Proof Search Specifications for Bisimulation and Modal Logics for the π-calculus

Alwen Tiu
The Australian National University
and
Dale Miller
INRIA-Futurs & LIX, École polytechnique

We specify the operational semantics and bisimulation relations for the finite π-calculus within a logic that contains the ∇ quantifier for encoding *generic judgments* and definitions for encoding fixed points. Since we restrict to the finite case, the ability of the logic to unfold fixed points allows this logic to be complete for both the inductive nature of operational semantics and the coinductive nature of bisimulation. The ∇ quantifier helps with the delicate issues surrounding the scope of variables within π-calculus expressions and their executions (proofs). We shall illustrate several merits of the logical specifications permitted by this logic: they are natural and declarative; they contain no side-conditions concerning names of variables while maintaining a completely formal treatment of such variables; differences between late and open bisimulation relations arises from familar logic distinctions; the interplay between the three quantifiers ($\forall$, $\exists$, and ∇) and their scopes can explain the differences between early and late bisimulation and between various modal operators based on bound input and output actions; and proof search involving the application of inference rules, unification, and backtracking can provide complete proof systems for one-step transitions, bisimulation, and satisfaction in modal logic.

Categories and Subject Descriptors: F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Specification Techniques*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Proof Theory*

General Terms: Theory, Verification

Additional Key Words and Phrases: proof search, λ-tree syntax, ∇ quantifier, generic judgments, higher-order abstract syntax, π-calculus, bisimulation, modal logics

1. INTRODUCTION

We present a formal specification of various aspects of the π-calculus, including its syntax, operational semantics, bisimulation relations, and modal logic. We shall do this by using the $FO\lambda^{\Delta\nabla}$ logic. We provide a high-level introduction to this logic here before proceeding to provide the technical aspects of it in the next section.

Authors' addresses: A. Tiu, Research School of Information Sciences and Engineering, Building 115, The Australian National University, Canberra, ACT 0200, Australia; D. Miller, Laboratoire d'Informatique (LIX), École Polytechnique, Rue de Saclay, 91128 Palaiseau Cedex, France.

Just as it is common to use meta-level application to represent object-level application (for example, the encoding of $P + Q$ is via the meta-level application of the encoding for plus to the encoding of its two arguments), we shall use meta-level λ-abstractions to encode object-level abstractions. The term *higher-order abstract syntax* (HOAS) [Pfenning and Elliott 1988] is commonly used to describe this approach to mapping object-level abstractions into some meta-level abstraction. Of course, the nature of the resulting encodings varies as one varies the meta-level. For example, if the meta-level is a higher-order functional programming language or a higher-order type theory, the usual abstraction available constructs function spaces. In this case, HOAS maps object-level abstractions to semantically rich function spaces: the question as to whether or not two syntactic objects are equal is mapped to the question of determining if two functions are equal (typically, an undecidable judgment). In such a settings, HOAS is less about syntax and more about a particular, mathematical denotation of the syntax. In this paper, we start with a subset of the Simple Theory of Types [Church 1940] that does not contain the *mathematical* axioms of extensionality, description, choice, and infinite (more precisely, intuitionistic versions of axioms 1 to 6 of [Church 1940]). In this setting, the meta-level abstraction is not strong enough to denote general computable functions and equality of terms is decidable. As a result, this weaker logic provides term-level bindings that can be used as a valuable meta-level abstraction for syntax containing bindings. This style of describing syntax via a meta-logic with a weak form of λ-abstraction has been called the λ-*tree* [Miller 2000] approach to HOAS to distinguish it from the other approach based on more general function spaces. The λ-tree approach to encoding syntax is an old one, dating back to [Huet and Lang 1978; Miller and Nadathur 1986; 1987] and is widely used in specifications written in the logic programming languages λProlog [Nadathur and Miller 1988] and Twelf [Pfenning and Schürmann 1999].

Following Church, we shall then use λ-abstractions to encode both *term-level abstractions* and *formula-level abstractions* (*e.g.*, quantifiers). The computational aspects of the π-calculus are usually specified via structured operational semantics [Plotkin 1981]: here, such specifications are encoded directly as inference rules and proofs over primitive relational judgments (*e.g.*, one-step transitions). As a result, a formal account for the interaction of binding in syntax and binding in computation leads to notions of *proof-level abstractions*. One such binding is the familiar *eigenvariable* abstraction of Gentzen [Gentzen 1969] to denote a universally quantified variable that has scope over an entire sequent. A second proof-level binding was introduced in [Miller and Tiu 2005] to capture a notion of *generic judgment*: this proof-level binding has a scope over individual entries within a sequent and is closely associated with the formula-level binding introduced by the ∇-quantifier. A major goal of this paper is to illustrate how well the ∇-quantifier and this second proof-level abstraction can be used to specify and reason about computation: the π-calculus has been chosen, in part, because it is a small calculus in which bindings play a dynamic role in computation.

A reading of the truth condition for $\nabla x_\gamma.Bx$ is something like: given the new element x of type γ, check the truth of Bx. Notice that this reading of the quantifier does not require knowing whether or not the type γ actually contains any

members. Being independent of the domain of quantification is, of course, central to the notion of *generic*: something holds generically usually means that it holds for certain "internal" reasons (the structure of an argument, for example) and not for some accident concerning members of the domain. This is quite different from determining the truth of $\forall x_\gamma.Bx$: check that Bt is true for all t in the type γ. If the type is empty, this condition is vacuously true and if the type is infinite, we have an infinite number of checks to make in principle.

It is useful to provide here a high-level comparison between the ∇-quantifier and the "new" quantifier of Gabbay and Pitts [Gabbay and Pitts 2001]. In their set theory foundations, a domain containing an infinite number of names is assumed. To deal with notions of freshness, renaming of bound variables, substitution, etc, they provide a series of primitives between names and terms that can be used to guarantee that a name does not occur within a term, that one name can be swapped for another, etc. Based on these concepts, they can define a new quantifier that guarantees the selection of a "fresh" name for some specific context. In our approach here, there is no particular class of names: the ∇ quantifier will work at any type. (Later, when we discuss the π-calculus explicitly, we shall assume a type for names since this is required by this particular application.) Also, here types do not need to be infinite or even non-empty. Thus, the Gabbay-Pitts approach assumes that the type of names is fixed and closed, while the type used with ∇ is open, in the sense that new members of that type can be constructed by the logic for use within a ∇-bound scope. More specifically, the set of theorems for these two quantifiers is quite different. For example, in the logic considered here, the formulas $\forall x.Bx \supset \nabla x.Bx$ and $\nabla x.Bx \supset \exists x.Bx$ are not theorems, but if ∇ is replaced with the Gabbay-Pitts quantifier, they do hold in their theory.

This distinction between having an open versus closed datatype is also a theme that highlights the differences between intuitionistic and classical logic. The logic in this paper is based on intuitionistic logic, a weaker logic than classical logic. One of the principles missing from intuitionistic logic is that of the excluded middle: that is, $A \vee \neg A$ is not generally provable in intuitionistic logic. Consider, for example, the following formula concerning the variable w:

$$\forall x_\gamma[x = w \vee x \neq w]. \qquad (*)$$

In classical logic, this formula is a trivial theorem. From a constructive point-of-view, this formula is not trivial and might not be desirable in all cases. If the type of quantification γ is a conventional (closed) datatype, then we might expect to have a decision procedure for equality. For example, if γ is the type for lists, then it is a simple matter to construct a procedure that decides whether or not two members of γ are equal by considering the top constructor of the list and, in the event of comparing two non-empty lists, making a recursive call (assuming a decision procedure is available for the elements of the list). In fact, it is possible to prove in an intuitionistic logic augmented with induction (see, for example, [Tiu 2004]) the formula $(*)$ for closed, first-order datatypes.

If the type γ is not given inductively, as is the usual case for names in intuitionistic formalizations of the π-calculus (see [Fiore et al. 1999; Hofmann 1999; Despeyroux 2000] and below), then the corresponding instance of $(*)$ is not provable. Thus, whether or not we allow instances of $(*)$ to be assumed can change the nature of

a specification. In fact, we show in Section 5, that if we add to our specification of *open bisimulation* [Sangiorgi 1996] assumptions corresponding to $(*)$, then we get a specification of *late bisimulation*. If we were working with a classical logic, such a declarative presentation of these two bisimulations would not be so easy to describe. In a similar fashion, Mitchell and Moggi [Mitchell and Moggi 1991] used an intuitionistic logic model allowing for open types to help establish completeness theorems for the simply typed λ-calculus.

The authors first presented the logic used in this paper in [Miller and Tiu 2003] and illustrated its usefulness with the π-calculus: in particular, the specifications of one-step transitions in Figure 2 and of late bisimulation in Figure 3 also appear in [Miller and Tiu 2003] but without proof. In this paper, we state the formal properties of our specifications, provide a specification of late bisimulation and provide a novel comparison between open and late bisimulation. In particular, we show that the difference between open and late bisimulation (apart from the difference that arises from the use of closed and open types discussed above) can be captured by the different quantification of free names using $\forall$ and ∇. The different treatment of free names, that is, whether some free names are instantiable or not, highlights the difference between late and open bisimulation, as noted in [Sangiorgi and Walker 2001], where the notion of *distinction* among names is introduced to define the open bisimulation relation. We show in Section 5 that a natural class of distinctions can be captured by the alternation of $\forall$ and ∇ quantifiers, and in the case where we are interested only in checking open bisimilarity modulo empty distinction, the notion of distinction that arises in the process of checking bisimilarity is completely subsumed by quantifier alternation. In Section 6 we show that "modal logics for mobility" can easily be handled as well and present, for the first time, a modal characterization of open bisimulation. Since our focus in this paper is on names, scoping of names, dependency of names, and distinction of names, we have chosen to focus on finite π-calculus. The treatment of π-calculus with replication is presented in Section 7 through an example. In Section 8 we outline the automation of proof search based on these specifications: when such automation is applied to our specification of open bisimulation, a symbolic bisimulation procedures arises. In Section 9 we present some related and future work and in Section 10 we conclude with a brief description of related and future research. In order to improve the readability of the main part of the paper, numerous technical proofs have been moved to the appendicies.

