sigma = M$, $\mathcal{U}(\tilde{Q}, \sigma) \equiv Q$.*

(3) *$\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv P' \mid P''$ iff $\exists J, K$: $J \cup K = I$, $J \cap K = \varnothing$, $P' \equiv \mathcal{U}(\prod_{j \in J} \pi_j, \sigma)$, $P'' \equiv \mathcal{U}(\prod_{k \in K} \pi_k, \sigma)$.*

(4) *$\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv \langle M \rangle$ iff $\exists M'$: I is a singleton $\{i\}$, $\pi_i = \langle M' \rangle$ and $M'\sigma = M$.*

(5) *$\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv (n).P$ iff $\exists \tilde{P}$: I is a singleton $\{i\}$, $\pi_i = (n).\tilde{P}$ and $\mathcal{U}(\tilde{P}, \sigma) \equiv P$.*

Proof For the first point, if $I = \varnothing$ then $\tilde{P} = \{\}$; so, by definition for $\mathcal{U}$, $\mathcal{U}(\tilde{P},\sigma) \equiv \mathbf{0}$. Now for the other direction, the closure $\langle\tilde{P};\sigma\rangle$ being normal, if I is not empty, then by Proposition A.4 and the definition for $\mathcal{U}$, $\mathcal{U}(\prod_{\in I}\pi,\sigma) \not\equiv \mathbf{0}$.

For the second point, for the direction from right to left: $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv \mathcal{U}(\{\pi_i\},\sigma) \equiv \mathcal{U}(\{M'[\tilde{Q}]\},\sigma)$ since I is a singleton $\{i\}$ and $\pi_i = M'[\tilde{Q}]$. Now, by definition for $\mathcal{U}$, $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv M'\sigma[\mathcal{U}(\tilde{Q},\sigma)] \equiv M[\mathcal{U}(\tilde{Q},\sigma)]$ since $M'\sigma = M$. So, $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv M[Q]$. From left to right: let us assume that I is not a singleton. For $I = \varnothing$, according to the first point, $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv \mathbf{0}$ and thus, by Proposition A.4, $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \not\equiv M[Q]$ for any M, Q. Now, the closure $\langle\tilde{P};\sigma\rangle$ being normal, if I contains at least two elements then by definition of $\mathcal{U}$, $\mathcal{U}(\tilde{P},\sigma) \equiv R' \mid R''$ for some $R, R' \not\equiv \mathbf{0}$ by Propositions A.5 and A.4 . Thus, still by Proposition A.4, $\mathcal{U}(\tilde{P},\sigma) \not\equiv M[Q]$ whatever M, Q are. So, I is a singleton. Now, if $\pi_i \neq M'[\tilde{Q}]$ or $M'\sigma, M$ are two different sequences, once again from the definition of $\mathcal{U}$ and Proposition A.4, $\mathcal{U}(\tilde{P},\sigma) \not\equiv M[Q]$. Finally, since $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) = M[\mathcal{U}(\tilde{Q},\sigma)]$, we have $\mathcal{U}(\tilde{Q},\sigma) \equiv Q$.

For the third point, from right to left: we have $P' \mid P'' \equiv \mathcal{U}(\prod_{j\in J}\pi_j,\sigma) \mid \mathcal{U}(\prod_{k\in K}\pi_k,\sigma)$. By definition of $\mathcal{U}$, since J, K are disjoint and $J \cup K = I$, $P' \mid P'' \equiv \mathcal{U}(\prod_{i\in I}\pi_i,\sigma)$. From left to right: by definition, $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) = \mathcal{U}(\pi_1,\sigma) \mid \ldots \mid \mathcal{U}(\pi_k,\sigma)$ where I is assumed to be $\{1,\ldots,k\}$ and the π_i's are primes. Since $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) = P' \mid P''$, there must exist I, J two disjoint sets of indices such that $I \cup J = 1..k$, $P' \equiv \mathcal{U}(\prod_{i\in I}\pi_i,\sigma)$ and $P'' \equiv \mathcal{U}(\prod_{j\in J}\pi_j,\sigma)$.

For the fourth point, from right to left: from the definition of $\mathcal{U}$, we have $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) = \mathcal{U}(\pi_i,\sigma) = \langle M'\sigma\rangle$. So, using the hypothesis, $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv \langle M\rangle$. From left to right: similar to the second point.

For the fifth point, from right to left: from the definition of $\mathcal{U}$, we have $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) = \mathcal{U}(\pi_i,\sigma) = (n).\mathcal{U}(\tilde{P},\sigma)$. Using the hypothesis, $\mathcal{U}(\prod_{i\in I}\pi_i,\sigma) \equiv (n).P$. From left to right: similar to the second point. ■

A.2 Properties of the Auxiliary Functions

Here, we state and prove correctness properties needed in subsequent sections of the auxiliary functions *nam*, *len*, *fst*, and *split*.

First, the function *nam* is correct in the following sense.

Proposition A.6 *$nam(M,\sigma) = n$ iff $M\sigma = n$.*

Proof Straightforward by induction over the length of the sequential substitution σ. ■

Second, the function *len* has the following property.

Proposition A.7 *$len(M,\sigma) = l$ iff $M\sigma = N_1.\ldots.N_l$ with N_i being either a name or of the form cap N' with $cap \in \{in, out, open\}$.*

Proof The proof goes by induction on the length of the sequential substitution σ.

For σ being the empty sequence ι: $M\iota = M = N_1.\ldots.N_l$. By definition, $len(N_1.\ldots.N_l, \iota) = \sum_{i=1}^{l} len(N_i, \iota)$. Since each N_i is either a name n or of the form *in* N', *out* N' or *open* N', we have $len(N_i, \iota) = 1$. This is equivalent to $len(N_1.\ldots.N_l, \iota) = l$.

For σ being the sequence $\{x \leftarrow M'\}\sigma'$ of length at least 1:
let $M = N'_1.\ldots.N'_k$. By induction over k:

- $k = 0$: in this case, $M = \epsilon$ and $M\{x \leftarrow M'\}\sigma = \epsilon$. So, $l = 0$ and by definition $len(M, \sigma) = 0$.

- $k = 1$: in this case $M = N'_1$ and we have three cases

  - N'_1 is of the form *cap* N' for some *cap* $\in \{in, out, open\}$: in this case, $M\{x \leftarrow M'\}\sigma$ is of the form $cap\, N''$ and by definition, $len(M, \sigma\{x \leftarrow M\}) = 1$.
  - N'_1 is a name different from x: in this case, $M\{x \leftarrow M'\}\sigma = M\sigma$ and $len(M, \{x \leftarrow M'\}\sigma) = len(M, \sigma)$. Using the induction hypothesis, $M\sigma = N''_1.\ldots.N''_l$ iff $len(M, \sigma) = l$, therefore $M\{x \leftarrow M'\}\sigma = N''_1.\ldots.N''_l$ iff $len(M, \{x \leftarrow M'\}\sigma) = l$.
  - $N'_1 = x$: in this case, $M\{x \leftarrow M'\}\sigma = M'\sigma$ and $len(M, \{x \leftarrow M'\}\sigma) = len(M', \sigma)$. By induction hypothesis $M'\sigma = N''_1.\ldots.N''_l$ iff $len(M', \sigma) = l$, so $M\{x \leftarrow M'\}\sigma = N''_1.\ldots.N''_l$ iff $len(M, \{x \leftarrow M'\}\sigma) = l$.

- $k > 1$: using the induction hypothesis, $len(N'_1.\ldots.N'_{k-1}, \{x \leftarrow M'\}\sigma) = l'$ iff $N'_1\{x \leftarrow M'\}\sigma.\ldots.N'_{k-1}\{x \leftarrow M'\}\sigma = N''_1.\ldots.N''_{l'}$ and for the expression N_k, $len(N_k, \{x \leftarrow M'\}\sigma) = l''$ iff $N'_k\{x \leftarrow M'\}\sigma = N''_{l'+1}.\ldots.N''_{l'+l''}$. By definition, $len(M, \{x \leftarrow M'\}\sigma)$ is the sum of $len(N'_1.\ldots.N'_{k-1}, \{x \leftarrow M'\}\sigma)$ and of $len(N'_k, \{x \leftarrow M'\}\sigma)$. So, we can conclude that $M\{x \leftarrow M'\}\sigma = N''_1.\ldots.N''_{l'+l''}$ iff $len(M, \{x \leftarrow M'\}\sigma) = l' + l''$. ■

Third, we state the correctness of *fst* in Proposition A.9. To prove it, we need the following lemma.

Lemma A.8 *Let $\langle \tilde{P}; \{x \leftarrow N\}\sigma \rangle$ be a normal closure. Then $\langle \tilde{P}\{x \leftarrow N\}; \sigma \rangle$ is normal and $\mathcal{U}(\tilde{P}, \{x \leftarrow N\}\sigma) \equiv \mathcal{U}(\tilde{P}\{x \leftarrow N\}, \sigma)$.*

Proof For the normality of $\langle \tilde{P}\{x \leftarrow N\}; \sigma \rangle$: we can show that $\mathcal{U}(\tilde{P}\{x \leftarrow N\}, \sigma)$ is defined by induction over the structure of processes and primes. The only non-trivial case is for $\tilde{P} = M(o).\tilde{P}'$: then, $\tilde{P}\{x \leftarrow N\} = M\{x \leftarrow N\}(o).\tilde{P}'\{x \leftarrow N\}$. Since $\mathcal{U}(\tilde{P}, \{x \leftarrow N\}\sigma)$ by assumption and $\mathcal{U}(\tilde{P}'\{x \leftarrow N\}, \sigma)$ by induction hypothesis are defined and $(M\{x \leftarrow N\})\sigma = M(\{x \leftarrow N\}\sigma)$, $\mathcal{U}(\tilde{P}\{x \leftarrow N\}, \sigma)$ is defined. For the second statement, since $\langle \tilde{P}; \{x \leftarrow N\}\sigma \rangle$ is normal, x and names from N are not bound in $\tilde{P}$, so $bn(\tilde{P}\{x \leftarrow N\}) = bn(\tilde{P})$ and $fn(\tilde{P}\{x \leftarrow N\})$ contains $fn(\tilde{P})$ and some possibly other names that do not belong to $bn(\tilde{P})$. So, $fn(\tilde{P}\{x \leftarrow N\}) \cap bn(\tilde{P}\{x \leftarrow N\}) = \varnothing$. Moreover, as the bound names from $\tilde{P}$ do not occur in $\{x \leftarrow N\}\sigma$ and $bn(\tilde{P}\{x \leftarrow N\}) = bn(\tilde{P})$, $bn(\tilde{P}\{x \leftarrow N\}) \cap names(\sigma) = \varnothing$. Since x is not bound in $\tilde{P}$, occurrences of bound variables in $\tilde{P}$ are not affected by the substitution $\{x{\leftarrow}N\}$. The requirement on offsets is trivially preserved and finally, as $\{x{\leftarrow}N\}\sigma$ is acyclic, σ is so.

We show that $\mathcal{U}(\tilde{P}, \{x{\leftarrow}N\}\sigma) \equiv \mathcal{U}(\tilde{P}\{x{\leftarrow}N\}, \sigma)$ by induction over the structures of processes and primes taking into account that x in not a bound variable in $\tilde{P}$. ■

Proposition A.9 *Let N be a capability of the form $in\,n$, $out\,n$ or $open\,n$. Then for all normal closures $\langle\tilde{Q};\sigma\rangle$, there exists Q such that $\mathcal{U}(M(o).\tilde{Q}, \sigma) \equiv N.Q$ iff $\mathit{fst}(M, o, \sigma) = N$.*

Proof Let us assume that $M = N_1. \dots .N_l$ and that $N = cap\ n$ where cap ranges over in, out, $open$. The proof goes by induction over the offset o.

Case where $o = 0$: we have $\mathit{fst}(M, 0, \sigma) = cap\ n$. We follow by induction over the length of the sequential substitution σ.

- case where the length of σ is 0: $\sigma = \iota$ and $\mathit{fst}(M, 0, \iota) = cap\,n$. By definition of fst, this is equivalent to $\mathit{fst}(N_1, 0, \iota) = cap\,n$ and to $N_1 = cap\,n$. Furthermore, as $\mathcal{U}(M(0).\tilde{Q}, \iota) = N_1. \dots .N_l.\mathcal{U}(\tilde{Q})$, this is equivalent to $\mathcal{U}(M(0).\tilde{Q}, \iota) \equiv cap\,n.Q$ for some Q.

- case where σ is of the form $\{x{\leftarrow}M'\}\sigma'$ and the proposition holds for σ': by definition of fst, $\mathit{fst}(M, 0, \sigma) = \mathit{fst}(N_1, 0, \sigma) = cap\ n$. Now, according to the value of N_1:

  - N_1 is of the form $cap\ L$: so, $nam(L, \sigma) = n$ which is equivalent due to Proposition A.6, to $L\sigma = n$. As $\mathcal{U}(M(0).\tilde{Q}, \sigma) = N_1\sigma. \dots .N_l\sigma.\mathcal{U}(\tilde{Q}, \sigma)$, $\mathcal{U}(M(0).\tilde{Q}, \sigma) = cap\,n.N_2\sigma \dots .N_l\sigma.\mathcal{U}(\tilde{Q}, \sigma)$. Therefore, this is equivalent to that $\mathcal{U}(M(0).\tilde{Q}, \sigma) \equiv cap\ n.Q$ for some Q.
  - N_1 is a name m: for each of the two cases in the definition of fst.

    Case where $m = x$: we have $\mathit{fst}(N_1, 0, \sigma) = \mathit{fst}(m, 0, \{x{\leftarrow}M'\}\sigma') = \mathit{fst}(M', 0, \sigma') = cap\,n$. By induction hypothesis, it is equivalent to that for any $\tilde{Q}$, $\mathcal{U}(M'(0).\tilde{Q}, \sigma') \equiv cap\ n.Q$ for some Q. In particular for some P, $cap\ n.P \equiv \mathcal{U}(M'(0).N_2\{x{\leftarrow}M'\}. \dots .N_l\{x{\leftarrow}M'\}(0).\tilde{P}\{x{\leftarrow}N'\}, \sigma')$, that is $cap\,n.P \equiv M'\sigma'.N_2\{x{\leftarrow}M'\}\sigma'. \dots .N_l\{x{\leftarrow}M'\}\sigma'.\mathcal{U}(\tilde{P}\{x{\leftarrow}N'\}, \sigma')$. So $cap\ n.P \equiv m\{x{\leftarrow}M'\}\sigma'.N_2\sigma. \dots .N_l\sigma.\mathcal{U}(\tilde{P}, \{x{\leftarrow}N'\}\sigma')$ by Lemma A.8. And thus, by definition of $\mathcal{U}$, this is equivalent to that for some P, $cap\ n.P \equiv \mathcal{U}(M(0).\tilde{P}, \sigma)$.