Parts of this paper, in their preliminary forms and without proofs, have been presented in [Tiu and Miller 2004; Tiu 2005]. This paper contains revisions and extensions of these previous works. In particular, the materials on bisimulation encodings (Section 5) corresponds to [Tiu and Miller 2004] and those that concern the encodings of modal logics for the π-calculus (Section 6) corresponds to [Tiu 2005].

2. OVERVIEW OF THE LOGIC

This paper is about the use of a certain logic to specify and reason about computation. We shall not assume that the reader is interested in an in-depth analysis of the logic but with its applications. Of course, a number of properties of the

logic are required: in general, we state the most relevant results we shall need in reasoning about our π-calculus specifications. The reader who is interested in more details about this logic is referred to [Tiu 2004] and [Miller and Tiu 2005].

In this section we describe the logic $FO\lambda^{\Delta\nabla}$ (pronounced "fold-nabla"). The etymology of this term is roughly the following. The logic $FO\lambda$ (a first-order logic for λ-terms) is the result of extending Gentzen's LJ sequent calculus [Gentzen 1969] for first-order logic to allow terms to be simply typed λ-terms and quantifiers to range over all non-predicate types. This basic logic supports the *λ-tree syntax* [Miller 2000] approach to *higher-order abstract syntax* [Pfenning and Elliott 1988]. In this logic, λ-abstractions can be used to directly represent abstractions within syntax (as illustrated by Church in [Church 1940]) but without the additional mathematical presuppositions that λ-terms should also denote functions. As a result, syntax containing bindings can be directly encoded using λ-abstractions in $FO\lambda$. The logic $FO\lambda^{\Delta}$ extends $FO\lambda$ with a notion of "fixed points" via the technical device of "definitions," presented below and marked with the symbol $\stackrel{\triangle}{=}$. This logic now allows for capturing important forms of "negation-as-failure" and of "must behavior" in encodings of operational semantics [McDowell and Miller 2000; McDowell et al. 2003]. The addition of this kind of negation also allows one to observe that the usual treatment of "generic judgments" in $FO\lambda$ based on the universal quantifier is inadequate and that a more intensional treatment of such judgments are needed. Such a treatment is given by the addition of the ∇-quantifier [Miller and Tiu 2005]. The logic $FO\lambda^{\Delta\nabla}$ is the result of adding this quantifier.

A *sequent* is an expression of the form $B_1, \ldots, B_n \vdash B_0$ where $B_0, \ldots, B_n$ are formulas and the elongated turnstile $\vdash$ is the sequent arrow. To the left of the turnstile is a multiset: thus repeated occurrences of a formula are allowed. If the formulas $B_0, \ldots, B_n$ contain free variables, they are considered universally quantified outside the sequent, in the sense that if the above sequent is provable then every instance of it is also provable. In proof theoretical terms, such free variables are called *eigenvariables*.

A first attempt at using sequent calculus to capture judgments about the π-calculus could be to use eigenvariables to encode names in the π-calculus, but this is certainly problematic. For example, if we have a proof for the sequent $\vdash Pxy$, where x and y are different eigenvariables, then logic dictates that the sequent $\vdash Pzz$ is also provable (given the universal quantifier reading of eigenvariables). If the judgment P is about, say, bisimulation, then it is not likely that a statement about bisimulation involving two different names x and y remains true if they are identified to the same name z.

To address this problem, the logic $FO\lambda^{\Delta\nabla}$ extends sequents with a new notion of "local scope" for proof-level bound variables (originally motivated in [Miller and Tiu 2003] to encode "generic judgments"). In particular, sequents in $FO\lambda^{\Delta\nabla}$ are of the form

$$\Sigma\,;\, \sigma_1 \triangleright B_1, \ldots, \sigma_n \triangleright B_n \vdash \sigma_0 \triangleright B_0$$

where Σ is a *global signature*, i.e., the set of eigenvariables whose scope is over the whole sequent, and σ_i is a *local signature*, i.e., a list of variables scoped over B_i. We shall consider sequents to be binding structures in the sense that the signatures, both the global and local ones, are abstractions over their respective scopes.

$$\frac{}{\Sigma\,;\,\sigma \triangleright B, \Gamma \vdash \sigma \triangleright B}\ init \qquad \frac{\Sigma\,;\,\Delta \vdash \mathcal{B} \qquad \Sigma\,;\,\mathcal{B}, \Gamma \vdash \mathcal{C}}{\Sigma\,;\,\Delta, \Gamma \vdash \mathcal{C}}\ cut$$

$$\frac{\Sigma\,;\,\sigma \triangleright B, \sigma \triangleright C, \Gamma \vdash \mathcal{D}}{\Sigma\,;\,\sigma \triangleright B \wedge C, \Gamma \vdash \mathcal{D}}\ \wedge\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \vdash \sigma \triangleright B \qquad \Sigma\,;\,\Gamma \vdash \sigma \triangleright C}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright B \wedge C}\ \wedge\mathcal{R}$$

$$\frac{\Sigma\,;\,\sigma \triangleright B, \Gamma \vdash \mathcal{D} \qquad \Sigma\,;\,\sigma \triangleright C, \Gamma \vdash \mathcal{D}}{\Sigma\,;\,\sigma \triangleright B \vee C, \Gamma \vdash \mathcal{D}}\ \vee\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \vdash \sigma \triangleright B}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright B \vee C}\ \vee\mathcal{R}$$

$$\frac{}{\Sigma\,;\,\sigma \triangleright \bot, \Gamma \vdash \mathcal{B}}\ \bot\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \vdash \sigma \triangleright C}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright B \vee C}\ \vee\mathcal{R}$$

$$\frac{\Sigma\,;\,\Gamma \vdash \sigma \triangleright B \qquad \Sigma\,;\,\sigma \triangleright C, \Gamma \vdash \mathcal{D}}{\Sigma\,;\,\sigma \triangleright B \supset C, \Gamma \vdash \mathcal{D}}\ \supset\mathcal{L} \qquad \frac{\Sigma\,;\,\sigma \triangleright B, \Gamma \vdash \sigma \triangleright C}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright B \supset C}\ \supset\mathcal{R}$$

$$\frac{\Sigma, \sigma \vdash t : \gamma \qquad \Sigma\,;\,\sigma \triangleright B[t/x], \Gamma \vdash \mathcal{C}}{\Sigma\,;\,\sigma \triangleright \forall_\gamma x.B, \Gamma \vdash \mathcal{C}}\ \forall\mathcal{L} \qquad \frac{\Sigma, h\,;\,\Gamma \vdash \sigma \triangleright B[(h\ \sigma)/x]}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright \forall x.B}\ \forall\mathcal{R}$$

$$\frac{\Sigma, h\,;\,\sigma \triangleright B[(h\ \sigma)/x], \Gamma \vdash \mathcal{C}}{\Sigma\,;\,\sigma \triangleright \exists x.B, \Gamma \vdash \mathcal{C}}\ \exists\mathcal{L} \qquad \frac{\Sigma, \sigma \vdash t : \gamma \qquad \Sigma\,;\,\Gamma \vdash \sigma \triangleright B[t/x]}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright \exists_\gamma x.B}\ \exists\mathcal{R}$$

$$\frac{\Sigma\,;\,(\sigma, y) \triangleright B[y/x], \Gamma \vdash \mathcal{C}}{\Sigma\,;\,\sigma \triangleright \nabla x\ B, \Gamma \vdash \mathcal{C}}\ \nabla\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \vdash (\sigma, y) \triangleright B[y/x]}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright \nabla x\ B}\ \nabla\mathcal{R}$$

$$\frac{\Sigma\,;\,\mathcal{B}, \mathcal{B}, \Gamma \vdash \mathcal{C}}{\Sigma\,;\,\mathcal{B}, \Gamma \vdash \mathcal{C}}\ c\mathcal{L} \qquad \frac{\Sigma\,;\,\Gamma \vdash \mathcal{C}}{\Sigma\,;\,\mathcal{B}, \Gamma \vdash \mathcal{C}}\ w\mathcal{L} \qquad \frac{}{\Sigma\,;\,\Gamma \vdash \sigma \triangleright \top}\ \top\mathcal{R}$$

Fig. 1. The core rules of $FO\lambda^{\Delta\nabla}$.

The variables in Σ and σ_i will admit α-conversion by systematically changing the names of variables in signatures as well as those in their scope, following the usual convention of the λ-calculus. The meaning of eigenvariables is as before except that now instantiation of eigenvariables has to be capture-avoiding with respect to the local signatures. The variables in local signatures act as locally scoped *generic constants*: that is, they do not vary in proofs since they will not be instantiated. The expression $\sigma \triangleright B$ is called a *generic judgment* or simply a *judgment*. We use script letters $\mathcal{A}$, $\mathcal{B}$, *etc* to denote judgments. We write simply B instead of $\sigma \triangleright B$ if the signature σ is empty. We shall often write the list σ as a string of variables: *e.g.*, a judgment $(x_1, x_2, x_3) \triangleright B$ will be written as $x_1x_2x_3 \triangleright B$. If the list x_1, x_2, x_3 is known from context we shall also abbreviate the judgment as $\bar{x} \triangleright B$.

Following Church [1940], the type o is used to denote the type of formulas. The propositional constants of $FO\lambda^{\Delta\nabla}$ are $\wedge$ (conjunction), $\vee$ (disjunction), $\supset$ (implication), $\top$ (true) and $\bot$ (false). Syntactically, logical constants can be see as typed constants: for example, the binary connective have type $o \to o \to o$. For each simple type γ that does not contain o, there are three quantifiers in $FO\lambda^{\Delta\nabla}$: namely, $\forall_\gamma$ (universal quantifier), $\exists_\gamma$ (existential quantifier), ∇_γ (nabla), each one of type $(\gamma \to o) \to o$. The subscript type γ is often dropped when it can be inferred from context or its value is not important. Since we do not allow quantification over predicates, this logic is proof-theoretically similar to first-order logic (hence, the letters FO in $FO\lambda^{\Delta\nabla}$). The core inference rules for $FO\lambda^{\Delta\nabla}$ are given in Figure 1.

During the search for proofs (reading rules bottom up), inference rules for $\forall$ and

$\exists$ quantifier place new variables (eigenvariables) into the global signature while the inference rules for ∇ place new variables into a local signature. In the $\forall\mathcal{R}$ and $\exists\mathcal{L}$ rules, raising [Miller 1992] is used when moving the bound variable x (which can be substituted for by terms containing variables in both the global signature and the local signature σ) with the variable h (which can only be instantiated with terms containing variables in the global signature). In order not to miss substitution terms, the variable x is replaced by the term $(h\, x_1 \ldots x_n)$: the latter expression is written simply as $(h\, \sigma)$ where σ is the list $x_1, \ldots, x_n$. As is usual, the eigenvariable h must not be free in the lower sequent of these rules. In $\forall\mathcal{L}$ and $\exists\mathcal{R}$, the term t can have free variables from both Σ and σ, a fact that is given by the typing judgment $\Sigma, \sigma \vdash t : \tau$. The $\nabla\mathcal{L}$ and $\nabla\mathcal{R}$ rules have the proviso that y is not free in $\nabla x\ B$.

While sequent calculus introduction rules generally only introduce logical connectives, the full logic $FO\lambda^{\Delta\nabla}$ additionally allows introduction of atomic judgments; that is, judgments which do not contain any occurrences of logical constants. To each atomic judgment, $\mathcal{A}$, we associate a defining judgment, $\mathcal{B}$, the *definition* of $\mathcal{A}$. The introduction rule for the judgment $\mathcal{A}$ is in effect done by replacing $\mathcal{A}$ with $\mathcal{B}$ during proof search. This notion of definitions is an extension of work by Schroeder-Heister [Schroeder-Heister 1993], Eriksson [Eriksson 1991], Girard [Girard 1992], Stärk [Stärk 1994], and McDowell and Miller [McDowell and Miller 2000]. These inference rules for definitions allow for modest reasoning about the fixed points of definitions.

DEFINITION 1. A *definition clause* is written $\forall\bar{x}[p\,\bar{t} \triangleq B]$, where p is a predicate constant, every free variable of the formula B is also free in at least one term in the list $\bar{t}$ of terms, and all variables free in $p\,\bar{t}$ are contained in the list $\bar{x}$ of variables. The atomic formula $p\,\bar{t}$ is called the *head* of the clause, and the formula B is called the *body*. The symbol $\triangleq$ is used simply to indicate a definitional clause: it is not a logical connective. The predicate p occurs strictly positively in B: that is, it does not occur to the left of any $\supset$ (implication).

Let $\forall_{\tau_1} x_1 \ldots \forall_{\tau_n} x_n.H \triangleq B$ be a definition clause. Let $y_1, \ldots, y_m$ be a list of variables of types $\alpha_1, \ldots, \alpha_m$, respectively. The *raised definition clause* of H with respect to the signature $\{y_1 : \alpha_1, \ldots, y_m : \alpha_m\}$ is defined as

$$\forall h_1 \ldots \forall h_n.\bar{y} \triangleright H\theta \triangleq \bar{y} \triangleright B\theta$$

where θ is the substitution $[(h_1\, \bar{y})/x_1, \ldots, (h_n\, \bar{y})/x_n]$ and h_i is of type $\alpha_1 \to \ldots \to \alpha_m \to \tau_i$, for every $i \in \{1, \ldots, n\}$. A *definition* is a set of definition clauses together with their raised clauses.

To guarantee the consistency (and cut-elimination) of the logic $FO\lambda^{\Delta\nabla}$, we need some kind of stratification of definition that limits the definition of one predicate to depend negatively on another predicate (see [Miller and Tiu 2003] for the full details). All definitions considered in this paper are stratified appropriately and cut-elimination holds for the logic using them.

The introduction rules for a defined judgment are as follow. When applying the introduction rules, we shall omit the outer quantifiers in a definition clause and assume implicitly that the free variables in the definition clause are distinct from

other variables in the sequent.

$$\frac{\{\Sigma\theta\,;\, \mathcal{B}\theta, \Gamma\theta \vdash \mathcal{C}\theta \mid \theta \in CSU(\mathcal{A}, \mathcal{H}) \text{ for some clause } \mathcal{H} \stackrel{\triangle}{=} \mathcal{B}\}}{\Sigma\,;\, \mathcal{A}, \Gamma \vdash \mathcal{C}}\ def\mathcal{L}$$

$$\frac{\Sigma\,;\, \Gamma \vdash \mathcal{B}\theta}{\Sigma\,;\, \Gamma \vdash \mathcal{A}}\ def R, \quad \text{where } \mathcal{H} \stackrel{\triangle}{=} \mathcal{B} \text{ is a definition clause and } \mathcal{H}\theta = \mathcal{A}$$

In the above rules, we apply substitution to judgments. The result of applying a substitution θ to a generic judgment $x_1, \ldots, x_n \triangleright B$, written as $(x_1, \ldots, x_n \triangleright B)\theta$, is $y_1, \ldots, y_n \triangleright B'$, if $(\lambda x_1 \ldots \lambda x_n.B)\theta$ is equal (modulo λ-conversion) to $\lambda y_1 \ldots \lambda y_n.B'$. If Γ is a multiset of generic judgments, then $\Gamma\theta$ is the multiset $\{J\theta \mid J \in \Gamma\}$. In the *def*$\mathcal{L}$ rule, we use the notion of *complete set of unifiers* (CSU) [Huet 1975]. We denote by $CSU(\mathcal{A}, \mathcal{H})$ the complete set of unifiers for the pair $(\mathcal{A}, \mathcal{H})$: that is, for any substitution θ such that $\mathcal{A}\theta = \mathcal{H}\theta$, there is a substitution $\rho \in CSU(\mathcal{A}, \mathcal{H})$ such that $\theta = \rho \circ \theta'$ for some substitution θ'. In all the applications of *def*$\mathcal{L}$ in this paper, the set $CSU(\mathcal{A}, \mathcal{H})$ is either empty (the two judgments are not unifiable) or contains a single substitution denoting the most general unifier. The signature $\Sigma\theta$ in *def*$\mathcal{L}$ denotes a signature obtained from Σ by removing the variables in the domain of θ and adding the variables in the range of θ. In the *def*$\mathcal{L}$ rule, reading the rule bottom-up, eigenvariables can be instantiated in the premise, while in the *def*R rule, eigenvariables are not instantiated. The set that is the premise of the *def*$\mathcal{L}$ rule means that that rule instance has a premise for every member of that set: if that set is empty, then the premise is proved.

One might find the following analogy with logic programming helpful: if a definition is viewed as a logic program, then the *def*R rule captures backchaining and the *def*$\mathcal{L}$ rule corresponds to *case analysis* on all possible ways an atomic judgment could be proved. In the case where the program has only finitely many computation paths, we can effectively encode *negation-as-failure* using *def*$\mathcal{L}$ [Hallnäs and Schroeder-Heister 1991].

3. SOME META-THEORY OF THE LOGIC

Once we have written a computational specification as logical formulas, it is important that the underlying logic has a number of formal properties that will allow us to reason about that specification.

Cut-elimination for $FO\lambda^{\Delta\nabla}$ [Miller and Tiu 2005; Tiu 2004] is probably the single most important meta-theoretic property needed. Beside guaranteeing the consistency of the logic, it also provides completeness for *cut-free* proofs: these proofs are used to help prove the *adequacy* of a logical specification. For example, the fact that a certain specification actually encodes the one-step transition relation or the bisimulation relation starts by examining the highly restricted structure of cut-free proofs. Also, cut-elimination allows use to modus ponens and substitutions into cut-free proofs and to be assured that another cut-free proof arises from that operation.

Another important structural property of provability is the *invertibility* of inference rules. An inference rule of logic is *invertible* if the provability of the conclusion implies the provability of the premise(s) of the rule. The following rules in $FO\lambda^{\Delta\nabla}$ are invertible: $\wedge\mathcal{R}, \wedge\mathcal{L}, \vee\mathcal{L}, \supset\mathcal{R}, \forall\mathcal{R}, \exists\mathcal{L}$, *def*$\mathcal{L}$ (see [Tiu 2004] for a proof). Knowing

the invertibility of a rule can be useful in determining some structure of a proof. For example, if we know that a sequent $A \vee B, \Gamma \vdash C$ is provable, then by the invertibility of $\vee\mathcal{L}$, we know that it must be the case that $A, \Gamma \vdash C$ and $B, \Gamma \vdash C$ are provable.

We now present several meta-theoretic properties of provability that are specifically targeted at the ∇-quantifier. The first proposition below states a weakening result for local signatures and the second states that the global scope of an eigenvariable can be weakened to be a locally scoped variable. (See [Tiu 2004, Chapter 3.3] for more proofs.)

PROPOSITION 2. *If $\vdash B$ and x is not free in B, then $\vdash \nabla x B$.*

PROPOSITION 3. *If $\vdash \forall x B$ then $\vdash \nabla x B$.*

One might expect that the implication $\forall_\tau x B \supset \nabla_\tau x B$ holds as well. Notice that if τ is empty this statement would not be expected to be true and, hence, we do not accept it in the core logic.