    Case where $m \neq x$: in this case, $\mathit{fst}(M, 0, \sigma) = \mathit{fst}(m, 0, \sigma') = cap\ n$. By induction hypothesis, this is equivalent to that for any $\tilde{Q}$, $\mathcal{U}(m(0).\tilde{Q}, \sigma') \equiv cap\,n.Q$ for some Q. The rest of the proof is similar to the previous case, using the fact that $m\sigma' = m\{x{\leftarrow}M'\}\sigma'$ since $m \neq x$.

Case where the proposition holds for any $o' < o$: we have $\mathit{fst}(M, o, \sigma) = cap\,n$. By induction over the length of the sequential substitution σ.

- case where the length of σ is 0: $\sigma = \iota$ and $\mathit{fst}(M, o, \iota) = cap\ n$. Since $len(N_1. \dots .N_o, \iota) = o$, $cap\ n = \mathit{fst}(N_{o+1}. \dots .N_l, 0, \iota)$. Using the base case, this latter is equivalent to that for any $\tilde{P}$, $\mathcal{U}(N_{o+1}.\dots.N_l(0).\tilde{P},\iota) \equiv cap\ n.P$ for some P. Now, this is equivalent to $cap\ n.P \equiv N_{o+1}.\dots.N_l.\mathcal{U}(\tilde{P},\iota)$ by definition of $\mathcal{U}$. Finally, as $M\iota = N_1.\dots.N_l$, by definition of $\mathcal{U}$, it is equivalent to that $cap\ n.P \equiv \mathcal{U}(M(o).\tilde{P},\iota)$ for some P.

- case where σ is of the form $\{x \leftarrow M'\}\sigma'$ and the proposition holds for σ': since $fst(M,o,\sigma)$ is defined, $o < len(M,\sigma)$. Let i be the unique integer such that $len(N_1.\dots.N_{i-1},\sigma) \leq o$ and $len(N_1.\dots.N_i,\sigma) > o$ and p be $o - len(N_1.\dots.N_{i-1},\sigma)$. Then we have $capn = fst(M,o,\sigma) = fst(N_i.\dots.N_l,p,\sigma)$. Now, according to the value of N_i:

  - N_i is of the form $cap\ L$: so, $nam(L,\sigma) = n$ which is equivalent due to Proposition A.6, to $L\sigma = n$. Furthermore, since $len(N_i,\sigma) = 1$, we have $o = len(N_1.\dots.N_{i-1},\sigma)$ and thus, $p = 0$. Hence, $cap\ n = fst(N_i.\dots.N_l,0,\sigma)$. According to the base case, this is equivalent to that for any $\tilde{P}$, $\mathcal{U}(N_i.\dots.N_l(0).\tilde{P},\sigma) \equiv cap\ n.P$ for some P. Let $M\sigma$ be $N'_1.\dots.N'_k$. So by definition of $\mathcal{U}$, $\mathcal{U}(M(o).\tilde{P},\sigma) = N'_{o+1}.\dots.N'_k.\mathcal{U}(\tilde{P},\sigma)$. Now, as $o = len(N_1.\dots.N_{i-1},\sigma)$, $N_i\sigma.\dots.N_l\sigma = N'_{o+1}.\dots.N'_k$. Hence, $\mathcal{U}(M(o).\tilde{P},\sigma) = N_i\sigma.\dots.N_l\sigma.\mathcal{U}(\tilde{P},\sigma)$. Equivalently, $\mathcal{U}(M(o).\tilde{P},\sigma) = \mathcal{U}(N_i.\dots.N_l(0).\tilde{P},\sigma)$ and so, $\mathcal{U}(M(o).\tilde{P},\sigma) \equiv cap\ n.P$ for some P.
  - N_i is a name m: in this case, we have $len(N_i,\sigma) > p$. Hence, by definition of fst, $cap\ n = fst(M,o,\sigma) = fst(N_i,p,\{x \leftarrow M'\}\sigma')$. For each of the two cases in the definition of fst:

    Case where $m = x$: we have $cap\ n = fst(M',p,\sigma')$. By induction hypothesis, this is equivalent to that for any $\tilde{Q}$, $\mathcal{U}(M'(p).\tilde{Q},\sigma') \equiv cap\ n.Q$ for some Q. As a particular case, this latter holds for $Q = P$ and for $\tilde{Q} = N_{i+1}\{x \leftarrow M'\}.\dots.N_l\{x \leftarrow M'\}(0).\tilde{P}\{x \leftarrow M'\}$. Now, from the definition of $\mathcal{U}$ and using that $M' = N_i\{x \leftarrow M'\}$, this is equivalent to that $\mathcal{U}(N_i\{x \leftarrow M'\}.\dots.N_l\{x \leftarrow M'\}(p).\tilde{P}\{x \leftarrow M'\},\sigma') = cap\ n.P$ for some P. Let $N'_1.\dots.N'_k$ be $N_i\sigma$. Then, still by definition of $\mathcal{U}$, it is equivalent to that $N'_{p+1}.\dots.N'_k.N_{i+1}\sigma.\dots.N_l\sigma.\mathcal{U}(P\{x \leftarrow M'\},\sigma') = cap\ n.P$. By Lemma A.8, it is equivalent to $N'_{p+1}.\dots.N'_k.N_{i+1}\sigma.\dots.N_l\sigma.\mathcal{U}(P,\sigma) = cap\ n.P$. Once again, by definition of $\mathcal{U}$, we have $\mathcal{U}(N_i.\dots.N_l(p).\tilde{P},\sigma) = cap\ n.P$. Let p' be $len(N_1.\dots.N_{i-1},\sigma)$. By definition of $\mathcal{U}$, we have $\mathcal{U}(N_1.\dots.N_{i-1}(p').N_i.\dots.N_l(p).\tilde{P},\sigma) = cap\ n.P$. By definition of $\mathcal{U}$, $\mathcal{U}(N_1.\dots.N_{i-1}.N_i.\dots.N_l(p+p').\tilde{P},\sigma) = cap\ n.P$. Finally, as $p + p' = o$, this latter is equivalent to that $\mathcal{U}(M(o).\tilde{P},\sigma) = cap\ n.P$ for some P.

    Case where $m \neq x$: by definition of fst, $cap\ n = fst(m,p,\{x \leftarrow M'\}\sigma') = fst(m,p,\sigma')$. By induction hypothesis, this is equivalent to that for all $\tilde{Q}$, there exists $\tilde{Q}$ such that $\mathcal{U}(m(p).\tilde{Q},\sigma) \equiv cap\ n.\tilde{Q}$. The rest of the proof is similar to the previous case, using the fact that $m\sigma' = m\{x \leftarrow M'\}\sigma'$ since $m \neq x$. ■

Fourth, we prove that *split* is correct in the following sense.

Proposition A.10 *Let $\langle \prod_{i \in I} \pi_i; \sigma \rangle$ be a normal closure, and let L be of the form in n, out n or open n. Then $\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv L.P$ iff $\exists L', o, \tilde{P}, \tilde{P}'$: I is a singleton $\{i\}$, $\pi_i = L'(o).\tilde{P}'$, $split(\pi_i, \sigma) = (L, \tilde{P})$ and $\mathcal{U}(\tilde{P}, \sigma) \equiv P$.*

Proof From right to left: we have $\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) = \mathcal{U}(\pi_i, \sigma)$, $\pi_i = L'(o).\tilde{P}'$, $split(\pi_i, \sigma) = (L, \tilde{P})$. By Proposition A.9, $\mathcal{U}(\pi_i, \sigma) \equiv L.P$ for some P. Moreover, for $L'\sigma$ being of the form $L'_1. \dots .L'_l$, $\mathcal{U}(\pi_i, \sigma) = L'_{o+1}. \dots .L'_l.\mathcal{U}(\tilde{P}, \sigma)$ and $L'_{o+1} = L$. Note that $\mathcal{U}(\pi_i, \sigma)$ being defined, we have $o < len(L', \sigma) = l$. Now, by the definition of *split*, according to the values of o and $len(L', \sigma)$:

- $len(L', \sigma) > o + 1$: in this case, $\tilde{P} = \{L'(o+1).\tilde{P}'\}$. So, by definition of $\mathcal{U}$, $\mathcal{U}(\{L'(o+1).\tilde{P}'\}, \sigma) = L'_{o+2}. \dots .L'_l.\mathcal{U}(\tilde{P}', \sigma)$ and thus, $\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv L'_{o+1}.\mathcal{U}(\{L'(o+1).\tilde{P}'\}, \sigma) \equiv L.P$ for $P \equiv \mathcal{U}(\{L'(o+1).\tilde{P}'\}, \sigma) \equiv \mathcal{U}(\tilde{P}, \sigma)$.

- $len(L', \sigma) = o + 1$: in this case, $\tilde{P} = \tilde{P}'$. Therefore, $\mathcal{U}(\{L'(o+1).\tilde{P}'\}, \sigma) = L'_l.\mathcal{U}(\tilde{P}', \sigma) = L'_{o+1}.\mathcal{U}(\tilde{P}', \sigma) = L.\mathcal{U}(\tilde{P}', \sigma)$. Thus, $\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv L.P$ for $P \equiv \mathcal{U}(\tilde{P}', \sigma) \equiv \mathcal{U}(\tilde{P}, \sigma)$.

From left to right: let us assume that $\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \equiv L.P$. Using Proposition A.4, the set I has to be a singleton and π_i has to be of the form $L'(o).\tilde{P}'$. Now, by Proposition A.9, we know that $fst(L', o, \sigma) = L$. Thus, it is sufficient to prove that $P \equiv \mathcal{U}(\tilde{P}, \sigma)$ for $split(\pi_i, \sigma) = (L, \tilde{P})$. From the definitions of $\mathcal{U}$ and *split* and from Proposition A.4, it is straightforward to see that $P \not\equiv \mathcal{U}(\tilde{P}, \sigma)$ implies $\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \not\equiv L.P$. ■

A.3 Proof of Proposition 3.2

Using Lemma A.11 below, we show Proposition 3.2(1), that $\downarrow^*$, the reflexive and transitive closure of the sublocation relation $\downarrow$, preserves normality of closures.

Lemma A.11 *If $\langle \tilde{P}; \sigma \rangle$ is normal, then for any $\langle \tilde{P}'; \sigma \rangle$ such that $\langle \tilde{P}; \sigma \rangle \downarrow \langle \tilde{P}'; \sigma \rangle$, the closure $\langle \tilde{P}'; \sigma \rangle$ is normal.*

Proof From the definition of $\downarrow$, we have $\tilde{P} = \tilde{Q} +\!\!+ \{M[\tilde{P}']\}$ for some $\tilde{Q}$, M. Thus, by the first point of Proposition A.1, the closure $\langle \{M[\tilde{P}']\}; \sigma \rangle$ is normal. Now, the names from M occur freely in $\{M[\tilde{P}']\}$. So, $\langle \{M[\tilde{P}']\}; \sigma \rangle$ being normal, none of the names from M is in $bn(\{M[\tilde{P}']\})$ and thus, in $bn(\tilde{P}')$. Therefore, by the second point of Proposition A.1, $\langle \tilde{P}'; \sigma \rangle$ is normal. ■

Restatement of Proposition 3.2(1) *If $\langle \tilde{P}; \sigma \rangle$ is normal and $\langle \tilde{P}; \sigma \rangle \downarrow^* \langle \tilde{P}'; \sigma \rangle$ then $\langle \tilde{P}'; \sigma \rangle$ is normal.*

Proof A simple induction using Lemma A.11. ■

Using Lemmas A.12 and A.13 below, we show Proposition 3.2(2), that $\rightarrow^*$, the reflexive and transitive closure of the reduction relation $\rightarrow$, preserves normality of closures.

Lemma A.12 *If $\langle\{\pi\};\sigma\rangle$ is normal and $split(\pi,\sigma) = (N,\tilde{S})$ then $\langle\tilde{S};\sigma\rangle$ is normal.*

Proof Since $split(\pi,\sigma) = (N,\tilde{S})$, $\pi = M(o).\tilde{S}'$ for some expression M and some annotated process $\tilde{S}'$. Furthermore, $\mathcal{U}(\{\pi\},\sigma)$ being defined, $\mathcal{U}(\tilde{S}',\sigma)$ is defined. Now, according to the value of $\tilde{S}$: if $\tilde{S} = M(o+1).\tilde{S}'$ then, from the definition of *split*, $o+1 < len(M,\sigma)$. So, from the definition of $\mathcal{U}$, $\mathcal{U}(\tilde{S}',\sigma)$ being defined, $\mathcal{U}(M(o+1).\tilde{S}',\sigma)$= $\mathcal{U}(\tilde{S},\sigma)$ is defined. If $\tilde{S} = \tilde{S}'$ then $\mathcal{U}(\tilde{S},\sigma)$ is defined.

Let us first notice that $bn(\{\pi\}) = bn(\{M(o+1).\tilde{S}'\}) = bn(\tilde{S}')$ and that $fn(\{\pi\}) = fn(\{M(o+1).\tilde{S}'\}) \supseteq fn(\tilde{S}')$. Therefore, since by normality $bn(\{\pi\}) \cap (fn(\{\pi\}) \cup names(\sigma)) = \varnothing$, we have $bn(\tilde{S}) \cap (fn(\tilde{S}) \cup names(\sigma) = \varnothing$.