The converse of Proposition 3 is not true in general. However, it is true for a restricted class of formulas and definitions, called $\mathsf{hc}^{\forall\nabla}$-*formulas* (for Horn clauses with $\forall$ and ∇) and $\mathsf{hc}^{\forall\nabla}$-*definitions*, respectively. A $\mathsf{hc}^{\forall\nabla}$-formula is a formula which does not contain any occurrence of the logical constant $\supset$ (implication). A $\mathsf{hc}^{\forall\nabla}$-definition is a definition whose bodies are $\mathsf{hc}^{\forall\nabla}$-formulas. For example, the definition of the one-step transition in Figure 2 is an $\mathsf{hc}^{\forall\nabla}$-definition.

PROPOSITION 4. *Let $\mathcal{D}$ be a $\mathsf{hc}^{\forall\nabla}$-definition and $\forall x G$ be a $\mathsf{hc}^{\forall\nabla}$-formula. Then $\vdash \forall x G$ if and only if $\vdash \nabla x G$.*

Finally, provability of a formula or a sequent is not affected by the exchange in the order of ∇-quantification or by substitutions.

PROPOSITION 5. *The formula $\nabla x \nabla y\ B$ is provable if and only if $\nabla y \nabla x\ B$ is provable.*

PROPOSITION 6. *Let Π be a proof of $\Sigma\,;\Gamma \vdash \mathcal{C}$. Then for any substitution, there exists a proof Π' of $\Sigma\theta\,;\Gamma\theta \vdash \mathcal{C}\theta$ such that the height of proof of Π' is less or equal to the height of proof of Π.*

4. LOGICAL SPECIFICATION OF ONE-STEP TRANSITION

In this paper, we shall mainly focus on the finite π-calculus; that is, the fragment of the π-calculus without recursion (or replication). In particular, process expressions are defined as usual as

$$\mathtt{P} ::= 0 \mid \bar{x}y.\mathtt{P} \mid x(y).\mathtt{P} \mid \tau.\mathtt{P} \mid (x)\mathtt{P} \mid [x = y]\mathtt{P} \mid \mathtt{P}|\mathtt{P} \mid \mathtt{P} + \mathtt{P}.$$

We use the symbols $\mathtt{P}$, $\mathtt{Q}$, $\mathtt{R}$, $\mathtt{S}$, and $\mathtt{T}$ to denote processes and lower case letters, *e.g.*, a, b, c, d, x, y, z to denote names. The occurrence of y in the process $x(y).\mathtt{P}$ and $(y)\mathtt{P}$ is a binding occurrence, with $\mathtt{P}$ as its scope. The set of free names in $\mathtt{P}$ is denoted by $\mathrm{fn}(\mathtt{P})$, the set of bound names is denoted by $\mathrm{bn}(\mathtt{P})$. We write $\mathrm{n}(\mathtt{P})$ for the set $\mathrm{fn}(\mathtt{P}) \cup \mathrm{bn}(\mathtt{P})$. We consider processes to be syntactical equivalent up to renaming of bound names.

The relation of one-step (late) transition [Milner et al. 1992] for the π-calculus is denoted by $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$, where $\mathtt{P}$ and $\mathtt{Q}$ are processes and α is an action. The kinds of actions are *the silent action* τ, *the free input action* xy, *the free output action* $\bar{x}y$, *the bound input action* $x(y)$, and *the bound output action* $\bar{x}(y)$. The name y in $x(y)$ and $\bar{x}(y)$ is a binding occurrence. Just like we did with processes, we use $\mathrm{fn}(\alpha)$, $\mathrm{bn}(\alpha)$ and $\mathrm{n}(\alpha)$ to denote free names, bound names, and names in α. An action without binding occurrences of names is a *free action*; otherwise it is a *bound action*.

Three primitive syntactic categories are used to encode the π-calculus into λ-tree syntax: n for names, p for processes, and a for actions. We do not assume any inhabitants of type n: as a consequence, a free name is translated to a variable of type n that is either universally or ∇-quantified, depending on whether we want to treat that name as instantiable or not. For instance, when encoding late bisimulation, free names correspond to ∇-quantified variables, while when encoding open bisimulation, free names correspond to universally quantified variables (Section 5). Since the rest of this paper is about the π-calculus, the ∇ quantifier will from now on only be used at type n.

There are three constructors for actions: $\tau : a$ (for the silent action) and the two constants $\downarrow$ and $\uparrow$, both of type $n \to n \to a$ (for building input and output actions, respectively). The free output action $\bar{x}y$, is encoded as $\uparrow xy$ while the bound output action $\bar{x}(y)$ is encoded as $\lambda y\ (\uparrow xy)$ (or the η-equivalent term $\uparrow x$). The free input action xy, is encoded as $\downarrow xy$ while the bound input action $x(y)$ is encoded as $\lambda y\ (\downarrow xy)$ (or simply $\downarrow x$). Notice that bound input and bound output actions have type $n \to a$ instead of a.

The following are process constructors, where $+$ and $|$ are written as infix:

$$\begin{array}{cccc} 0 : p & \tau : p \to p & \mathit{out} : n \to n \to p \to p & \mathit{in} : n \to (n \to p) \to p \\ + : p \to p \to p & | : p \to p \to p & \mathit{match} : n \to n \to p \to p & \nu : (n \to p) \to p \end{array}$$

Notice τ is overloaded by being used as a constructor of actions and of processes. The one-step transition relation is represented using two predicates: The predicate $\cdot \xrightarrow{\cdot} \cdot$ of type $p \to a \to p \to o$ encodes transitions involving the silent and free actions while the predicate $\cdot \xrightarrow{\cdot} \cdot$ of type $p \to (n \to a) \to (n \to p) \to o$ encodes transitions involving bound values. The precise translation of π-calculus syntax into simply typed λ-terms is given in the following definition.

Definition 7. The following function $[\![.]\!]$ translates process expressions to $\beta\eta$-long normal terms of type p.

$$\begin{array}{ccccc} [\![0]\!] = 0 & [\![\mathtt{P} + \mathtt{Q}]\!] = [\![\mathtt{P}]\!] + [\![\mathtt{Q}]\!] & [\![\mathtt{P}|\mathtt{Q}]\!] = [\![\mathtt{P}]\!] \mid [\![\mathtt{Q}]\!] & [\![\tau.\mathtt{P}]\!] = \tau\ [\![\mathtt{P}]\!] \\ [\![[x = y]\mathtt{P}]\!] = \mathit{match}\ x\ y\ [\![\mathtt{P}]\!] & & [\![\bar{x}y.\mathtt{P}]\!] = \mathit{out}\ x\ y\ [\![\mathtt{P}]\!] & \\ [\![x(y).\mathtt{P}]\!] = \mathit{in}\ x\ \lambda y.[\![\mathtt{P}]\!] & & [\![(x)\mathtt{P}]\!] = \nu\lambda x.[\![\mathtt{P}]\!] & \end{array}$$

We abbreviate $\nu\lambda x.P$ as simply $\nu x.P$. The one-step transition judgments are trans-

$$
\begin{array}{rrcl}
\textsc{tau:} & \tau\ P \xrightarrow{\tau} P & \triangleq & \top \\
\textsc{in:} & in\ X\ M \xrightarrow{\downarrow X} M & \triangleq & \top \\
\textsc{out:} & out\ x\ y\ P \xrightarrow{\uparrow xy} P & \triangleq & \top \\
\textsc{match:} & match\ x\ x\ P \xrightarrow{A} Q & \triangleq & P \xrightarrow{A} Q \\
 & match\ x\ x\ P \xrightarrow{A} Q & \triangleq & P \xrightarrow{A} Q \\
\textsc{sum:} & P + Q \xrightarrow{A} R & \triangleq & P \xrightarrow{A} R \\
 & P + Q \xrightarrow{A} R & \triangleq & Q \xrightarrow{A} R \\
 & P + Q \xrightarrow{A} R & \triangleq & P \xrightarrow{A} R \\
 & P + Q \xrightarrow{A} R & \triangleq & Q \xrightarrow{A} R \\
\textsc{par:} & P \mid Q \xrightarrow{A} P' \mid Q & \triangleq & P \xrightarrow{A} P' \\
 & P \mid Q \xrightarrow{A} P \mid Q' & \triangleq & Q \xrightarrow{A} Q' \\
 & P \mid Q \xrightarrow{A} \lambda n(M\ n \mid Q) & \triangleq & P \xrightarrow{A} M \\
 & P \mid Q \xrightarrow{A} \lambda n(P \mid N\ n) & \triangleq & Q \xrightarrow{A} N. \\
\textsc{res:} & \nu n.Pn \xrightarrow{A} \nu n.Qn & \triangleq & \nabla n(Pn \xrightarrow{A} Qn) \\
 & \nu n.Pn \xrightarrow{A} \lambda m\ \nu n.P'nm & \triangleq & \nabla n(Pn \xrightarrow{A} P'n) \\
\textsc{open:} & \nu y.My \xrightarrow{\uparrow X} M' & \triangleq & \nabla y(My \xrightarrow{\uparrow Xy} M'y) \\
\textsc{close:} & P \mid Q \xrightarrow{\tau} \nu y.(My \mid Ny) & \triangleq & \exists X.P \xrightarrow{\downarrow X} M \wedge Q \xrightarrow{\uparrow X} N \\
 & P \mid Q \xrightarrow{\tau} \nu y.(My \mid Ny) & \triangleq & \exists X.P \xrightarrow{\uparrow X} M \wedge Q \xrightarrow{\downarrow X} N \\
\textsc{com:} & P \mid Q \xrightarrow{\tau} MY \mid Q' & \triangleq & \exists X.P \xrightarrow{\downarrow X} M \wedge Q \xrightarrow{\uparrow XY} Q' \\
 & P \mid Q \xrightarrow{\tau} P' \mid NY & \triangleq & \exists X.P \xrightarrow{\uparrow XY} P' \wedge Q \xrightarrow{\downarrow X} N
\end{array}
$$