The last three statements are obvious to check. ■

Lemma A.13 *If $\langle\tilde{P};\sigma\rangle$ is normal, then for any $\langle\tilde{P}';\sigma\rangle$ such that $\langle\tilde{P};\sigma\rangle \to \langle\tilde{P}';\sigma\rangle$, the closure $\langle\tilde{P}';\sigma\rangle$ is normal, and moreover*

- *either $\sigma' = \sigma$, $bn(\tilde{P}) = bn(\tilde{P}')$ and $fn(\tilde{P}') \subseteq fn(\tilde{P})$,*
- *or for some x, M, $\sigma' = \{x \leftarrow M\}\sigma$, $bn(\tilde{P}) = bn(\tilde{P}') \cup \{x\}$ and $fn(\tilde{P}') \subseteq fn(\tilde{P}) \cup \{x\}$.*

Proof The proof goes by induction over the structure of the context under which the reduction takes place.

If the context is empty, then the applied reduction corresponds to one of the rules (Trans In), (Trans Out), (Trans Open) and (Trans I/O). For (Trans In), (Trans Out) and (Trans Open) respectively, $\langle\{N[\tilde{Q} +\!\!+ \{\pi\}], M[\tilde{R}]\};\sigma\rangle$, $\langle\{M[\{N[\tilde{Q} +\!\!+ \pi]\} +\!\!+ \tilde{R}]\};\sigma\rangle$ and $\langle\{M[\tilde{P}],\pi\};\sigma\rangle$ are normal by assumption.

Concerning the second claim of the lemma: obviously, $\sigma' = \sigma$, $bn(\tilde{P}) = bn(\tilde{P}')$. For the rules (Trans In) and (Trans Out), $fn(\tilde{P}) = fn(\tilde{P}')$ and for (Trans Open) $fn(\tilde{P}') \subseteq fn(\tilde{P})$ (the execution of *open* may let an ambient name disappeared).

Now for the first claim, by using Proposition A.1, $\langle\pi;\sigma\rangle$ is normal. Then, from Lemma A.12 together with the transition rules on closures, $\langle\tilde{P};\sigma\rangle$ is normal (where $split(\pi,\sigma) = (N,\tilde{P})$ and N being respectively *in* m, *out* m and *open* m). Finally, using the fact that $bn(\{\pi\}) = bn(\tilde{P})$ and that $fn(\{\pi\}) \subseteq fn(\tilde{P})$ and by applying once more Proposition A.1, the closures $\langle\{M[\{N[\tilde{Q} +\!\!+ \pi]\} +\!\!+ \tilde{R}]\};\sigma\rangle$, $\langle\{N[\tilde{Q} +\!\!+ \{\pi\}], M[\tilde{R}]\};\sigma\rangle$ and $\langle\tilde{P} +\!\!+ \tilde{Q};\sigma\rangle$ are normal.

For (Trans I/O), $\langle\{\langle M\rangle,(x).\tilde{P}\};\sigma\rangle$ is normal by assumption. Let us start with the second claim of the lemma. We have $\sigma' = \{x \leftarrow M\}\sigma$. Due to the assumption of normality, x occurs at most once within an input in $\tilde{P}$ and bound and free names are disjoint in $\tilde{P}$. So, $bn(\{\langle M\rangle,(x).\tilde{P}\}) = bn(\tilde{P}) \cup \{x\}$ and $fn(\tilde{P}) \subseteq fn(\{\langle M\rangle,(x).\tilde{P}\}) \cup \{x\}$. Now, for the first claim, let us first prove that $\mathcal{U}(norm(\tilde{P},\{x \leftarrow M\}\sigma),\{x \leftarrow M\}\sigma)$ is defined by induction over the structure of $\tilde{P}$: this is obvious for $\tilde{P}$ being the empty multiset or the singleton $\{\langle M'\rangle\}$. For the induction step, this is also straightforward for $\tilde{P}$ being a multiset of primes or a singleton $\{(x').\tilde{Q}\}$ or $\{M'[\tilde{Q}]\}$. Now, for $\tilde{P} =$

$\{M'(o).\tilde{Q}\}$. By hypothesis, $\mathcal{U}(M'(o).\tilde{Q},\sigma)$ is defined. So, $o < len(M',\sigma)$. If $len(M',\{x{\leftarrow}M\}\sigma) = 0$, then $norm(\tilde{P},\{x{\leftarrow}M\}\sigma) = norm(\tilde{Q},\{x{\leftarrow}M\}\sigma)$ and so $\mathcal{U}(norm(\tilde{P},\{x{\leftarrow}M\}\sigma),\{x{\leftarrow}M\}\sigma)$ is defined by induction hypothesis. Otherwise, $len(M',\sigma) \leq len(M',\{x{\leftarrow}M\}\sigma)$. So $\mathcal{U}(norm(\tilde{P},\{x{\leftarrow}M\}\sigma),\{x{\leftarrow}M\}\sigma) = \mathcal{U}(M'(o).norm(\tilde{Q},\{x{\leftarrow}M\}\sigma),\{x{\leftarrow}M\}\sigma)$ is defined. Since every variable occurs at most once within an input in the annotated process of a normal closure, $bn(\tilde{P}) = bn(\{(x).\tilde{P},\langle M\rangle\}) \setminus \{x\}$; Moreover, since $fn(\tilde{P}) \subseteq fn(\{\langle M\rangle,(x).\tilde{P}\}) \cup \{x\}$, $bn(\{\langle M\rangle,(x).\tilde{P}\}) \cap fn(\{\langle M\rangle,(x).\tilde{P}\}) = \varnothing$. Let us show that names from $bn(\tilde{P})$ do not occur in $\sigma' = \{x{\leftarrow}M\}\sigma$. As $bn(\tilde{P}) \subseteq bn(\{\langle M\rangle,(x).\tilde{P}\})$, because of the hypothesis of normality, names from $bn(\tilde{P})$ do not occur in σ. Moreover, we know that $x \notin bn(\tilde{P})$ and names occurring in M are free in $\{\langle M\rangle,(x).\tilde{P}\}$ and so, in $\tilde{P}$. It is straightforward that the property of the uniqueness of variable within an input and the fact that offsets are equal to 0 in the scope of an input are preserved. Finally, since $\langle\{\langle M\rangle,(x).\tilde{P}\};\sigma\rangle$ is normal, σ is acyclic and as x is bound, x does not occur in σ; so the last point holds for $\langle\tilde{P};\{x{\leftarrow}M\}\sigma\rangle$.

Now, we investigate the case where the context of reduction is non-empty, that is the rule used for reduction is either (Trans Par) or (Trans Amb). We show in this case that the second claim of the lemma holds and then that normality is preserved.

For (Trans Amb): we assume the closure $\langle M[\tilde{P}];\sigma\rangle$ to be normal. For any $\tilde{S}$, we have $bn(M[\tilde{S}]) = bn(\tilde{S})$, $fn(M[\tilde{S}]) = fn(\tilde{S}) \cup fn(M[\mathbf{0}])$. Let us first consider the case where $\sigma = \sigma'$: by induction hypothesis $bn(\tilde{P}) = bn(\tilde{P}')$, $fn(\tilde{P}') \subseteq fn(\tilde{P})$. So, $bn(M[\tilde{P}]) = bn(M[\tilde{P}'])$ and $fn(M[\tilde{P}']) \subseteq fn(M[\tilde{P}])$. Now, for the case where $\sigma' = \{x{\leftarrow}M\}\sigma$: By induction hypothesis, $bn(\tilde{P}) = bn(\tilde{P}') \cup \{x\}$, $fn(\tilde{P}') = fn(\tilde{P}) \cup \{x\}$. So, $bn(M[\tilde{P}]) = bn(M[\tilde{P}']) \cup \{x\}$ and $fn(M[\tilde{P}']) = fn(M[\tilde{P}]) \cup \{x\}$.

Let us show now that $\langle M[\tilde{P}'];\sigma'\rangle$ is normal: since $\langle M[\tilde{P}];\sigma\rangle$ is normal, by Proposition A.1, $\langle\tilde{P};\sigma\rangle$ is normal. Then, since $\langle\tilde{P};\sigma\rangle \rightarrow \langle\tilde{P}';\sigma'\rangle$, by induction hypothesis, $\langle\tilde{P}';\sigma'\rangle$ is normal. So, as $bn(\tilde{P}') \subseteq bn(\tilde{P})$, by Proposition A.1, $\langle M[\tilde{P}'];\sigma'\rangle$ is normal.

For (Trans Par): we assume the closure $\langle\tilde{P} +\!\!+ \tilde{Q};\sigma\rangle$ to be normal. For any $\tilde{S},\tilde{S}'$, we have $bn(\tilde{S} +\!\!+ \tilde{S}') = bn(\tilde{S}) \cup bn(\tilde{S}')$ and $fn(\tilde{S} +\!\!+ \tilde{S}') = fn(\tilde{S}) \cup fn(\tilde{S}')$. Let us first consider the case where $\sigma = \sigma'$: as by induction hypothesis $bn(\tilde{P}) = bn(\tilde{P}')$ and $fn(\tilde{P}') \subseteq fn(\tilde{P})$, we have $bn(\tilde{P} +\!\!+ \tilde{Q}) = bn(\tilde{P}' +\!\!+ \tilde{Q})$ and $fn(\tilde{P}' +\!\!+ \tilde{Q}) \subseteq fn(\tilde{P} +\!\!+ \tilde{Q})$. Now, for the case where $\sigma' = \{x{\leftarrow}M\}\sigma$: as by induction hypothesis $bn(\tilde{P}) = bn(\tilde{P}') \cup \{x\}$ and $fn(\tilde{P}') \subseteq fn(\tilde{P}) \cup \{x\}$, we have $bn(\tilde{P} +\!\!+ \tilde{Q}) = bn(\tilde{P}' +\!\!+ \tilde{Q}) \cup \{x\}$ and $fn(\tilde{P}' +\!\!+ \tilde{Q}) \subseteq fn(\tilde{P} +\!\!+ \tilde{Q}) \cup \{x\}$.

Let us show now that $\langle\tilde{P} +\!\!+ \tilde{Q};\sigma'\rangle$ is normal: $\langle\tilde{P} +\!\!+ \tilde{Q};\sigma'\rangle$ being normal, by Proposition A.1, both $\langle\tilde{P};\sigma\rangle$ and $\langle\tilde{Q};\sigma\rangle$ are normal. Now, since $\langle\tilde{P};\sigma\rangle \rightarrow \langle\tilde{P}';\sigma'\rangle$, by induction hypothesis, $\langle\tilde{P}';\sigma'\rangle$ is normal. Let us now prove that $\langle\tilde{Q};\sigma'\rangle$ is normal: we know that $x \in bn(\tilde{P})$; so, by normality of $\langle\tilde{P} +\!\!+ \tilde{Q};\sigma\rangle$, x does not occur in $\tilde{Q}$, so $\mathcal{U}(\tilde{Q},\sigma') \equiv \mathcal{U}(\tilde{Q},\sigma)$ and thus, $\mathcal{U}(\tilde{Q},\sigma')$ is defined. The other points are obviously implied by the normality of $\langle\tilde{Q};\sigma\rangle$ and $\langle\tilde{P}';\sigma'\rangle$. Finally, the fact that $\langle\tilde{P} +\!\!+ \tilde{Q};\sigma\rangle$ and $\langle\tilde{Q};\sigma'\rangle$ are normal together with Proposition A.1 implies that $\langle\tilde{P}' +\!\!+ \tilde{Q};\sigma'\rangle$ is normal. ∎

Restatement of Proposition 3.2(2) *If $\langle \tilde{P};\sigma\rangle$ is normal and $\langle \tilde{P};\sigma\rangle \rightarrow^* \langle \tilde{P}';\sigma'\rangle$ then $\langle \tilde{P}';\sigma'\rangle$ is normal.*

Proof An induction with appeal to Lemma A.13. ∎

A.4 Proof of Proposition 3.3

We prove now that the sublocation relation defined on closures simulates the sublocation relation defined on processes.

Restatement of Proposition 3.3 *Assume $\langle \tilde{P};\sigma\rangle$ is a normal closure. If $\langle \tilde{P};\sigma\rangle \downarrow \langle \tilde{Q};\sigma\rangle$ then $\mathcal{U}(\tilde{P},\sigma) \downarrow \mathcal{U}(\tilde{Q},\sigma)$. If $\mathcal{U}(\tilde{P},\sigma) \downarrow Q$ then there exists $\tilde{Q}$ such that $\langle \tilde{P};\sigma\rangle \downarrow \langle \tilde{Q};\sigma\rangle$ and $\mathcal{U}(\tilde{Q},\sigma) \equiv Q$.*

Proof For the first point, by definition for $\downarrow$ on closures, we have $\tilde{P} = \tilde{Q} +\!\!+ \{M[\tilde{P}']\}$ for some $\tilde{Q}$, M, n such that $nam(M,\sigma) = n$. Therefore, by definition of $\mathcal{U}$, $\mathcal{U}(\tilde{P},\sigma) = \mathcal{U}(\tilde{Q},\sigma) \mid M\sigma[\mathcal{U}(\tilde{P}',\sigma)]$. Note that $\langle \tilde{P};\sigma\rangle$ being normal, both $\langle \tilde{Q};\sigma\rangle$, $\langle \tilde{P}';\sigma\rangle$ are defined and thus, processes. Now, for the two processes $\mathcal{U}(\tilde{P},\sigma), \mathcal{U}(\tilde{P}',\sigma)$, there exists a process Q (namely $\mathcal{U}(\tilde{Q},\sigma)$) and a name n ($n = M\sigma$ by Proposition A.6) such that $\mathcal{U}(\tilde{P},\sigma) \equiv Q \mid n[\mathcal{U}(\tilde{P}',\sigma)]$. So, $\mathcal{U}(\tilde{P},\sigma) \downarrow \mathcal{U}(\tilde{P}',\sigma)$.

For the second point, by definition of $\downarrow$ on processes, $\mathcal{U}(\tilde{P},\sigma) \downarrow P'$ iff there exists Q, n such that $\mathcal{U}(\tilde{P},\sigma) \equiv Q \mid n[P']$. The annotated process $\tilde{P}$ being of the form $\prod_{k\in K} \pi_k$, by Proposition 3.1, there exists I, J such that $I \cup J = K$, $I \cap J = \varnothing$ and $\mathcal{U}(\prod_{i\in I} \pi_i,\sigma) \equiv Q$, $\mathcal{U}(\prod_{j\in J} \pi_j,\sigma) \equiv n[P']$. From $\mathcal{U}(\prod_{j\in J} \pi_j,\sigma) \equiv n[P']$, by Proposition 3.1, there exists $M', \tilde{P}'$ such that J is a singleton $\{j\}$, $\pi_j = M'[\tilde{P}']$, $M'\sigma = n$ and $\mathcal{U}(\tilde{P}',\sigma) \equiv P'$. Since $M'\sigma = n$, by Proposition A.6, $nam(M',\sigma) = n$. Furthermore, $\tilde{P}$ is equal to $\prod_{i\in I} \pi_i +\!\!+ \{M'[\tilde{P}']\}$. So, $\langle \tilde{P};\sigma\rangle \downarrow \langle \tilde{P}';\sigma\rangle$ and $\mathcal{U}(\tilde{P}',\sigma) \equiv P'$. ∎

A.5 Proof of Proposition 3.4

Given Lemmas A.14, A.15, and A.16 below, we prove Proposition 3.4, that the reduction relation defined on closures simulates the reduction relation defined on processes.

Lemma A.14 *Let $\langle \tilde{P};\sigma\{x\leftarrow M\}\rangle$ be a normal closure such that all the offsets o occurring in $\tilde{P}$ are set to 0. Then $\mathcal{U}(\tilde{P},\sigma\{x\leftarrow M\}) \equiv \mathcal{U}(\tilde{P},\sigma)\{x\leftarrow M\}$.*

Proof The proof goes by induction over the structures of processes and primes. Most of the cases simply uses the definition of $\mathcal{U}$ and the application of a substitution. We detail here the only two cases that are not straightforward.

For primes π:

- case where $\pi = (y).\tilde{P}'$:

$$
\begin{array}{rcl}
\mathcal{U}((y).\tilde{P}',\sigma)\{x\leftarrow M\} & \equiv & ((y).\mathcal{U}(\tilde{P}',\sigma))\{x\leftarrow M\} \\
& \equiv & ((y)\{x\leftarrow M\}).(\mathcal{U}(\tilde{P}',\sigma)\{x\leftarrow M\}) \\
& \equiv & (y).(\mathcal{U}(\tilde{P}',\sigma)\{x\leftarrow M\}) \\
& \equiv & (y).(\mathcal{U}(\tilde{P}',\sigma\{x\leftarrow M\})) \\
& \equiv & \mathcal{U}((y).\tilde{P}',\sigma\{x\leftarrow M\})
\end{array}
$$

The first and the last equivalences follow from the definition of $\mathcal{U}$; the second one corresponds simply to the application of the substitution $\{x\leftarrow M\}$. For the third one, the closure $\langle\tilde{P};\sigma\{x\leftarrow M\}\rangle$ being normal, by Proposition A.1, the closure $\langle\{\pi\};\sigma\{x\leftarrow M\}\rangle$ is normal too. Therefore, as y is a bound variable and $bn(\tilde{P})\cap dom(\sigma\{x\leftarrow M\})=\varnothing$, x and y are different. So, $y\{x\leftarrow M\}=y$. The fourth equivalence appeals to the induction hypothesis.

- case where $\pi = M'(o).\tilde{P}'$:

$$
\begin{array}{rcl}
\mathcal{U}(M'(o).\tilde{P}',\sigma)\{x\leftarrow M\} & \equiv & (M'\sigma.\mathcal{U}(\tilde{P}',\sigma))\{x\leftarrow M\} \\
& \equiv & M'\sigma\{x\leftarrow M\}.\mathcal{U}(\tilde{P}',\sigma)\{x\leftarrow M\} \\
& \equiv & M'\sigma\{x\leftarrow M\}.\mathcal{U}(\tilde{P}',\sigma\{x\leftarrow M\}) \\
& \equiv & \mathcal{U}(M'(o).\tilde{P}',\sigma\{x\leftarrow M\})
\end{array}
$$

The first equivalence uses the definition of $\mathcal{U}$ and the fact that by hypothesis, o is equal to 0; the second one is simply the application of the substitution $\{x\leftarrow M\}$. The third equivalence is due to the induction hypothesis. Finally, the last equivalence is a direct consequence of the definition of $\mathcal{U}$ and of $o=0$.

■

Lemma A.15 *Let $\langle\tilde{P};\{x\leftarrow M\}\sigma\rangle$ be a normal closure such that all the offsets o occurring in $\tilde{P}$ are set to* 0. *Then $\mathcal{U}(\tilde{P},\{x\leftarrow M\}\sigma)\equiv\mathcal{U}(\tilde{P},\sigma)\{x\leftarrow M\sigma\}$.*

Proof The proof goes by induction on the length of the sequential substitution σ.

For σ being the empty substitution ι: $\mathcal{U}(\tilde{P},\{x\leftarrow M\}\iota)\equiv\mathcal{U}(\tilde{P},\iota\{x\leftarrow M\})$ since ι corresponds to the identity. So, by Lemma A.14, $\mathcal{U}(\tilde{P},\{x\leftarrow M\}\iota)\equiv\mathcal{U}(\tilde{P},\iota)\{x\leftarrow M\}$.

For σ being of the form $\sigma'\{y\leftarrow M'\}$:

$$
\begin{array}{rcl}
\mathcal{U}(\tilde{P},\{x\leftarrow M\}\sigma'\{y\leftarrow M'\}) & \equiv & \mathcal{U}(\tilde{P},\{x\leftarrow M\}\sigma')\{y\leftarrow M'\} \\
& \equiv & (\mathcal{U}(\tilde{P},\sigma')\{x\leftarrow M\sigma'\})\{y\leftarrow M'\}
\end{array}
$$

The first equivalence follows from Lemma A.14 and the second one from the induction hypothesis.

Now, the fact that $\langle\tilde{P};\{x\leftarrow M\}\sigma'\{y\leftarrow M'\}\rangle$ is normal implies that $x\neq y$ and that x does not occur in M'. Let us consider now the process $\mathcal{U}(\tilde{P},\sigma')\{x\leftarrow M\sigma'\}$. As $x\neq y$, the occurrences of y in $\mathcal{U}(\tilde{P},\sigma')$ are preserved in $\mathcal{U}(\tilde{P},\sigma')\{x\leftarrow M\sigma'\}$ and some new occurrences of y may appear in this latter, due to the possible occurrences of y in $M\sigma'$. As x does not occur in M', we can first replace $\mathcal{U}(\tilde{P},\sigma')$ the occurrences of y with M' and then, replace the occurrences of x with an expression L; this expression L is the expression $M\sigma$ in which the occurrences of y are replaced by M'. Hence,

$$(\mathcal{U}(\tilde{P},\sigma')\{x\leftarrow M\sigma'\})\{y\leftarrow M'\} \equiv (\mathcal{U}(\tilde{P},\sigma')\{y\leftarrow M'\})\{x\leftarrow M\sigma'\{y\leftarrow M'\}\}$$

By Lemma A.14, this latter is equivalent to $\mathcal{U}(\tilde{P},\sigma'\{y\leftarrow M'\})\{x\leftarrow M\sigma'\{y\leftarrow M'\}\}$ and so, to $\mathcal{U}(\tilde{P},\sigma)\{x\leftarrow M\sigma\}$. ∎

Lemma A.16 *Suppose $\langle\tilde{P};\sigma\rangle$ is a normal closure such that all the offsets o occurring in $\tilde{P}$ are set to 0 and x occurs neither in σ nor in $bn(\tilde{P})$. Then $\mathcal{U}(norm(\tilde{P},\{x\leftarrow M\}\sigma),\{x\leftarrow M\}\sigma) \equiv \mathcal{U}(\tilde{P},\sigma)\{x\leftarrow M\sigma\}$.*

Proof First, observe that normality of $\langle\tilde{P};\sigma\rangle$ and the assumption about x imply normality of $\langle norm(\tilde{P},\{x\leftarrow M\}\sigma);\{x\leftarrow M\}\sigma\rangle$. Therefore, by Lemma A.15, $\mathcal{U}(norm(\tilde{P},\{x\leftarrow M\}\sigma),\{x\leftarrow M\}\sigma) \equiv \mathcal{U}(norm(\tilde{P},\{x\leftarrow M\}\sigma),\sigma)\{x\leftarrow M\sigma\}$. So, it is enough to prove that

$$\mathcal{U}(norm(\tilde{P},\{x\leftarrow M\}\sigma),\sigma)\{x\leftarrow M\sigma\} \equiv \mathcal{U}(\tilde{P},\sigma)\{x\leftarrow M\sigma\}.$$

Let us consider two cases: $len(M,\sigma) \neq 0$ and $len(M,\sigma) = 0$. In the first case, $norm(\tilde{P},\{x\leftarrow M\}\sigma) = \tilde{P}$ and there is nothing to prove. In the second case, normality of $\langle\tilde{P};\sigma\rangle$ implies that $norm(\tilde{P},\{x\leftarrow M\}\sigma)$ differs from $\tilde{P}$ only by some occurrences of $x(0)$. The equivalence $\mathcal{U}(norm(\tilde{P},\{x\leftarrow M\}\sigma),\sigma)\{x\leftarrow M\sigma\} \equiv \mathcal{U}(\tilde{P},\sigma)\{x\leftarrow M\sigma\}$ follows then by induction on the structure of $M\sigma$ using the congruence rule (Struct ϵ). ∎

Restatement of Proposition 3.4 *Assume $\langle\tilde{P};\sigma\rangle$ is a normal closure. If $\langle\tilde{P};\sigma\rangle \to \langle\tilde{P}';\sigma'\rangle$ then $\mathcal{U}(\tilde{P},\sigma) \to \mathcal{U}(\tilde{P}',\sigma')$. If $\mathcal{U}(\tilde{P},\sigma) \to P'$ then there exists $\langle\tilde{P}';\sigma'\rangle$ such that $\langle\tilde{P};\sigma\rangle \to \langle\tilde{P}';\sigma'\rangle$ and $\mathcal{U}(\tilde{P}',\sigma') \equiv P'$.*

Proof The proof goes by induction over the structure of the context under which the reduction takes place.

If the context is empty, then for the first point, the reduction applied corresponds to one of the rules (Trans In), (Trans Out), (Trans Open) and (Trans I/O).

For the first point and the rule (Trans In):

$$\begin{array}{rcl}
\mathcal{U}(\{N[\tilde{Q} +\!\!+ \{\pi\}], M[\tilde{R}]\},\sigma) & \equiv & N\sigma[\mathcal{U}(\tilde{Q},\sigma) \mid \mathcal{U}(\{\pi\},\sigma)] \mid M\sigma[\mathcal{U}(\tilde{R},\sigma)] \\
& \equiv & n[\mathcal{U}(\tilde{Q},\sigma) \mid \mathcal{U}(\{\pi\},\sigma)] \mid m[\mathcal{U}(\tilde{R},\sigma)] \\
& \equiv & n[\mathcal{U}(\tilde{Q},\sigma) \mid in\, m.\mathcal{U}(\tilde{P},\sigma)] \mid m[\mathcal{U}(\tilde{R},\sigma)]
\end{array}$$

The first equivalence follows from the definition of $\mathcal{U}$. The second one is a consequence of the conditions of the rule (Trans In) and of Proposition A.6. The third equivalence follows from the conditions of the rule (Trans In) and from Proposition A.10.

On the other hand,

$$\begin{array}{rcl}\mathcal{U}(M[N[\tilde{Q} +\!\!+ \tilde{P}] +\!\!+ \tilde{R}], \sigma) & \equiv & M\sigma[N\sigma[\mathcal{U}(\tilde{Q}, \sigma) \mid \mathcal{U}(\tilde{P}, \sigma)] \mid \mathcal{U}(\tilde{R}, \sigma)] \\ & \equiv & m[n[\mathcal{U}(\tilde{Q}, \sigma) \mid \mathcal{U}(\tilde{P}, \sigma)] \mid \mathcal{U}(\tilde{R}, \sigma)]\end{array}$$

The first equivalence follows from the definition of $\mathcal{U}$ and the second one from the conditions of the rule (Trans In) and from Proposition A.6. Therefore, $\mathcal{U}(N[\tilde{Q} +\!\!+ \{\pi\}] +\!\!+ M[\tilde{R}], \sigma) \to \mathcal{U}(M[N[\tilde{Q} +\!\!+ \tilde{P}] +\!\!+ \tilde{R}], \sigma)$.

The proof is similar for the rules (Trans Out) and (Trans Open). Now, for the first point and the rule (Trans I/O): by the definition of $\mathcal{U}$, we have $\mathcal{U}(\{\langle M\rangle, (x).\tilde{P}\}, \sigma) \equiv \langle M\sigma\rangle \mid (x).\mathcal{U}(\tilde{P}, \sigma)$. Let $\tilde{P}'$ be $norm(\tilde{P}, \{x \leftarrow M\}\sigma)$. By Lemma A.15, the closure $\langle\{\langle M\rangle, (x).\tilde{P}\}; \sigma\rangle$ being normal, $\mathcal{U}(\tilde{P}', \{x \leftarrow M\}\sigma) \equiv \mathcal{U}(\tilde{P}', \sigma)\{x \leftarrow M\sigma\}$. Therefore, $\mathcal{U}(\{\langle M\rangle, (x).\tilde{P}\}, \sigma) \to \mathcal{U}(\tilde{P}', \{x \leftarrow M\}\sigma)$.

Let us consider now the second point with the assumption that the context is empty, that is the reduction is made by (Red In), (Red Out), (Red Open) or (Red I/O).