Fig. 2. Definition clauses for the late transition system.

lated to atomic formulas as follows (we overload the symbol $[\![.]\!]$).

$$
\begin{array}{lll}
[\![\mathtt{P} \xrightarrow{xy} \mathtt{Q}]\!] = [\![\mathtt{P}]\!] \xrightarrow{\downarrow xy} [\![\mathtt{Q}]\!] & & [\![\mathtt{P} \xrightarrow{x(y)} \mathtt{Q}]\!] = [\![\mathtt{P}]\!] \xrightarrow{\downarrow x} \lambda y.[\![\mathtt{Q}]\!] \\
[\![\mathtt{P} \xrightarrow{\bar{x}y} \mathtt{Q}]\!] = [\![\mathtt{P}]\!] \xrightarrow{\uparrow xy} [\![\mathtt{Q}]\!] & & [\![\mathtt{P} \xrightarrow{\bar{x}(y)} \mathtt{Q}]\!] = [\![\mathtt{P}]\!] \xrightarrow{\uparrow x} \lambda y.[\![\mathtt{Q}]\!] \\
[\![\mathtt{P} \xrightarrow{\tau} \mathtt{Q}]\!] = [\![\mathtt{P}]\!] \xrightarrow{\tau} [\![\mathtt{Q}]\!] & &
\end{array}
$$

Figure 2 contains a definition, called $\mathbf{D}_\pi$, that encodes the operational semantics of the late transition system for the finite π-calculus. In this specification, free variables are schema variables that are assumed to be universally scoped over the definition clause in which they appear. These schema variables have primitive types such as a, n, and p as well as functional types such as $n \to a$ and $n \to p$.

Notice that, as a consequence of using λ-tree syntax for this specification, the usual side conditions in the original specifications of the π-calculus [Milner et al. 1992] are no longer present. For example, the side condition that $X \neq y$ in the open rule is implicit, since X is outside the scope of y and therefore cannot be instantiated with y (substitutions into logical expressions cannot capture bound variable names). The adequacy of our encoding is stated in the following lemma and proposition (their proofs can be found in [Tiu 2004]).

LEMMA 8. *The function $[\![.]\!]$ is a bijection between α-equivalence classes of process expressions and $\beta\eta$-equivalence classes of terms of type p.*

PROPOSITION 9. *Let* P *and* Q *be processes and α an action. Let $\bar{n}$ be a list of free names containing the free names in* P, Q, *and α. The transition* $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ *is derivable in π-calculus if and only if* $.; . \vdash \nabla\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ *in* $FO\lambda^{\Delta\nabla}$ *with the definition* $\mathbf{D}_\pi$.

If our goal was only to correctly encode one-step transitions for the π-calculus then we would need neither ∇ nor definitions. In particular, let $\mathbf{D}_\pi^\forall$ be the result of replacing all ∇ quantifiers in $\mathbf{D}_\pi$ with $\forall$ quantifiers. A slight generalization of Proposition 4 (see [Miller and Tiu 2005; Tiu 2004]) allows us to conclude that $.; . \vdash \nabla\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ in $FO\lambda^{\Delta\nabla}$ with the definition $\mathbf{D}_\pi$ if and only if $.; . \vdash \forall\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ in $FO\lambda^{\Delta\nabla}$ with the definition $\mathbf{D}_\pi^\forall$. Furthermore, we can also do with the simpler notions of *theory* or *assumptions* and not *definition*. In particular, let $\mathbf{P}_\pi$ be the set of implications that result from changing all definition clauses in $\mathbf{D}_\pi^\forall$ into reverse implications (i.e., the head is implied by the body). We can then conclude that $.; . \vdash \forall\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ in $FO\lambda^{\Delta\nabla}$ with the definition $\mathbf{D}_\pi^\forall$ if and only if $.; \mathbf{P}_\pi \vdash \forall\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ in intuitionistic (and classical) logic. In fact, such a specification of the one-step transitions in the π-calculus as a theory without ∇ dates back to at least Miller and Palamidessi [1999].

Definitions and ∇ are needed, however, for proving non-Horn properties (that is, properties requiring a strong notion of negation). The following proposition is a dual of Proposition 9. Its proof can be found in the appendix.

PROPOSITION 10. *Let* P *and* Q *be processes and α an action. Let $\bar{n}$ be a list of free names containing the free names in* P, Q, *and α. The transition* $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ *is not derivable in π-calculus if and only if* $.; . \vdash \neg\nabla\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ *in* $FO\lambda^{\Delta\nabla}$ *with the definition* $\mathbf{D}_\pi$.

The following example illustrates how a negation can be proved in $FO\lambda^{\Delta\nabla}$. When writing encoded process expressions, we shall use, instead, the syntax of π-calculus along with the usual abbreviations: for example, when a name z is used as a prefix, it denotes the prefix $z(w)$ where w is vacuous in its scope; when a name $\bar{z}$ is used as a prefix it denotes the output prefix $\bar{z}a$ for some fixed name a. We also abbreviate $(y)\bar{x}y.P$ as $\bar{x}(y).P$ and the process term 0 is omitted if it appears as the continuation of a prefix. We assume that the operators $|$ and $+$ associate to the right, *e.g.*, we write $P + Q + R$ to denote $P + (Q + R)$.

EXAMPLE 11. Consider the process $(y)([x = y]\bar{x}z)$, which could be the continuation of some other process which inputs x on some channel, *e.g.*, $a(x).(y)[x = y]\bar{x}z$. Since the bound variable y is different from any name substituted for x, that process cannot make a transition and the following formula should be provable.

$$\forall x\forall z\forall Q\forall\alpha.[((y)[x = y](\bar{x}z) \xrightarrow{\alpha} Q) \supset \bot]$$

Since y is bound inside the scope of x, no instantiation for x can be equal to y. The formal derivation of the above formula is (ignoring the initial uses of $\supset\mathcal{R}$ and

$\forall\mathcal{R}$):

$$\dfrac{\dfrac{\dfrac{}{\{x,z,Q',\alpha\}\,;\, y \triangleright ([x=y](\bar{x}z.0) \xrightarrow{\alpha} Q'y) \vdash \bot}\, def\mathcal{L}}{\{x,z,Q',\alpha\}\,;\, . \triangleright \nabla y.([x=y](\bar{x}z.0) \xrightarrow{\alpha} Q'y) \vdash \bot}\, \nabla\mathcal{L}}{\{x,z,Q,\alpha\}\,;\, . \triangleright ((y)[x=y](\bar{x}z.0) \xrightarrow{\alpha} Q) \vdash \bot}\, def\mathcal{L}$$

The success of the topmost instance of $def\mathcal{L}$ depends on the failure of the unification problem $\lambda y.x = \lambda y.y$. Notice that the scoping of term-level variables is maintained at the proof-level by the separation of (global) eigenvariables and (locally bound) generic variables. The "newness" of y is internalized as a λ-abstraction and, hence, it is not subject to instantiation.

The ability to prove a negation is implied by any proof system that can also prove bisimulation for (at least finite) π-calculus: for example, the negation above holds because the process $(y)([x=y]\bar{x}z)$ is bisimilar to 0 (see Section 5).

We shall soon need syntactic equality as a predicate in the $FO\lambda^{\Delta\nabla}$ logic. This predicate is defined using the definition containing the single clause $X = X \stackrel{\triangle}{=} \top$. Note that the symbol $=$ here is a predicate symbol written in infix notation. The inequality $x \neq y$ is an abbreviation for $x = y \supset \bot$.

5. LOGICAL SPECIFICATIONS OF STRONG BISIMILARITY

We consider specifying three notions of bisimilarity tied to the late transition system: the strong early bisimilarity, the strong late bisimilarity and the strong open bisimilarity. As it turns out, the definition clauses corresponding to strong late and strong open bisimilarity coincide. Their essential differences are in the quantification of free names and in the presence (or the absence) of the axiom of excluded middle on the equality of names. The difference between early and late bisimulation is tied to the scope the quantification of names in the case involving bound input (see the definitions below). The original definitions of early, late, and open bisimilarity are given in [Milner et al. 1992; Sangiorgi and Walker 2001]. Here we choose to make the side conditions explicit, instead of adopting the bound variable convention in [Sangiorgi and Walker 2001].

Given a relation on processes $\mathcal{R}$, we write $P\ \mathcal{R}\ Q$ to denote $(P,Q) \in \mathcal{R}$.

Definition 12. A process relation $\mathcal{R}$ is a *strong late bisimulation* if $\mathcal{R}$ is symmetric and whenever $\mathtt{P}\ \mathcal{R}\ \mathtt{Q}$,

(1) if $\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'$ and α is a free action, then there is $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{\alpha} \mathtt{Q}'$ and $\mathtt{P}'\ \mathcal{R}\ \mathtt{Q}'$;

(2) if $\mathtt{P} \xrightarrow{x(z)} \mathtt{P}'$ and $z \notin \mathrm{n}(\mathtt{P},\mathtt{Q})$ then there is $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{x(z)} \mathtt{Q}'$ and, for every name y, $\mathtt{P}'[y/z]\ \mathcal{R}\ \mathtt{Q}'[y/z]$; and

(3) if $\mathtt{P} \xrightarrow{\bar{x}(z)} \mathtt{P}'$ and $z \notin \mathrm{n}(\mathtt{P},\mathtt{Q})$ then there is $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{\bar{x}(z)} \mathtt{Q}'$ and $\mathtt{P}'\ \mathcal{R}\ \mathtt{Q}'$.

The processes $\mathtt{P}$ and $\mathtt{Q}$ are *strong late bisimilar*, written $\mathtt{P} \sim_l \mathtt{Q}$, if there is a strong late bisimulation $\mathcal{R}$ such that $\mathtt{P}\ \mathcal{R}\ \mathtt{Q}$.

DEFINITION 13. A process relation $\mathcal{R}$ is a *strong early bisimulation* if $\mathcal{R}$ is symmetric and whenever $\mathtt{P}\ \mathcal{R}\ \mathtt{Q}$,

(1) if $\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'$ and α is a free action, then there is $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{\alpha} \mathtt{Q}'$ and $\mathtt{P}'\ \mathcal{R}\ \mathtt{Q}'$,
(2) if $\mathtt{P} \xrightarrow{x(z)} \mathtt{P}'$ and $z \notin \mathrm{n}(\mathtt{P},\mathtt{Q})$ then for every name y, there is $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{x(z)} \mathtt{Q}'$ and $\mathtt{P}'[y/z]\ \mathcal{R}\ \mathtt{Q}'[y/z]$,
(3) if $\mathtt{P} \xrightarrow{\bar{x}(z)} \mathtt{P}'$ and $z \notin \mathrm{n}(\mathtt{P},\mathtt{Q})$ then there is $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{\bar{x}(z)} \mathtt{Q}'$ and $\mathtt{P}'\ \mathcal{R}\ \mathtt{Q}'$.

The processes $\mathtt{P}$ and $\mathtt{Q}$ are *strong early bisimilar*, written $\mathtt{P} \sim_e \mathtt{Q}$, if there is a strong early bisimulation $\mathcal{R}$ such that $\mathtt{P}\ \mathcal{R}\ \mathtt{Q}$.

DEFINITION 14. A *distinction* D is a finite symmetric and irreflexive relation on names. A substitution θ *respects* a distinction D if $(x,y) \in D$ implies $x\theta \neq y\theta$. We refer to the substitution θ as a *D-substitution*. Given a distinction D and a D-substitution θ, the result of applying θ to all variables in D, written $D\theta$, is another distinction. We denote with $\mathrm{fn}(D)$ the set of free names occurring in D.

Since distinctions are symmetric by definition, when we enumerate a distinction, we often omit the symmetric part of the distinction. For instance, we shall write $\{(a,b)\}$ to mean the distinction $\{(a,b),(b,a)\}$, and we shall also write $D \cup (S \times T)$, for some distinction D and finite sets of names S and T, to mean the distinction $D \cup (S \times T) \cup (T \times S)$.

Following Sangiorgi [Sangiorgi 1996], we use a set of relations, each indexed by a distinction, to define open bisimulation.

DEFINITION 15. The set $\mathcal{S} = \{\mathcal{S}_D\}_D$ of process relations is an indexed open bisimulation if for each D, $\mathcal{S}_D$ is symmetric and for every θ that respects D, $\mathtt{P}\ \mathcal{S}_D\ \mathtt{Q}$ implies:

(1) if $\mathtt{P}\theta \xrightarrow{\alpha} \mathtt{P}'$ and α is a free action, then there is $\mathtt{Q}'$ such that $\mathtt{Q}\theta \xrightarrow{\alpha} \mathtt{Q}'$ and $\mathtt{P}'\mathcal{S}_{D\theta}\mathtt{Q}'$,
(2) if $\mathtt{P}\theta \xrightarrow{x(z)} \mathtt{P}'$ and $z \notin \mathrm{n}(\mathtt{P}\theta,\mathtt{Q}\theta)$ then there is $\mathtt{Q}'$ such that $\mathtt{Q}\theta \xrightarrow{x(z)} \mathtt{Q}'$ and $\mathtt{P}'\ \mathcal{S}_{D\theta}\ \mathtt{Q}'$,
(3) if $\mathtt{P}\theta \xrightarrow{\bar{x}(z)} \mathtt{P}'$ and $z \notin \mathrm{n}(\mathtt{P}\theta,\mathtt{Q}\theta)$ then there is $\mathtt{Q}'$ such that $\mathtt{Q}\theta \xrightarrow{\bar{x}(z)} \mathtt{Q}'$ and $\mathtt{P}'\ \mathcal{S}_{D'}\ \mathtt{Q}'$ where $D' = D\theta \cup (\{z\} \times \mathrm{fn}(\mathtt{P}\theta,\mathtt{Q}\theta,D\theta))$.

The processes $\mathtt{P}$ and $\mathtt{Q}$ are *strong open D-bisimilar*, written $\mathtt{P} \sim_o^D \mathtt{Q}$, if there is an indexed open bisimulation $\mathcal{S}$ such that $P\ \mathcal{S}_D\ \mathtt{Q}$. The processes $\mathtt{P}$ and $\mathtt{Q}$ are *strong open bisimilar* if $\mathtt{P} \sim_o^{\emptyset} \mathtt{Q}$.

Note that we strengthen a bit the condition 3 in Definition 15 to include the distinction $(\{z\} \times \mathrm{fn}(D\theta))$. Strengthening the distinction this way does not change the open bisimilarity, as noted in [Sangiorgi and Walker 2001], but in our encoding of open bisimulation, the distinction D is part of the specification and the modified definition above helps us account for names better.

Early and late bisimulation can be specified in $FO\lambda^{\Delta\nabla}$ using the definition clauses in Figure 3. The definition clause for open bisimulation is the same as

$$
\begin{aligned}
\mathit{ebisim}\ P\ Q \triangleq\ & \forall A \forall P'\ [P \xrightarrow{A} P' \supset \exists Q'.\ Q \xrightarrow{A} Q' \wedge \mathit{ebisim}\ P'\ Q'] \wedge \\
& \forall A \forall Q'\ [Q \xrightarrow{A} Q' \supset \exists P'.\ P \xrightarrow{A} P' \wedge \mathit{ebisim}\ Q'\ P'] \wedge \\
& \forall X \forall P'\ [P \xrightarrow{\downarrow X} P' \supset \forall w \exists Q'.\ Q \xrightarrow{\downarrow X} Q' \wedge \mathit{ebisim}\ (P'w)\ (Q'w)] \wedge \\
& \forall X \forall Q'\ [Q \xrightarrow{\downarrow X} Q' \supset \forall w \exists P'.\ P \xrightarrow{\downarrow X} P' \wedge \mathit{ebisim}\ (Q'w)\ (P'w)] \wedge \\
& \forall X \forall P'\ [P \xrightarrow{\uparrow X} P' \supset \exists Q'.\ Q \xrightarrow{\uparrow X} Q' \wedge \nabla w.\mathit{ebisim}\ (P'w)\ (Q'w)] \wedge \\
& \forall X \forall Q'\ [Q \xrightarrow{\uparrow X} Q' \supset \exists P'.\ P \xrightarrow{\uparrow X} P' \wedge \nabla w.\mathit{ebisim}\ (Q'w)\ (P'w)]
\end{aligned}
$$

$$
\begin{aligned}
\mathit{lbisim}\ P\ Q \triangleq\ & \forall A \forall P'\ [P \xrightarrow{A} P' \supset \exists Q'.\ Q \xrightarrow{A} Q' \wedge \mathit{lbisim}\ P'\ Q'] \wedge \\
& \forall A \forall Q'\ [Q \xrightarrow{A} Q' \supset \exists P'.\ P \xrightarrow{A} P' \wedge \mathit{lbisim}\ Q'\ P'] \wedge \\
& \forall X \forall P'\ [P \xrightarrow{\downarrow X} P' \supset \exists Q'.\ Q \xrightarrow{\downarrow X} Q' \wedge \forall w.\mathit{lbisim}\ (P'w)\ (Q'w)] \wedge \\
& \forall X \forall Q'\ [Q \xrightarrow{\downarrow X} Q' \supset \exists P'.\ P \xrightarrow{\downarrow X} P' \wedge \forall w.\ \mathit{lbisim}\ (Q'w)\ (P'w)] \wedge \\
& \forall X \forall P'\ [P \xrightarrow{\uparrow X} P' \supset \exists Q'.\ Q \xrightarrow{\uparrow X} Q' \wedge \nabla w.\ \mathit{lbisim}\ (P'w)\ (Q'w)] \wedge \\
& \forall X \forall Q'\ [Q \xrightarrow{\uparrow X} Q' \supset \exists P'.\ P \xrightarrow{\uparrow X} P' \wedge \nabla w.\ \mathit{lbisim}\ (Q'w)\ (P'w)]
\end{aligned}
$$

Fig. 3. Specification of strong early, *ebisim*, and late, *lbisim*, bisimulations.

the one for late bisimulation. The exact relationship between these definitions and the bisimulation relations repeated above will be stated later in this section.

The definition clauses shown in the figures do not fully capture bisimulations since they do not yet address the notion of distinction among names. To illustrate this issue now, note that when reasoning about the specifications of early/late bisimulation, we encode free names as ∇-quantified variables whereas in the specification of open bisimulation we encode free names as $\forall$-quantified variables. For example, the processes $Pxy = (x|\bar{y})$ and $Qxy = (x.\bar{y} + \bar{y}.x)$ are late bisimilar. The corresponding encoding in $FO\lambda^{\Delta\nabla}$ would be $\nabla x \nabla y.\mathit{lbisim}\ (Pxy)\ (Qxy)$. The free names x and y should not be $\forall$-quantified for the following, simple reason: in logic we have the implication $\forall x \forall y\ \mathit{lbisim}\ (Pxy)\ (Qxy) \supset \forall z\ \mathit{lbisim}\ (Pzz)\ (Qzz)$. That is, either $\forall x \forall y\ \mathit{lbisim}\ (Pxy)\ (Qxy)$ is not provable, or it is provable and we have a proof of $\forall z\ \mathit{lbisim}\ (Pzz)\ (Qzz)$. In either case we lose the adequacy of the encoding.