For the second point and the rule (Red In): let us assume that $\mathcal{U}(\tilde{S}, \sigma) \to S'$ by the rule (Red In). Therefore, $S' \equiv m[n[Q \mid P] \mid R]$ for some m, n, P, Q, R and $\mathcal{U}(\tilde{S}, \sigma) \equiv n[Q \mid in\, m.P] \mid m[R]$. So, by Proposition 3.1 and Proposition A.10, there exists N, M, L', $\tilde{P}, \tilde{P}', \tilde{Q}, \tilde{R}$ such that $\tilde{S} = \{N[\tilde{Q} +\!\!+ \{L'(o).\tilde{P}'\}], M[\tilde{R}]\}$, $N\sigma = n$, $M\sigma = m$, $\mathcal{U}(\tilde{Q}, \sigma) \equiv Q$, $\mathcal{U}(\tilde{R}, \sigma) \equiv R$, $split(L'(o).\tilde{P}') = (in\, m, \tilde{P})$ and $\mathcal{U}(\tilde{P}, \sigma) \equiv P$. Using Proposition A.6, we have $nam(M, \sigma) = m$ and $nam(N, \sigma) = n$. So, by definition for (Red In),

$$\langle\tilde{S}; \sigma\rangle \to \langle\{M[\{N[\tilde{P} +\!\!+ \tilde{Q}]\} +\!\!+ \tilde{R}]\}; \sigma\rangle$$

and furthermore,

$$\begin{array}{rcl}\mathcal{U}(M[N[\tilde{Q} +\!\!+ \tilde{P}] +\!\!+ \tilde{R}], \sigma) & \equiv & m[n[\mathcal{U}(\tilde{Q}, \sigma) \mid \mathcal{U}(\tilde{P}, \sigma)] \mid \mathcal{U}(\tilde{R}, \sigma)] \\ & \equiv & m[n[Q \mid P] \mid R] \equiv S'\end{array}$$

The proof is similar for the rules (Red Out) and (Red Open). Now, for the second point and the rule (Red I/O): let us assume that $\mathcal{U}(\tilde{S}, \sigma) \to S'$ by the rule (Red I/O). Therefore, $S' \equiv P\{x \leftarrow M\}$ and $\mathcal{U}(\tilde{S}, \sigma) \equiv (x).P \mid \langle M\rangle$. So, by Proposition 3.1, there exists $M', \tilde{P}$ such that $\tilde{S} = \{\langle M'\rangle, (x).\tilde{P}\}$, $M'\sigma = M$ and $\mathcal{U}(\tilde{P}, \sigma) \equiv P$. Therefore, $\langle\tilde{S}; \sigma\rangle \to \langle\tilde{P}'; \{x \leftarrow M'\}\sigma\rangle$ where $\tilde{P}' = norm(\tilde{P}, \{x \leftarrow M\}\sigma)$. Furthermore, $\langle\{\langle M'\rangle, (x).\tilde{P}\}; \sigma\rangle$ being normal, by Lemma A.16

$$\begin{array}{rcl}\mathcal{U}(\tilde{P}', \{x \leftarrow M'\}\sigma) & \equiv & \mathcal{U}(\tilde{P}, \sigma)\{x \leftarrow M'\sigma\} \\ & \equiv & P\{x \leftarrow M\}.\end{array}$$

Now, we investigate the case where the context of reduction is non-empty: for the first point, the rule used for reduction is either (Trans Par) or (Trans Amb).

For the rule (Trans Amb): if $\langle\tilde{P}; \sigma\rangle \to \langle\tilde{P}'; \sigma'\rangle$ then $\langle M[\tilde{P}]; \sigma\rangle \to \langle M[\tilde{P}']; \sigma'\rangle$. In this case, $\mathcal{U}(M[\tilde{P}], \sigma) = M\sigma[\mathcal{U}(\tilde{P}, \sigma)]$ and $\mathcal{U}(M[\tilde{P}'], \sigma') = M\sigma'[\mathcal{U}(\tilde{P}', \sigma')]$. By

A.13, either $\sigma' = \sigma$ or $\sigma' = \{x \leftarrow L\}\sigma$. In this last case, x is bound in $\tilde{P}$ and thus, by normality, x does not occur in M. So in both cases, $M\sigma' = M\sigma$. Moreover, by the rule (Red Amb), $M\sigma[\mathcal{U}(\tilde{P},\sigma)] \to M\sigma[\mathcal{U}(\tilde{P}',\sigma')]$. So, $\mathcal{U}(M[\tilde{P}],\sigma) \to \mathcal{U}(M[\tilde{P}'],\sigma')$

For the rule (Trans Par): if $\langle \tilde{P};\sigma\rangle \to \langle \tilde{P}';\sigma'\rangle$ then $\langle \tilde{P} +\!\!+ \tilde{Q};\sigma\rangle \to \langle \tilde{P}' +\!\!+ \tilde{Q};\sigma'\rangle$. In this case, $\mathcal{U}(\tilde{P} +\!\!+ \tilde{Q},\sigma) \equiv \mathcal{U}(\tilde{P},\sigma) \mid \mathcal{U}(\tilde{Q},\sigma)$ and $\mathcal{U}(\tilde{P}' +\!\!+ \tilde{Q},\sigma') \equiv \mathcal{U}(\tilde{P}',\sigma') \mid \mathcal{U}(\tilde{Q},\sigma')$. By A.13, either $\sigma' = \sigma$ or $\sigma' = \{x \leftarrow M\}\sigma$. In this last case, x is bound in $\tilde{P}$ and thus, by normality does not occur in $\tilde{Q}$. So, in both cases, we have $\mathcal{U}(\tilde{Q},\sigma') \equiv \mathcal{U}(\tilde{Q},\sigma)$. Moreover, by the rule (Red Par), $\mathcal{U}(\tilde{P},\sigma) \mid \mathcal{U}(\tilde{Q},\sigma) \to \mathcal{U}(\tilde{P}',\sigma') \mid \mathcal{U}(\tilde{Q},\sigma)$. So, $\mathcal{U}(\tilde{P} +\!\!+ \tilde{Q},\sigma) \to \mathcal{U}(\tilde{P}' +\!\!+ \tilde{Q},\sigma)$.

For the second point, the rule used for reduction is either (Red Par) or (Red Amb).

For (Red Amb): let us assume that $\mathcal{U}(\tilde{S},\sigma) \to S'$ by (Red Amb). We have $S' = n[P']$ and $\mathcal{U}(\tilde{S},\sigma) \equiv n[P]$. So, by Proposition 3.1, there exists N,σ such that $\tilde{S}$ is a singleton $\{\pi\}$, $\pi = N[\tilde{P}]$, $N\sigma = n$ and $\mathcal{U}(\tilde{P},\sigma) \equiv P$. By hypothesis $P \to P'$, so $\mathcal{U}(\tilde{P},\sigma) \to P'$. By induction hypothesis, there exists $\tilde{P}',\sigma'$ such that $\langle \tilde{P};\sigma\rangle \to \langle \tilde{P}';\sigma'\rangle$ and $\mathcal{U}(\tilde{P}',\sigma') \equiv P'$. Then by the rule (Trans Amb), $\langle\{N[\tilde{P}]\};\sigma\rangle \to \langle\{N[\tilde{P}']\};\sigma'\rangle$; so, $\langle\tilde{S};\sigma\rangle \to \langle\{N[\tilde{P}']\};\sigma'\rangle$. Finally, $\mathcal{U}(\{N[\tilde{P}']\},\sigma') \equiv N\sigma'[\mathcal{U}(\tilde{P}',\sigma')]$. By Lemma A.13, either $\sigma = \sigma'$ or $\sigma' = \{x \leftarrow M\}\sigma$ with x a bound variable in $\tilde{P}$. By normality x does not belong to N, so $N\sigma' = N\sigma = n$. Therefore, $N\sigma'[\mathcal{U}(\tilde{P}',\sigma')] \equiv n[\mathcal{U}(\tilde{P}',\sigma')] \equiv n[P'] \equiv S'$.

For (Red Par): let us assume that $\mathcal{U}(\tilde{S},\sigma) \to S'$ by (Red Par). We have $S' = P' \mid Q$ and $\mathcal{U}(\tilde{S},\sigma) \equiv P \mid Q$. So, by Proposition 3.1, there exists $\tilde{P},\tilde{Q}$ such that $\tilde{S} = \tilde{P} +\!\!+ \tilde{Q}$, $\mathcal{U}(\tilde{P},\sigma) \equiv P$ and $\mathcal{U}(\tilde{Q},\sigma) \equiv Q$. By hypothesis, $P \to P'$, so $\mathcal{U}(\tilde{P},\sigma) \to P'$. By induction hypothesis, there exists $\tilde{P}',\sigma'$ such that $\langle \tilde{P};\sigma\rangle \to \langle \tilde{P}';\sigma'\rangle$ and $\mathcal{U}(\tilde{P}',\sigma') \equiv P'$. Then by the rule (Trans Par), $\langle \tilde{P} +\!\!+ \tilde{Q};\sigma\rangle \to \langle \tilde{P}' +\!\!+ \tilde{Q};\sigma'\rangle$; so, $\langle\tilde{S};\sigma\rangle \to \langle \tilde{P}' +\!\!+ \tilde{Q};\sigma'\rangle$. Finally, $\mathcal{U}(\tilde{P}' +\!\!+ \tilde{Q},\sigma') \equiv \mathcal{U}(\tilde{P}',\sigma') \mid \mathcal{U}(\tilde{Q},\sigma')$. Now, by Lemma A.13, either $\sigma = \sigma'$ or $\sigma' = \{x \leftarrow M\}\sigma$ with x a bound variable in $\tilde{P}$. By normality x does not occur in $\tilde{Q}$; so, $\mathcal{U}(\tilde{Q},\sigma') \equiv \mathcal{U}(\tilde{Q},\sigma)$. Therefore, $\mathcal{U}(\tilde{P}' +\!\!+ \tilde{Q},\sigma') \equiv P' \mid Q \equiv S'$. ■

A.6 Proof of Proposition 3.9

Restatement of Proposition 3.9 *The model checking algorithm described in Section 3.3 preserves the normality of* $\mathit{Check}(\tilde{P},\sigma,\mathcal{A})$.

Proof By case inspection of the algorithm, we show that if $\mathit{Check}(\tilde{P},\sigma,\mathcal{A})$ is normal in the left-hand side of equality then any expression $\mathit{Check}(\tilde{P}',\sigma',\mathcal{A}')$ occurring in the right-hand side is also normal.

- for the Boolean connectives $\neg, \vee$: since in any case, $\tilde{P}' = \tilde{P}$ and $\sigma = \sigma'$ and $\mathcal{A}'$ is a closed formula such that and $\mathit{fn}(\mathcal{A}') \subseteq \mathit{fn}(\mathcal{A})$, this is straightforward.

- for the ambient match $\mathcal{A} = n[\mathcal{A}']$: in this case, $\tilde{P} = \{n[\tilde{Q}]\}$ and $\sigma = \sigma'$. By Proposition A.1 the closure $\langle\tilde{Q};\sigma\rangle$ is normal. The remaining conditions are fulfilled since $\mathit{bn}(P') = \mathit{bn}(P)$, $\sigma' = \sigma$ and for the closed formula $\mathcal{A}'$ $\mathit{fn}(\mathcal{A}') \subseteq \mathit{fn}(\mathcal{A})$.

- for the composition match $\mathcal{A} = \mathcal{A}' \mid \mathcal{A}''$: this proof is similar to the previous case.

- for the existential quantification $\exists x.\mathcal{A}$: in this case, $\tilde{P}' = \tilde{P}$ and $\sigma = \sigma'$ and the fact that $\mathcal{A}\{x{\leftarrow}m_i\}$ is closed is straightforward. So, it is sufficient to show that whatever the ambient name m_i is, $\mathit{fn}(\mathcal{A}\{x{\leftarrow}m_i\}) \cap (\mathit{bn}(\tilde{P}) \cup \mathit{dom}(\sigma)) = \varnothing$. By noticing that $\mathit{fn}(\mathcal{A}\{x{\leftarrow}m_i\})$ is either equal to $\mathit{fn}(\exists x.\mathcal{A})$ or to $\mathit{fn}(\exists x.\mathcal{A}) \cup \{m_i\}$ and using the normality for $\mathit{Check}(\tilde{P}, \sigma, \exists x.\mathcal{A})$, this amounts to prove that $m_i \notin \mathit{bn}(\tilde{P}) \cup \mathit{dom}(\sigma)$. According to the value of m_i:

  - for $mi = m_0$: straightforward.
  - $m_i \in \mathit{fn}(\tilde{P}, \sigma) \cup \mathit{fn}(\mathcal{A})$: let us assume that $m_i \in \mathit{fn}(\mathcal{A})$. Then, $m_i \in \mathit{fn}(\exists x.\mathcal{A})$. So, by normality of $\mathit{Check}(\tilde{P}, \sigma, \exists x.\mathcal{A})$, $m_i \notin \mathit{bn}(\tilde{P}) \cup \mathit{dom}(\sigma)$. Let us assume now that $m_i \in \mathit{fn}(\tilde{P}, \sigma)$: by definition, $m_i \notin \mathit{dom}(\sigma)$. Now, by normality of $\langle \tilde{P}; \sigma \rangle$, since $m_i \in \mathit{fn}(\tilde{P})$ or $m_i \in \mathit{names}(\sigma)$, $m_i \notin \mathit{bn}(\tilde{P})$.

- for the sometime modality $\Diamond\mathcal{A}$:

  - case where $\mathit{Check}(\tilde{P}', \sigma', \mathcal{A}') = \mathit{Check}(\tilde{P}, \sigma, \mathcal{A})$: obvious since $\mathit{fn}(\Diamond\mathcal{A}) = \mathit{fn}(\mathcal{A})$.
  - case where $\mathit{Check}(\tilde{P}', \sigma', \mathcal{A}') = \mathit{Check}(\tilde{P}', \sigma', \Diamond\mathcal{A})$ with $\langle \tilde{P}; \sigma \rangle \rightarrow \langle \tilde{P}'; \sigma' \rangle$: by Proposition 3.2(2), $\langle \tilde{P}'; \sigma' \rangle$ is normal. Now, according to Lemma A.13:
    - $\sigma = \sigma'$, $\mathit{bn}(\tilde{P}) = \mathit{bn}(\tilde{P}')$ and $\mathit{fn}(\Diamond\mathcal{A}) = \mathit{fn}(\mathcal{A})$: in this case, the requirement is trivially satisfied.
    - $\sigma' = \{x{\leftarrow}M\}\sigma$, $\mathit{bn}(\tilde{P}) = \mathit{bn}(\tilde{P}') \cup \{x\}$: by hypothesis, $\mathit{fn}(\Diamond\mathcal{A}) \cap (\mathit{bn}(\tilde{P}) \cup \mathit{dom}(\sigma)) = \varnothing$. So, $\mathit{fn}(\Diamond\mathcal{A}) \cap (\mathit{bn}(\tilde{P}') \cup \mathit{dom}(\sigma')) = \varnothing$.