To fully capture the notion of early and late bisimulation, we shall need an additional axiom of excluded middle on names; that is, given any two names, it should be the case that they are equal or not. This basic assumption on the ability to decide the equality among names is one of the differences between open and late bisimulation. Consider, for example, the processes (taken from [Sangiorgi 1996])

$$P = x(u).(\tau.\tau + \tau) \qquad \text{and} \qquad Q = x(u).(\tau.\tau + \tau + \tau.[u = z]\tau).$$

As shown in [Sangiorgi 1996] P and Q are late bisimilar but not open bisimilar: late bisimulation makes use of a case analysis whether or not the input name u is equal to z or not. The axiom of excluded middle on names is simply expressed as the formula $\forall x \forall y (x = y \vee x \neq y)$. Note that since we allow dynamic creation of scoped names (via ∇), we must also state this axiom for arbitrary extension of local signatures. The following set collect together such generalized excluded

middle formulas:

$$\mathcal{E} = \{\nabla n_1 \cdots \nabla n_k \forall x \forall y (x = y \vee x \neq y) \mid k \geq 0\}.$$

We shall write $\mathcal{X} \subseteq_f \mathcal{E}$ to indicate that $\mathcal{X}$ is a finite subset of $\mathcal{E}$.

The following theorem states the soundness and completeness of the *ebisim* and *lbisim* specifications with respect to the notions of early and late bisimilarity in the π-calculus. By soundness we mean that, given a pair of processes P and Q, if the encoding of the late (early) bisimilarity is provable in $FO\lambda^{\Delta\nabla}$ then the processes P and Q are late (early) bisimilar. Completeness is the converse. The soundness and completeness of the open bisimilarity encoding is presented at the end of this section, where we consider the encoding of the notion of distinction in π-calculus.

THEOREM 16. *Let* P *and* Q *be two processes and let* $\bar{n}$ *be the free names in* P *and* Q. *Then* $\mathtt{P} \sim_l \mathtt{Q}$ *if and only if the sequent* $.; \mathcal{X} \vdash \nabla \bar{n}.lbisim\ \mathtt{P}\ \mathtt{Q}$ *is provable for some* $\mathcal{X} \subseteq_f \mathcal{E}$.

THEOREM 17. *Let* P *and* Q *be two processes and let* $\bar{n}$ *be the free names in* P *and* Q. *Then* $\mathtt{P} \sim_e \mathtt{Q}$ *if and only if the sequent* $.; \mathcal{X} \vdash \nabla \bar{n}.ebisim\ \mathtt{P}\ \mathtt{Q}$ *is provable for some* $\mathcal{X} \subseteq_f \mathcal{E}$.

It is well-known that the late bisimulation relation is not a congruence since it is not preserved by the input prefix. Part of the reason why the congruence property fails is that in the late bisimilarity there is no syntactic distinction made between instantiable names and non-instantiable names. Addressing this difference between variables is one of the motivations behind the introduction of of distinctions and open bisimulation. There is another important difference between open and late bisimulation; in open bisimulation names are instantiated *lazily*, i.e., only when needed. The lazy instantiation of names is intrinsic in $FO\lambda^{\Delta\nabla}$; eigenvariables are instantiated only when applying the *def*$\mathcal{L}$-rule. The syntactic distinction between instantiable and non-instantiable names are reflected in $FO\lambda^{\Delta\nabla}$ by the difference between the quantifier $\forall$ and ∇. The alternation of quantifiers in $FO\lambda^{\Delta\nabla}$ gives rise to a particular kind of distinction, the precise definition of which is given below.

DEFINITION 18. A *quantifier prefix* is a list $\mathcal{Q}_1 x_1 \mathcal{Q}_2 x_2 \ldots \mathcal{Q}_n x_n$ for some $n \geq 0$, where $\mathcal{Q}_i$ is either ∇ or $\forall$. If $\mathcal{Q}\bar{x}$ is the above quantifier prefix, then the $\mathcal{Q}\bar{x}$-*distinction* is the distinction

$$\{(x_i, x_j), (x_j, x_i) \mid i \neq j \text{ and } \mathcal{Q}_i = \mathcal{Q}_j = \nabla, \text{ or } i < j \text{ and } \mathcal{Q}_i = \forall \text{ and } \mathcal{Q}_j = \nabla\}.$$

Notice that if $\mathcal{Q}\bar{x}$ consists only of universal quantifiers then the $\mathcal{Q}\bar{x}$-distinction is empty. Obviously, the alternation of quantifiers does not capture all possible distinction, *e.g.*, the distinction

$$\{(x, y), (y, x), (x, z), (z, x), (u, z), (z, u)\}$$

does not correspond to any quantifier prefix. However, we can encode the full notion of distinction by explicit encoding of the unequal pairs, as shown later.

It is interesting to see the effect of substitutions on D when D corresponds to a prefix $\mathcal{Q}\bar{x}$. Suppose $\mathcal{Q}\bar{x}$ is the prefix $\mathcal{Q}_1 \bar{u} \forall x \mathcal{Q}_2 \bar{v} \forall y \mathcal{Q}_3 \bar{w}$. Since any two $\forall$-quantified variables are not made distinct in the definition of $\mathcal{Q}\bar{x}$ prefix, there is a θ which respects D and which can identify x and y. Applying θ to D changes D to some D'

which corresponds to the prefix $\mathcal{Q}_1\bar{u}\forall z\mathcal{Q}_2\bar{v}\mathcal{Q}_3\bar{w}$. Interestingly, these two prefixes are related by logical implication:

$$\mathcal{Q}_1\bar{u}\forall x\mathcal{Q}_2\bar{v}\forall y\mathcal{Q}_3\bar{w}.P \supset \mathcal{Q}_1\bar{u}\forall z\mathcal{Q}_2\bar{v}\mathcal{Q}_3\bar{w}.P[z/x, z/y]$$

for any formula P. This observation suggests the following lemma.

LEMMA 19. *Let D be a $\mathcal{Q}\bar{x}$-distinction and let θ be a D-substitution. Then the distinction $D\theta$ corresponds to some prefix $\mathcal{Q}'\bar{y}$ such that $\mathcal{Q}\bar{x}.P \supset \mathcal{Q}\bar{y}.P\theta$ for any formula P such that* $\mathrm{fv}(P) \subseteq \{\bar{x}\}$.

DEFINITION 20. Let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a distinction. The distinction D is translated as the formula $[\![D]\!] = x_1 \neq y_1 \wedge \ldots \wedge x_n \neq y_n$. If $n = 0$ then $[\![D]\!]$ is the logical constant $\top$ (empty conjunction).

THEOREM 21. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes, let D be a distinction and let $\mathcal{Q}\bar{x}$ be a quantifier prefix, where $\bar{x}$ contains the free names in* $\mathtt{P}, \mathtt{Q}$ *and D. If the formula $\mathcal{Q}\bar{x}.([\![D]\!] \supset$ lbisim* $\mathtt{P}\ \mathtt{Q}$*) is provable then* $\mathtt{P} \sim_o^{D'} \mathtt{Q}$*, where D' is the union of D and the $\mathcal{Q}\bar{x}$-distinction.*

THEOREM 22. *If* $\mathtt{P} \sim_o^D \mathtt{Q}$ *then the formula $\forall\bar{x}.[\![D]\!] \supset$ lbisim* $\mathtt{P}\ \mathtt{Q}$*, where $\bar{x}$ are the free names in* $\mathtt{P}, \mathtt{Q}$ *and D, is provable.*

If a distinction D corresponds to a quantifier prefix $\mathcal{Q}\vec{x}$, then it is easy to show that $\mathcal{Q}\vec{x}.[\![D]\!]$ is derivable in $FO\lambda^{\Delta\nabla}$. Therefore, we can state more concisely the adequacy result for the class of D-open bisimulation in which D corresponds to a quantifier prefix, which is a corollary of Theorem 21 and Theorem 22.

COROLLARY 23. *Let D be a distinction, let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes and let $\mathcal{Q}\vec{x}$ be a quantifier prefix such that $\vec{x}$ contains the free names of D,* $\mathtt{P}$ *and* $\mathtt{Q}$*, and D corresponds to the $\mathcal{Q}\vec{x}$-distinction. Then* $\mathtt{P} \sim_o^D \mathtt{Q}$ *if and only if* $\vdash \mathcal{Q}\vec{x}.$*lbisim* $\mathtt{P}\ \mathtt{Q}$.

Notice the absence of the excluded middle assumption on names in the specification of open bisimulation. Since $FO\lambda^{\Delta\nabla}$ is intuitionistic, this difference between late and open bisimulation is easily observed. This would not be the case if the specification logic were classical. Since the axiom of excluded middle is present as well in the specification of early bisimulation (Theorem 17), one might naturally wonder if there is a meaningful notion of bisimulation obtained from removing the excluded middle in the specification of early bisimulation and $\forall$-quantify the free names. In other words, we would like to see if there is a notion of "open-early" bisimulation. In fact, the resulting bisimulation relation is exactly the same as open "late" bisimulation.