- for the somewhere modality $\diamond\mathcal{A}$:

  - case where $\mathit{Check}(\tilde{P}', \sigma', \mathcal{A}') = \mathit{Check}(\tilde{P}, \sigma, \mathcal{A})$: obvious since $\mathit{fn}(\Diamond\mathcal{A}) = \mathit{fn}(\mathcal{A})$.
  - case where $\mathit{Check}(\tilde{P}', \sigma', \mathcal{A}') = \mathit{Check}(\tilde{P}', \sigma', \Diamond\mathcal{A})$ with $\langle \tilde{P}; \sigma \rangle \downarrow \langle \tilde{P}'; \sigma' \rangle$: by Proposition 3.2, $\langle \tilde{P}'; \sigma' \rangle$ is normal. The last condition holds since $\sigma' = \sigma$ and $\mathit{fn}(\tilde{P}') \subseteq \mathit{fn}(\tilde{P})$.

- for the location adjunct modality $\mathcal{A}@n$: from the hypothesis of normality for $\mathit{Check}(\tilde{P}, \sigma, \mathcal{A}@n)$, since $n \in \mathit{fn}(\mathcal{A})$, $n \notin \mathit{bn}(\tilde{P})$. Therefore, by Proposition A.1, $\langle n[P]; \sigma \rangle$ is normal. Moreover, $\mathcal{A}$ is a closed formula. Finally, by hypothesis, $\mathit{fn}(\mathcal{A}@n) \cap (\mathit{bn}(\tilde{P}) \cup \mathit{dom}(\sigma)) = \varnothing$, and $\mathit{bn}(\tilde{P}) = \mathit{bn}(n[\tilde{P}])$, $\mathit{fn}(\mathcal{A}) \subseteq \mathit{fn}(\mathcal{A}@n)$. So, $\mathit{fn}(\mathcal{A}) \cap (\mathit{bn}(n[\tilde{P}]) \cup \mathit{dom}(\sigma)) = \varnothing$. ■

A.7 Proof of Proposition 3.10

The correctness of our algorithm, Proposition 3.10, is a corollary of Lemma A.18 below, which itself depends on the following fact.

Lemma A.17 (Cardelli and Gordon (2000b)) *For any ambient process P and any ambient formula $\mathcal{A}$, let $\{m_1, \ldots, m_k\} = fn(P) \cup fn(\mathcal{A})$ and suppose $m_0 \notin \{m_1, \ldots, m_k\}$. Then $P \models \exists x.\mathcal{A}$ iff $P \models \mathcal{A}\{x \leftarrow m_i\}$ for some i in $0 \ldots k$.*

Lemma A.18 *For any normal closure $\langle \tilde{P}; \sigma \rangle$, $\mathcal{U}(\tilde{P}, \sigma) \models \mathcal{A}$ if and only if $Check(\tilde{P}, \sigma, \mathcal{A}) = \mathbf{T}$.*

Proof The proof goes by induction on the structure of the ambient formula $\mathcal{A}$:

- the base case $\mathcal{A} = \mathbf{T}$ is trivial. The other base case $\mathcal{A} = \mathbf{0}$ is a consequence of Proposition 3.1.

- for Boolean connectives $\neg, \wedge$, this is obvious from the induction hypothesis and the algorithm.

- for the ambient match $\mathcal{A} = n[\mathcal{A}']$: according to the algorithm, we have $Check(\prod_{i \in 1 \ldots k} \pi_i, \sigma, n[\mathcal{A}']) = \mathbf{T}$ iff there exists $\tilde{Q}$ and M such that $k = 1$, $\pi_1 = M[\tilde{Q}]$, $nam(M, \sigma) = n$ and $Check(\tilde{Q}, \sigma, \mathcal{A}') = \mathbf{T}$. Then, by Proposition 3.1, $\mathcal{U}(\prod_{i \in 1 \ldots k} \pi_i, \sigma) \equiv n[\mathcal{U}(\tilde{Q}, \sigma)]$. By induction hypothesis, $Check(\tilde{Q}, \sigma, \mathcal{A}') = \mathbf{T}$ is equivalent to $\mathcal{U}(\tilde{Q}, \sigma) \models \mathcal{A}'$. So, it is equivalent to $\mathcal{U}(\prod_{i \in 1 \ldots k} \pi_i, \sigma) \models n[\mathcal{A}']$.

- for the composition match $\mathcal{A} = \mathcal{A}' \mid \mathcal{A}''$: according to the algorithm, we have $Check(\prod_{i \in 1 \ldots k} \pi_i, \sigma, \mathcal{A}' \mid \mathcal{A}'') = \mathbf{T}$ iff there exists I, J such that $I \cup J = 1 \ldots k$, $I \cap J = \varnothing$, $Check(\prod_{i \in I} \pi_i, \sigma, \mathcal{A}') = \mathbf{T}$ and $Check(\prod_{j \in J} \pi_j, \sigma, \mathcal{A}'') = \mathbf{T}$. Now, using the induction hypothesis, $Check(\prod_{i \in I} \pi_i, \sigma, \mathcal{A}') = \mathbf{T}$ and $Check(\prod_{j \in J} \pi_j, \sigma, \mathcal{A}'') = \mathbf{T}$ are equivalent respectively to $\mathcal{U}(\prod_{i \in I} \pi_i, \sigma) \models \mathcal{A}'$ and to $\mathcal{U}(\prod_{j \in J} \pi_j, \sigma) \models \mathcal{A}''$. Finally, by Proposition 3.1, it is equivalent to $\mathcal{U}(\prod_{i \in 1 \ldots k} \pi_i, \sigma) \models \mathcal{A}' \mid \mathcal{A}''$.

- for the existential quantification $\exists x.\mathcal{A}$: let us assume $Check(\tilde{P}, \sigma, \exists x.\mathcal{A}) = \mathbf{T}$. Let $\{m_1, \ldots, m_k\} = fn(\tilde{P}, \sigma) \cup fn(\mathcal{A})$ and m_0, an ambient name such that $m_0 \notin \{m_1, \ldots, m_k\} \cup bn(\tilde{P}) \cup dom(\sigma)$. From the algorithm, this implies that there exists i such that $Check(\tilde{P}, \sigma, \mathcal{A}\{x \leftarrow m_i\}) = \mathbf{T}$. So, by the induction hypothesis, $\mathcal{U}(\tilde{P}, \sigma) \models \mathcal{A}\{x \leftarrow m_i\}$. Now, according to the value of m_i:

  - $m_i \in \{m_1, \ldots, m_k\} \cap (fn(\mathcal{A}) \cup fn(\mathcal{U}(\tilde{P}, \sigma)))$: by Lemma A.17, we have $\mathcal{U}(\tilde{P}, \sigma) \models \exists x.\mathcal{A}$.
  - $m_i \in \{m_1, \ldots, m_k\}$ and $m_i \notin (fn(\mathcal{A}) \cup fn(\mathcal{U}(\tilde{P}, \sigma)))$: by Lemma A.17, we have $\mathcal{U}(\tilde{P}, \sigma) \models \exists x.\mathcal{A}$.
  - $m_i \notin \{m_1, \ldots, m_k\}$: it is obvious then that $m_i \notin fn(\mathcal{A}) \cup fn(\mathcal{U}(\tilde{P}, \sigma))$. So, by Lemma A.17, we have $\mathcal{U}(\tilde{P}, \sigma) \models \exists x.\mathcal{A}$.

  Conversely, let us assume that $\mathcal{U}(\tilde{P}, \sigma) \models \exists x.\mathcal{A}$. From Lemma A.17, this is equivalent to that for $\{m_1, \ldots, m_k\} = fn(\mathcal{U}(\tilde{P}, \sigma)) \cup fn(\mathcal{A})$ and for any arbitrary m_0 such that $m_0 \notin \{m_1, \ldots, m_k\}$, there exists i such that $\mathcal{U}(\tilde{P}, \sigma) \models \mathcal{A}\{x \leftarrow m_i\}$. This latter is equivalent to that $Check(\tilde{P}, \sigma, \mathcal{A}\{x \leftarrow m_i\}) = \mathbf{T}$ by induction hypothesis. Now according to the value of m_i:

- $m_i \in fn(\mathcal{U}(\tilde{P},\sigma)) \cup fn(\mathcal{A})$: in this case $m_i \in fn(\tilde{P},\sigma) \cup fn(\mathcal{A})$. So, by the algorithm, $Check(\tilde{P},\sigma,\exists x.\mathcal{A}) = \mathbf{T}$.
- $m_i \notin fn(\mathcal{U}(\tilde{P},\sigma)) \cup fn(\mathcal{A})$ and $m_i \in fn(\tilde{P},\sigma) \cup fn(\mathcal{A})$: once again, by the algorithm, $Check(\tilde{P},\sigma,\exists x.\mathcal{A}) = \mathbf{T}$.
- $m_i \notin fn(\tilde{P},\sigma) \cup fn(\mathcal{A})$: so, $m_i = m_0$. Since m_0 can be chosen arbitrarily, one can assume moreover that $m_i \notin bn(\tilde{P}) \cup dom(\sigma)$. So, by the algorithm, $Check(\tilde{P},\sigma,\exists x.\mathcal{A}) = \mathbf{T}$.

- for the Sometime modality $\Diamond\mathcal{A}$: $\mathcal{U}(\tilde{P},\sigma) \models \Diamond\mathcal{A}$ is by definition equivalent to the fact that there exists P', n such that $\mathcal{U}(\tilde{P},\sigma) \rightarrow^n P'$ and $P' \models \mathcal{A}$. By Proposition 3.4, this latter implies that there exists $\tilde{P}',\sigma'$ such that $\mathcal{U}(\tilde{P},\sigma) \rightarrow^n \mathcal{U}(\tilde{P}',\sigma')$ and $\mathcal{U}(\tilde{P}',\sigma') \equiv P'$ and thus, $\mathcal{U}(\tilde{P}',\sigma') \models \mathcal{A}$. Therefore, by induction hypothesis, this implies $Check(\tilde{P}',\sigma',\mathcal{A}) = \mathbf{T}$. Now, let us show by induction over n that $\mathcal{U}(\tilde{P},\sigma) \rightarrow^n \mathcal{U}(\tilde{P}',\sigma')$ and $\mathcal{U}(\tilde{P}',\sigma') \models \mathcal{A}$ implies $Check(\tilde{P},\sigma,\Diamond\mathcal{A}) = \mathbf{T}$.

  For $n = 0$: $\langle\tilde{P};\sigma\rangle = \langle\tilde{P}';\sigma'\rangle$ and $Check(\tilde{P},\sigma,\Diamond\mathcal{A}) = Check(\tilde{P},\sigma,\mathcal{A}) = \mathbf{T}$.

  For $0 < n$: in this case, by Proposition 3.4, there exists $\tilde{P}'',\sigma''$ such that $\langle\tilde{P};\sigma\rangle \rightarrow \langle\tilde{P}'';\sigma''\rangle \rightarrow^{n-1} \langle\tilde{P}';\sigma'\rangle$. So, by induction hypothesis using that $Check(\tilde{P}',\sigma',\mathcal{A}) = \mathbf{T}$, $Check(\tilde{P}'',\sigma'',\Diamond\mathcal{A}) = \mathbf{T}$. Since $\langle\tilde{P};\sigma\rangle \rightarrow \langle\tilde{P}'';\sigma''\rangle$, by the algorithm we have $Check(\tilde{P},\sigma,\Diamond\mathcal{A}) = \mathbf{T}$.

  Conversely, let us assume that $Check(\tilde{P},\sigma,\Diamond\mathcal{A}) = \mathbf{T}$ and let us show that there exists P', n such that $\mathcal{U}(\tilde{P},\sigma) \rightarrow^n P'$ and $P' \models \mathcal{A}$. The proof goes by induction on m the number of recursive calls of $Check(\tilde{P}',\sigma',\Diamond\mathcal{A}) = \mathbf{T}$.

  For $m = 0$: in this case, $Check(\tilde{P},\sigma,\Diamond\mathcal{A}) = \mathbf{T}$ since $Check(\tilde{P},\sigma,\mathcal{A}) = \mathbf{T}$. Then by induction hypothesis on the structure of the formula, $\mathcal{U}(\tilde{P},\sigma) \models \mathcal{A}$. So, we can choose $P' = \mathcal{U}(\tilde{P},\sigma)$ and $n = 0$.

  For $m > 0$: in this case, $Check(\tilde{P},\sigma,\Diamond\mathcal{A}) = \mathbf{T}$ due to the fact that for some $\langle\tilde{P}';\sigma'\rangle$ such that $\langle\tilde{P};\sigma\rangle \rightarrow \langle\tilde{P}';\sigma'\rangle$, $Check(\tilde{P}',\sigma',\mathcal{A}) = \mathbf{T}$. By the induction hypothesis, on the number of recursive calls, we have that there exists P', n such that $\mathcal{U}(\tilde{P}',\sigma') \rightarrow^n P'$ and $P' \models \mathcal{A}$. By Proposition 3.4, we have $\mathcal{U}(\tilde{P},\sigma) \rightarrow \mathcal{U}(\tilde{P}',\sigma')$. So, $\mathcal{U}(\tilde{P},\sigma) \rightarrow^{n+1} P'$ and $P' \models \mathcal{A}$.

- for the Somewhere modality $\diamond\mathcal{A}$: the proof is similar to the previous case using Proposition 3.3 instead of Proposition 3.4.

- for the location adjunct modality $\mathcal{A}@n$: by definition, $\mathcal{U}(\tilde{P},\sigma) \models \mathcal{A}@n$ iff $n[\mathcal{U}(\tilde{P},\sigma)] \models \mathcal{A}$. By assumption n does not belong to $dom(\sigma)$. So, from the definition for $\mathcal{U}$, $n[\mathcal{U}(\tilde{P},\sigma)] = \mathcal{U}(n[\tilde{P}],\sigma)$. So, $n[\mathcal{U}(\tilde{P},\sigma)] \models \mathcal{A}$ is equivalent to that $\mathcal{U}(n[\tilde{P}],\sigma) \models \mathcal{A}$. Using the induction hypothesis, this latter is equivalent to $Check(n[\tilde{P}],\sigma,\mathcal{A}) = \mathbf{T}$, and thus by the algorithm to $Check(\tilde{P},\sigma,\mathcal{A}@n) = \mathbf{T}$. ■

Restatement of Proposition 3.10 *For all processes P and closed formulas $\mathcal{A}$, we have $P \models \mathcal{A}$ if and only if $Check(\mathcal{F}(P),\iota,\mathcal{A}) = \mathbf{T}$.*

Proof As the closure $\langle\mathcal{F}(P);\iota\rangle$ is normal, this follows from Lemma A.18. ■

B Hardness Proofs

This appendix contains proofs of results stated in Section 4.

B.1 Proof of Lemma 4.1

Lemma 4.1 is the crux of correctness for the encoding from Section 4.1 of QBF satisfaction in the full calculus and logic.

Restatement of Lemma 4.1 *Consider a closed quantified boolean formula φ and its encoding $[\![\varphi]\!]$ in the ambient logic. The formula φ is valid if and only if the model checking problem $\mathbf{0} \models [\![\varphi]\!]$ holds.*

Proof Let us denote $C_1 \wedge \ldots \wedge C_k$ by ψ. We consider a closed QBF formula $Q_1 v_1 \ldots Q_n v_n \psi$. We are going to show that for any $0 \leq m \leq n$, denoting φ' the formula $Q_{m+1} v_{m+1} \ldots Q_n v_n\, \psi$,

$$v_1 \mapsto t_1, \ldots, v_m \mapsto t_m \models \varphi' \qquad \text{iff} \qquad \mathbf{0} \models [\![\varphi']\!]\{v_1 \leftarrow t_1, \ldots, v_m \leftarrow t_m\}$$

Note that this statement obviously implies Lemma 4.1.

The proof of this statement goes by induction on the number l of variables that are quantified in φ'.

For the base case $l = 0$: $v_1 \mapsto t_1, \ldots, v_n \mapsto t_n \models \psi$ iff for each C_i, there exists ℓ_j in C_i such that $t_j = tt$ iff $l_j = v_j$ and $t_j = f\!f$ iff $\ell_j = \overline{v_j}$. This is equivalent to saying that for each C_i, there exists ℓ_j in C_i such that $\mathbf{0} \models [\![l_j]\!]\{v_1 \leftarrow t_1, \ldots, v_n \leftarrow t_n\}$, which is equivalent to $\mathbf{0} \models \psi\{v_1 \leftarrow t_1, \ldots, v_n \leftarrow t_n\}$.

For the induction step $0 < l \leq n$: let us denote $\mathcal{M}$ the interpretation $v_1 \mapsto t_1, \ldots, v_{n-l} \mapsto t_{n-l}$, σ the corresponding substitution $\{v_1 \leftarrow t_1, \ldots, v_{n-l} \leftarrow t_{n-l}\}$ and φ' the formula $Q_{n-l+2} v_{n-l+2} \ldots Q_n v_n\, \psi$. Assuming that the statement holds for $l-1$, let us consider $\mathcal{M} \models Q_{n-l+1} v_{n-l+1} \varphi'$.

By case distinction over Q_{n-l+1}:

Case where $Q_{n-l+1} = \exists$: in this case, either $\mathcal{M}, v_{n-l+1} \mapsto tt \models \varphi'$ or $\mathcal{M}, v_{n-l+1} \mapsto f\!f \models \varphi'$. By induction hypothesis, this is equivalent to that either $\mathbf{0} \models [\![\varphi']\!]\sigma\{v_{n-l+1} \leftarrow tt\}$ or $\mathbf{0} \models [\![\varphi']\!]\sigma\{v_{n-l+1} \leftarrow f\!f\}$. This latter is equivalent to $\mathbf{0} \models \exists v_{n-l+1} \in \{tt, f\!f\}.[\![\varphi']\!]\sigma$ which is equivalent by definition of the encoding to $\mathbf{0} \models [\![Q_{n-l+1} v_{n-l+1} \varphi']\!]\sigma$.

Case where $Q_{n-l+1} = \forall$: this case is similar to the previous one. ∎

B.2 Proof of Lemma 4.3

Lemma 4.3 is the crux of correctness for the encoding from Section 4.2 of QBF satisfaction in the calculus of mobile ambients without I/O.

To prove Lemma 4.3, let us first fix some notations and prove some auxiliary lemmas.

For a given closed QBF formula $\varphi = Q_1 v_1 \dots Q_n v_n \psi$ in prenex and conjunctive normal form, we denote ψ by $C_1 \wedge \dots \wedge C_k$ and define for all $0 \leq i \leq n$

$$\begin{array}{l} V_i \triangleq v_i[pos[]] \mid v_i[neg[]] \\ V_i^{tt} \triangleq v_i[pos[] \mid v_i'[]] \mid v_i[neg[]] \\ V_i^{ff} \triangleq v_i[pos[]] \mid v_i[neg[] \mid v_i'[]] \end{array}$$

For all $0 \leq m \leq n$, $\mathcal{M}$ being equal to $v_1 \mapsto t_1, \dots, v_m \mapsto t_m$,

$$\varphi_m \triangleq Q_{m+1} v_{m+1} \dots Q_n v_n \psi$$

$$P_{\mathcal{M}} \triangleq V_1^{t_1} \mid \dots \mid V_m^{t_m} \mid V_{m+1} \mid \dots \mid V_n \mid P^{\varphi_m}$$

assuming that $[\![\varphi_m]\!] = (P^{\varphi_m}, \mathcal{A}^{\varphi_m})$.

It should be noticed that due to the definition of $[\![\ ]\!]$, for all $0 \leq m < n$, $P^{\varphi_m} \models v'_{m+1}[\mathbf{T}]$ and $P^{\varphi_n} \models end[\mathbf{T}]$.

Lemma B.1 *For all $0 \leq m < n$,*

$$P_{\mathcal{M}} \rightarrow^3 P_{\mathcal{M}, v_{m+1} \mapsto tt}$$

$$P_{\mathcal{M}} \rightarrow^3 P_{\mathcal{M}, v_{m+1} \mapsto ff}$$

and there does not exist P' such that $P' \not\equiv P_{\mathcal{M}, v_{m+1} \mapsto tt}$, $P' \not\equiv P_{\mathcal{M}, v_{m+1} \mapsto ff}$ and $P_{\mathcal{M}} \rightarrow^3 P'$.

Proof For $m < n-1$, we consider $\mathcal{M}$ to be $v_1 \mapsto t_1, \dots, v_m \mapsto t_m$ and we have $\varphi_m = Q_{m+1} v_{m+1} \dots Q_n v_n \psi$. Whatever Q_{m+1} is, by definition of *enc*,

$$P^{\varphi_m} = v'_{m+1}[in\ v_{m+1}.v_{m+2}[out\ v'_{m+1}.out\ v_{m+1}.R^{\varphi_{m+1}}]]$$

for $P^{\varphi_{m+1}} = v'_{m+1}[R^{\varphi_{m+1}}]$. Now from the process $P_{\mathcal{M}}$ equal to

$$\begin{array}{l} V_1^{t_1} \mid \dots \mid V_m^{t_m} \mid V_{m+2} \mid \dots \mid V_n \mid v_{m+1}[pos[]] \mid v_{m+1}[neg[]] \mid \\ v'_{m+1}[in\ v_{m+1}.v'_{m+2}[out\ v'_{m+1}.out\ v_{m+1}.R^{\varphi_{m+1}}]] \end{array}$$

only two reduction steps are possible leading either to

$$P_{\mathcal{M}}^{pos} \equiv \begin{array}{l} V_1^{t_1} \mid \dots \mid V_m^{t_m} \mid V_{m+2} \mid \dots \mid V_n \mid v_{m+1}[pos[]] \mid \\ v_{m+1}[neg[] \mid v'_{m+1}[v'_{m+2}[out\ v'_{m+1}.out\ v_{m+1}.R^{\varphi_{m+1}}]]] \end{array}$$

or to

$$P_{\mathcal{M}}^{neg} \equiv \begin{array}{l} V_1^{t_1} \mid \dots \mid V_m^{t_m} \mid V_{m+2} \mid \dots \mid V_n \mid v_{m+1}[neg[]] \mid \\ v_{m+1}[pos[] \mid v'_{m+1}[v'_{m+2}[out\ v'_{m+1}.out\ v_{m+1}.R^{\varphi_{m+1}}]]] \end{array}$$

Now, we have from each of $P_{\mathcal{M}}^{pos}$ and $P_{\mathcal{M}}^{neg}$ two deterministic reduction steps:

$$P_{\mathcal{M}}^{pos} \rightarrow \begin{array}{l} V_1^{t_1} \mid \dots \mid V_m^{t_m} \mid V_{m+2} \mid \dots \mid V_n \mid v_{m+1}[neg[]] \mid \\ v_{m+1}[pos[] \mid v'_{m+1}[] \mid v'_{m+2}[out\ v_{m+1}.R^{\varphi_{m+1}}]] \end{array}$$

$$\begin{array}{ll} \rightarrow & V_1^{t_1} \mid \ldots \mid V_m^{t_m} \mid V_{m+2} \mid \ldots \mid V_n \mid v_{m+1}[neg[]] \mid \\ & v_{m+1}[pos[] \mid v'_{m+1}[]] \mid v'_{m+2}[R^{\varphi_{m+1}}] \\ & \equiv P_{\mathcal{M},v_{m+1}\mapsto tt} \end{array}$$

and

$$\begin{array}{ll} P_{\mathcal{M}}^{neg} \rightarrow & V_1^{t_1} \mid \ldots \mid V_m^{t_m} \mid V_{m+2} \mid \ldots \mid V_n \mid v_{m+1}[pos[]] \mid \\ & v_{m+1}[neg[] \mid v'_{m+1}[] \mid v'_{m+2}[out\ v_{m+1}.R^{\varphi_{m+1}}]] \\ \rightarrow & V_1^{t_1} \mid \ldots \mid V_m^{t_m} \mid V_{m+2} \mid \ldots \mid V_n \mid v_{m+1}[pos[]] \mid \\ & v_{m+1}[neg[] \mid v'_{m+1}[]] \mid v'_{m+2}[R^{\varphi_{m+1}}] \\ & \equiv P_{\mathcal{M},v_{m+1}\mapsto ff} \end{array}$$

The proof goes in a similar way for the case where $m = n - 1$. ■

Lemma B.2 *For all m in $\{0, \ldots, n-1\}$, $\mathcal{M}$ being the interpretation $v_1 \mapsto t_1, \ldots, v_m \mapsto t_m$, we have*

- *for $0 \le m < n-1$, $P_{\mathcal{M},v_{m+1}\mapsto tt}$ and $P_{\mathcal{M},v_{m+1}\mapsto ff}$ are the two unique processes reachable from $P_{\mathcal{M}}$ that satisfy the ambient formula $v'_{m+2}[\mathbf{T}] \mid \mathbf{T}$.*
- *for $m = n-1$, $P_{\mathcal{M},v_{m+1}\mapsto tt}$ and $P_{\mathcal{M},v_{m+1}\mapsto ff}$ are the two unique processes reachable from $P_{\mathcal{M}}$ that satisfy the ambient formula $end[\mathbf{T}] \mid \mathbf{T}$.*

Proof For $0 \le m < n-1$, we know from the proof of Lemma B.1 that both $P_{\mathcal{M},v_{m+1}\mapsto tt}$ and $P_{\mathcal{M},v_{m+1}\mapsto ff}$ satisfy the ambient formula $v'_{m+2}[\mathbf{T}] \mid \mathbf{T}$ and do not satisfy formulas $v'[\mathbf{T}] \mid \mathbf{T}$ where v' is a primed ambient name different from v'_{m+2}. Now, still from the proof of Lemma B.1, we know that any reachable process from $P_{\mathcal{M}}$ is either $P_{\mathcal{M}'}$ for some extension $\mathcal{M}'$ of $\mathcal{M}$ or an "intermediate" process reachable from $P_{\mathcal{M}'}$ in one or two steps. It is easy to see that none of these "intermediate" processes satisfies an ambient formula $v'[\mathbf{T}] \mid \mathbf{T}$ whatever the primed name v' is. Finally, as $\mathcal{M}'$ is different from $\mathcal{M}$, $P_{\mathcal{M}'}$ will satisfy a formula $v'[\mathbf{T}] \mid \mathbf{T}$ for some $v' \ne v'_{m+2}$, but not the formula $v'_{m+2}[\mathbf{T}] \mid \mathbf{T}$.

The proof goes in a similar way for the case where $m = n - 1$. ■

Restatement of Lemma 4.3 *Assume φ is a closed quantified Boolean formula, and that $(P, \mathcal{A}) = enc(\varphi)$. Then $P \models \mathcal{A}$ if and only if φ is valid.*

Proof We are going to show for any $0 \le m \le n$ that for the interpretation $\mathcal{M}$ equal to $v_1 \mapsto t_1, \ldots, v_m \mapsto t_m$

$$\mathcal{M} \models \varphi_m \qquad \text{iff} \qquad P_{\mathcal{M}} \models \mathcal{A}^{\varphi_m}$$

Note that for $m = 0$, $\mathcal{M}$ is the empty interpretation, $\varphi_m = \varphi$, $P_{\mathcal{M}} = P$ and $\mathcal{A}^{\varphi_m} = \mathcal{A}$, so this statement obviously implies Lemma 4.3. The proof of this statement goes by induction on the number $l = n - m$ of quantifiers in φ_m.

For the base case $l = 0$: $\varphi_m = C_1 \wedge \ldots \wedge C_k$ is an unquantified formula and $\mathcal{M} = v_1 \mapsto t_1, \ldots v_n \mapsto t_n$. The interpretation $\mathcal{M}$ is a model for the formula φ_m if and only if $\mathcal{M}$ renders true at least one literal ℓ_i in each of the clauses C_i. Now, depending on whether ℓ_i occurs positively or negatively in C_i, we have two cases:

- $\ell_i = v_i$: by the encoding and the definition of $P_{\mathcal{M}}$, this is equivalent to that $[\![\ell_i]\!] = v_i[pos[\mathbf{0}] \mid v'_i[\mathbf{0}]] \mid \mathbf{T}$ and $P_{\mathcal{M}} = v_i[pos[\mathbf{0}] \mid v'_i[\mathbf{0}]] \mid P'$ for some ambient process P' which does not contain the ambient name v'_i. Therefore, it is equivalent to that $P_{\mathcal{M}} \models [\![\ell_i]\!]$.

- $\ell_i = \overline{v_i}$: this case is dual to the previous one.

Now, in both cases we have $P_{\mathcal{M}} \models [\![\ell_i]\!]$, which means that $P_{\mathcal{M}}$ is a model for at least one literal in each of the $[\![C_i]\!]$'s, and thus it is equivalent to that $P_{\mathcal{M}} \models \mathcal{A}^{\varphi_m}$.

For the induction step $1 < l \leq n$ (the particular base case where $l = 1$ differs only in the use of the ambient name *end* instead of v'_{n+1} and can be proved in the same way) we assume that the statement holds for $l-1$ (that is, it holds for $m+1$). The formula φ_m has the form $Q_{m+1}v_{m+1}\varphi_{m+1}$, so we have to consider two cases depending on whether Q_{m+1} is $\exists$ or $\forall$.

In the case of $\exists$, we have that $\mathcal{M} \models \varphi_m$ is equivalent to the disjunction $\mathcal{M}, v_{m+1} \mapsto tt \models \varphi_{m+1}$ or $\mathcal{M}, v_{m+1} \mapsto ff \models \varphi_{m+1}$. By induction hypothesis, this is equivalent to that either $P_{\mathcal{M},v_{m+1}\mapsto tt} \models \mathcal{A}^{\varphi_{m+1}}$ or $P_{\mathcal{M},v_{m+1}\mapsto ff} \models \mathcal{A}^{\varphi_{m+1}}$. By Lemma B.2, we know that $P_{\mathcal{M},v_{m+1}\mapsto tt}$ and $P_{\mathcal{M},v_{m+1}\mapsto ff}$ are the two unique processes reachable from $P_{\mathcal{M}}$ satisfying the ambient formula $v'_{m+2}[\mathbf{T}] \mid \mathbf{T}$. Therefore, the last statement is equivalent to that

$$P_{\mathcal{M}} \models \Diamond(v'_{m+2}[\mathbf{T}] \mid \mathbf{T}) \ \wedge \ \mathcal{A}^{\varphi_{m+1}}.$$

The case where $Q_{n-l+1} = \forall$ is dual to the previous one and leads to the equivalence with

$$P_{\mathcal{M}} \models \Box(v_{m+2}[\mathbf{T}] \mid \mathbf{T}) \ \Rightarrow \ \mathcal{A}^{\varphi_{m+1}}.$$

In both cases, by definition of *enc*, we have the equivalence with $P_{\mathcal{M}} \models \mathcal{A}^{\varphi_m}$. ■

B.3 Proof of Lemma 4.5

Lemma 4.5 is the crux of correctness for the encoding from Section 4.3 of QBF satisfaction in the calculus of immobile ambients with I/O. To prove it, let us first fix some notations and then prove some auxiliary lemmas.

We use notations similar to the previous section. For a given closed QBF formula $\varphi = Q_1v_1 \ldots Q_nv_n\psi$ in prenex and conjunctive normal form, we denote ψ by $C_1 \wedge \ldots \wedge C_k$. Let $\mathcal{M}$ be an interpretation $v_1 \mapsto t_1, \ldots, v_m \mapsto t_m$. We denote $\sigma_{\mathcal{M}}$ the substitution $\{v_1{\leftarrow}t_1, \overline{v_1}{\leftarrow}\overline{t_1}, \ldots, v_m{\leftarrow}t_m, \overline{v_m}{\leftarrow}\overline{t_m}\}$ where $\overline{t_i}$ is the negated value of t_i. If $\mathcal{M}$ is the empty interpretation, we let $\sigma_{\mathcal{M}}$ to be the identity.

For $0 \leq m \leq n$, let φ_m be the formula $Q_{m+1}v_{m+1} \ldots Q_nv_n\psi$ and $enc(\varphi_m) = (P^{\varphi_m}, \mathcal{A}^{\varphi_m})$. For $\mathcal{M} = v_1 \mapsto t_1, \ldots, v_m \mapsto t_m$, let us denote $P_{\mathcal{M}}$ the process $Q^{\varphi_m}\sigma_{\mathcal{M}}$ such that $P^{\varphi_m} \equiv v'_{m+1}[Q^{\varphi_m}]$. Note that in this notation $P^{\varphi_m}\sigma_{\mathcal{M}} = v'_{m+1}[P_{\mathcal{M}}]$. By $\mathcal{M}^+$ and $\mathcal{M}^-$ we denote respectively $\mathcal{M}, v_{m+1}{\leftarrow}tt, \overline{v_{m+1}}{\leftarrow}ff$ and $\mathcal{M}, v_{m+1}{\leftarrow}ff, \overline{v_{m+1}}{\leftarrow}tt$.

Lemma B.3 *For all $0 \leq m < n$,*

$$P_{\mathcal{M}} \rightarrow (\langle \mathit{ff} \rangle \mid v''_{m+1}[] \mid (\overline{v_{m+1}}).P^{\varphi_{m+1}})\sigma_{\mathcal{M},v_{m+1}\leftarrow tt}$$

and

$$P_{\mathcal{M}} \rightarrow (\langle tt \rangle \mid v''_{m+1}[] \mid (\overline{v_{m+1}}).P^{\varphi_{m+1}})\sigma_{\mathcal{M},v_{m+1}\leftarrow \mathit{ff}}$$

and there is no other P' such that $P \rightarrow P'$.

Proof *Straightforward from the encoding.* ■

Lemma B.4 *For all $0 \leq m < n$, $P_{\mathcal{M}} \rightarrow^2 (v''_{m+1}[] \mid P^{\varphi_{m+1}})\sigma_{\mathcal{M}^+}$ and $P_{\mathcal{M}} \rightarrow^2 (v''_{m+1}[] \mid P^{\varphi_{m+1}})\sigma_{\mathcal{M}^-}$ and there is no other P' such that $P \rightarrow^2 P'$.*

Proof *Straightforward from the encoding, Lemma B.3 and the definition of $P_{\mathcal{M}}$.* ■

Restatement of Lemma 4.5 *Assume φ is a closed quantified Boolean formula, and that $(P, \mathcal{A}) = enc(\varphi)$. Then $P \models \mathcal{A}$ if and only if φ is valid.*

Proof Let $V_0 = \mathbf{0}$ and for all $1 \leq m \leq n$ let $V_m = v''_m[]$. We are going to show for any $0 \leq m \leq n$ that for the interpretation $\mathcal{M}$ equal to $v_1 \mapsto t_1, \ldots, v_m \mapsto t_m$,

$$\mathcal{M} \models \varphi_m \ \text{ iff } \ V_m \mid P^{\varphi_m}\sigma_{\mathcal{M}} \models \mathcal{A}^{\varphi_m}.$$

The particular case of this statement with $m = 0$ is equivalent to Lemma 4.5. Its proof goes by induction over the number $l = n - m$ of quantified variables in φ_m.

Case where $l = 0$: the formula φ_m is equal to $C_1 \wedge \ldots \wedge C_k$, $\mathcal{M}$ has the form $v_1 \mapsto t_1, \ldots, v_n \mapsto t_n$ and $\mathcal{M} \models C_1 \wedge \ldots \wedge C_k$. As $C_1 \wedge \ldots \wedge C_k$ is in conjunctive normal form, for at least one literal ℓ_i in each C_i, $\mathcal{M}(\ell_i) = tt$. This is equivalent to that for each C_i, there exists at least one literal ℓ_i in C_i such that

- $v_j \leftarrow tt, \overline{v_j} \leftarrow \mathit{ff}$ belongs to $\sigma_{\mathcal{M}}$ if $\ell_i = v_j$ and
- $v_j \leftarrow \mathit{ff}, \overline{v_j} \leftarrow tt$ belongs to $\sigma_{\mathcal{M}}$ if $\ell_i = \overline{v_j}$.

By the definition of $enc(C_1 \wedge \ldots \wedge C_k)$, this is equivalent to that the interior of each C ambient (each marked by a D ambient) in the process $P^{\varphi_m}\sigma_{\mathcal{M}}$ contains a tt sub-ambient. This again is equivalent to $P^{\varphi_m}\sigma_{\mathcal{M}} \models \Box((D[\mathbf{0}] \mid \mathbf{T}) \Rightarrow (tt[\mathbf{0}] \mid \mathbf{T}))$ that is, to $P^{\varphi_m}\sigma_{\mathcal{M}} \models \mathcal{A}^{\varphi_m}$. Since V_m does not contain any subambient $D[\mathbf{0}]$, the statement follows.

Case where $l = 1$ (that is, $m = n - 1$): the formula φ_m is equal to $Q_n v_n \psi$, $\mathcal{M}$ is a the form $v_1 \mapsto t_1, \ldots, v_{n-1} \mapsto t_{n-1}$. We follow according to the value of Q_n:

- case where $Q_n = \exists$: $\mathcal{M} \models \varphi_m$ is equivalent to either $\mathcal{M}, v_n \leftarrow tt \models \psi$ or $\mathcal{M}, v_n \leftarrow \mathit{ff} \models \psi$. Using the case where $l = 0$, this is equivalent to that either $P^{\varphi_n}\sigma_{\mathcal{M}^+} \models \mathcal{A}^{\varphi_n}$ or $P^{\varphi_n}\sigma_{\mathcal{M}^-} \models \mathcal{A}^{\varphi_n}$.

By Lemma B.4, the processes $v''_n[] \mid P^{\varphi_n}\sigma_{\mathcal{M}^+}$ and $v''_n[] \mid P^{\varphi_n}\sigma_{\mathcal{M}^-}$ are the two unique ones reachable from $P_{\mathcal{M}}$ in two steps. Moreover, as P^{φ_n} can not be reduced, there is no process reachable from $P_{\mathcal{M}}$ in strictly more than two steps. It should be noticed that $P^{\varphi_n}\sigma_{\mathcal{M}^+}$ and $P^{\varphi_n}\sigma_{\mathcal{M}^-}$ both satisfy the formula $\mathit{Inst}(\mathit{end}) \wedge \neg\mathit{Inst}^+(\mathit{end})$ whereas by Lemma B.3 the two unique successors of $P_{\mathcal{M}}$ as well as $P_{\mathcal{M}}$ itself do not satisfy the formula $\mathit{Inst}(\mathit{end})$. Therefore, $P^{\varphi_n}\sigma_{\mathcal{M}^+} \models \mathcal{A}^{\varphi_n}$ or $P^{\varphi_n}\sigma_{\mathcal{M}^-} \models \mathcal{A}^{\varphi_n}$ holds iff $P_{\mathcal{M}} \models \Diamond((\mathit{Inst}(\mathit{end}) \wedge \neg\mathit{Inst}^+(\mathit{end})) \wedge \mathcal{A}^{\varphi_n})$. And thus, this is equivalent to $v''_{n-1}[] \mid v_n[P_{\mathcal{M}}] \models \mathbf{T} \mid v_n[\Diamond((\mathit{Inst}(\mathit{end}) \wedge \neg\mathit{Inst}^+(\mathit{end})) \wedge \mathcal{A}^{\varphi_n})]$, that is $v''_{n-1}[] \mid P^{\varphi_{n-1}} \models \mathcal{A}^{\varphi_{n-1}}$.

- case where $Q_n = \forall$: this case is dual to the previous one.

Case where $1 < l \leq n$: the formula φ_m is equal to $Q_{m+1}v_{m+1}\varphi_{m+1}$, $\mathcal{M}$ has the form $v_1 \mapsto t_1, \ldots, v_m \mapsto t_m$ and we assume that the statement holds for $l-1$ (that is, it holds for $m+1$). We follow according to the value of Q_{m+1}:

- case where $Q_{m+1} = \exists$: $\mathcal{M} \models \varphi_m$ is equivalent to either $\mathcal{M}, v_{m+1} \leftarrow \mathit{tt} \models \varphi_{m+1}$ or $\mathcal{M}, v_{m+1} \leftarrow \mathit{ff} \models \varphi_{m+1}$. By induction hypothesis, this is equivalent to that either $v''_{m+1}[] \mid P^{\varphi_{m+1}}\sigma_{\mathcal{M}^+} \models \mathcal{A}^{\varphi_{m+1}}$ or $v''_{m+1}[] \mid P^{\varphi_{m+1}}\sigma_{\mathcal{M}^-} \models \mathcal{A}^{\varphi_{m+1}}$.

  Let us have a look now at processes reachable from $P_{\mathcal{M}}$: of course, $P_{\mathcal{M}}$ itself is reachable, but by construction it does not satisfy the formula $\mathit{Inst}(v'_{m+2})$. By Lemma B.3, two processes are reachable in one step from $P_{\mathcal{M}}$, but they do not satisfy the formula $\mathit{Inst}(v'_{m+2})$. By Lemma B.4, two processes are reachable from $P_{\mathcal{M}}$ in two steps, namely $(v''_{m+1}[] \mid P^{\varphi_{m+1}})\sigma_{\mathcal{M}^+}$ and $(v''_{m+1}[] \mid P^{\varphi_{m+1}})\sigma_{\mathcal{M}^-}$ and they both satisfy the formulas $\mathit{Inst}(v'_{m+2})$ and $\neg\mathit{Inst}^+(v'_{m+2})$ (by construction). Now, by using once again Lemma B.3 for the internal of v'_{m+2} in $P^{\varphi_{m+1}}\sigma_{\mathcal{M}^+}$ and $P^{\varphi_{m+1}}\sigma_{\mathcal{M}^-}$, all the processes reachable from one of those latter satisfy $\mathit{Inst}^+(v'_{m+2})$.

  Therefore, the last statement is equivalent to that $P_{\mathcal{M}} \models \Diamond(\mathit{Inst}(v'_{m+2}) \wedge \neg\mathit{Inst}^+(v'_{m+2})) \wedge \mathcal{A}^{\varphi_{m+1}}$. Thus, it is equivalent to $V_m[] \mid v'_{m+1}[P_{\mathcal{M}}] \models \mathbf{T} \mid v'_{m+1}[\Diamond(\mathit{Inst}(v'_{m+2}) \wedge \neg\mathit{Inst}^+(v'_{m+2})) \wedge \mathcal{A}^{\varphi_{m+1}}]$, that is $V_m[] \mid P^{\varphi_m} \models \mathcal{A}^{\varphi_m}$.

- the case where $Q_{m+1} = \forall$ is dual to the previous one. ■