THEOREM 24. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes and let $\bar{n}$ be the free names in* $\mathtt{P}$ *and* $\mathtt{Q}$*. Then $\forall\bar{n}.$lbisim* $\mathtt{P}\ \mathtt{Q}$ *is provable if and only if $\forall\bar{n}.$ebisim* $\mathtt{P}\ \mathtt{Q}$ *is provable.*

To conclude this section, we should explicitly compare the two specifications of early bisimulation in Definition 13 and in Theorem 17, the two specifications of late bisimulation in Definition 12 and in Theorem 16 and the two specifications of open bisimulation in Definition 15 and in Corollary 23. Notice that those specifications that rely on logic are written without the need for any explicit conditions on variable names or any need to mention distinctions explicitly. These various conditions are,

(a) Propositional connectives and *basic* modality:

$$
\begin{array}{lll}
\text{(true :)} & P \models \text{true} & \triangleq \ \top. \\
\text{(and :)} & P \models A \& B & \triangleq \ P \models A \wedge P \models B. \\
\text{(or :)} & P \models A \hat{\vee} B & \triangleq \ P \models A \vee P \models B. \\
\text{(match :)} & P \models \langle X \doteq X \rangle A & \triangleq \ P \models A. \\
\text{(match :)} & P \models [X \doteq Y] A & \triangleq \ (X = Y) \supset P \models A. \\
\text{(free :)} & P \models \langle X \rangle A & \triangleq \ \exists P' (P \xrightarrow{X} P' \wedge P' \models A). \\
\text{(free :)} & P \models [X] A & \triangleq \ \forall P' (P \xrightarrow{X} P' \supset P' \models A). \\
\text{(out :)} & P \models \langle \uparrow X \rangle A & \triangleq \ \exists P' (P \xrightarrow{\uparrow X} P' \wedge \nabla y. P' y \models A y). \\
\text{(out :)} & P \models [\uparrow X] A & \triangleq \ \forall P' (P \xrightarrow{\uparrow X} P' \supset \nabla y. P' y \models A y). \\
\text{(in :)} & P \models \langle \downarrow X \rangle A & \triangleq \ \exists P' (P \xrightarrow{\downarrow X} P' \wedge \exists y. P' y \models A y). \\
\text{(in :)} & P \models [\downarrow X] A & \triangleq \ \forall P' (P \xrightarrow{\downarrow X} P' \supset \forall y. P' y \models A y).
\end{array}
$$

(b) *Late* modality:

$$
\begin{array}{l}
P \models \langle \downarrow X \rangle^l A \ \triangleq \ \exists P' (P \xrightarrow{\downarrow X} P' \wedge \forall y. P' y \models A y). \\
P \models [\downarrow X]^l A \ \triangleq \ \forall P' (P \xrightarrow{\downarrow X} P' \supset \exists y. P' y \models A y).
\end{array}
$$

(c) *Early* modality:

$$
\begin{array}{l}
P \models \langle \downarrow X \rangle^e A \ \triangleq \ \forall y \exists P' (P \xrightarrow{\downarrow X} P' \wedge P' y \models A y). \\
P \models [\downarrow X]^e A \ \triangleq \ \exists y \forall P' (P \xrightarrow{\downarrow X} P' \supset P' y \models A y).
\end{array}
$$

Fig. 4. Modal logics for π-calculus in λ-tree syntax

of course, present in the detailed description of the proof theory of our logic, but it seems to be very desirable to push the details of variable names, substitutions, free and bound-occurrence, and equalities into logic, where they have elegant and standard solutions.

6. SPECIFICATION OF MODAL LOGICS

We now present the modal logics for the π-calculus that were introduced in [Milner et al. 1993]. In order not to confuse meta-level ($FO\lambda^{\Delta\nabla}$) formulas (or connectives) with the formulas (connectives) of the modal logics under consideration, we shall refer to the latter as object formulas (respectively, object connectives). We shall work only with object formulas that are in negation normal form, i.e., negation appears only at the level of atomic object formulas: since there are no atomic formulas in these modal logics (in particular, true or false are not atomic), negations do not appear in modal formulas. The syntax of the object formulas is as follows.

$$
\begin{array}{rl}
\mathtt{A} ::= & \text{true} \mid \text{false} \mid \mathtt{A} \wedge \mathtt{A} \mid \mathtt{A} \vee \mathtt{A} \mid [x = z]\mathtt{A} \mid \langle x = z \rangle \mathtt{A} \\
& \mid \langle \alpha \rangle \mathtt{A} \mid [\alpha]\mathtt{A} \mid \langle \bar{x}(y) \rangle \mathtt{A} \mid [\bar{x}(y)]\mathtt{A} \mid \langle x(y) \rangle \mathtt{A} \mid [x(y)]\mathtt{A} \\
& \mid \langle x(y) \rangle^L \mathtt{A} \mid [x(y)]^L \mathtt{A} \mid \langle x(y) \rangle^E \mathtt{A} \mid [x(y)]^E \mathtt{A}
\end{array}
$$

In each of the formulas $\langle \bar{x}(y) \rangle \mathtt{A}$, $\langle x(y) \rangle \mathtt{A}$, $\langle x(y) \rangle^L \mathtt{A}$ and $\langle x(y) \rangle^E \mathtt{A}$ (and their dual 'boxed'-formulas), the occurrence of y in parentheses is a binding occurrence whose scope is $\mathtt{A}$. We use $\mathtt{A}$, $\mathtt{B}$, $\mathtt{C}$, and $\mathtt{D}$ to range over object formulas. Note that we consider only finite conjunctions since the transition system we are considering is finitely branching, and, therefore, an infinite conjunction is not needed (as noted

in [Milner et al. 1993]). Note also that we do not consider free input modality $\langle xy\rangle$ since we restrict ourselves to late transition system (but adding early transition rules and free input modality does not pose any difficulty). We consider object formulas equivalent up to renaming of bound variables.

To encode object formulas we introduce the type o' to denote such formulas and introduce the following constants for encoding the object connectives: true and false of type o'; & and $\hat{\vee}$ of type $o' \to o' \to o'$; $\langle\cdot\doteq\cdot\rangle\cdot$ and $[\cdot\doteq\cdot]\cdot$ of type $n \to n \to o' \to o$; $\langle\cdot\rangle\cdot$ and $[\cdot]\cdot$ of type $a \to o' \to o'$; and $\langle\downarrow\cdot\rangle\cdot$, $[\downarrow\cdot]\cdot$, $\langle\downarrow\cdot\rangle^l\cdot$, $[\downarrow\cdot]^l\cdot$, $\langle\downarrow\cdot\rangle^e\cdot$, and $[\downarrow\cdot]^e\cdot$ of type $n \to (n \to o') \to o'$. The translation of object formulas to λ-tree syntax is given in the following definition.

Definition 25. The following function $[\![.]\!]$ translates object formulas to $\beta\eta$-long normal terms of type o'.

$$
\begin{array}{ll}
[\![\mathrm{true}]\!] = \mathrm{true} & [\![\mathrm{false}]\!] = \mathrm{false} \\
[\![\mathtt{A} \wedge \mathtt{B}]\!] = [\![\mathtt{A}]\!] \& [\![\mathtt{B}]\!] & [\![\mathtt{A} \vee \mathtt{B}]\!] = [\![\mathtt{A}]\!] \hat{\vee} [\![\mathtt{B}]\!] \\
[\![[x = y]\mathtt{A}]\!] = [x \doteq y][\![\mathtt{A}]\!] & [\![\langle x = y\rangle \mathtt{A}]\!] = \langle x \doteq y\rangle [\![\mathtt{A}]\!] \\
[\![\langle\alpha\rangle \mathtt{A}]\!] = \langle\alpha\rangle [\![\mathtt{A}]\!] & [\![[\alpha]\mathtt{A}]\!] = [\alpha][\![\mathtt{A}]\!] \\
[\![\langle x(y)\rangle \mathtt{A}]\!] = \langle\downarrow x\rangle(\lambda y [\![\mathtt{A}]\!]) & [\![[x(y)]\mathtt{A}]\!] = [\downarrow x](\lambda y [\![\mathtt{A}]\!]) \\
[\![\langle x(y)\rangle^L \mathtt{A}]\!] = \langle\downarrow x\rangle^l(\lambda y [\![\mathtt{A}]\!]) & [\![[x(y)]^L \mathtt{A}]\!] = [\downarrow x]^l(\lambda y [\![\mathtt{A}]\!]) \\
[\![\langle x(y)\rangle^E \mathtt{A}]\!] = \langle\downarrow x\rangle^e(\lambda y [\![\mathtt{A}]\!]) & [\![[x(y)]^E \mathtt{A}]\!] = [\downarrow x]^e(\lambda y [\![\mathtt{A}]\!])
\end{array}
$$

The satisfaction relation $\models$ between processes and formulas are encoded using the same symbol, which is given the type $p \to o' \to o$. This satisfaction relation is defined by the clauses in Figure 4. This definition, called $\mathcal{DA}$, corresponds to the modal logic $\mathcal{A}$ defined in [Milner et al. 1993]. Notice that $\mathcal{DA}$ interprets object-level disjunction and conjunction with, respectively, meta-level disjunction and conjunction. Since the modal logic $\mathcal{A}$ is classical and the meta-logic $FO\lambda^{\Delta\nabla}$ is intuitionistic, one may wonder whether such an encoding is complete. But since we consider only object formulas in negation normal form and since there are no atomic formulas nor occurrences of negations in the formulas, classical and intuitionistic provability coincide for the non-modal fragment of $\mathcal{A}$. The definition $\mathcal{DA}$ is, however, incomplete for the full logic $\mathcal{A}$, in the sense that there are true assertion of modal logics that are not provable using this definition alone. Using the 'box' modality, one can still encode some limited forms of negation, e.g., inequality of names. For instance, the modal judgment

$$x(y).x(z).0 \models \langle x(y)\rangle\langle x(z)\rangle(\langle x = z\rangle \mathrm{true}\ \hat{\vee}\ [x = z]\mathrm{false}),$$

which essentially asserts that any two names are equal or unequal, is valid in $\mathcal{A}$, but its encoding in $FO\lambda^{\Delta\nabla}$ is not provable without additional assumptions. It turns out that, as in the case with the specification of late bisimulation, the only assumption we need to assure completeness is the axiom of excluded middle on the equality of names: $\forall x \forall y. x = y \vee x \neq y$. Again, as in the specification of late bisimulation, we must also state this axiom for arbitrary extension of local signatures. The adequacy of the specification of modal logics is stated in the following theorem.

Theorem 26. *Let* $\mathtt{P}$ *be a process, let* $\mathtt{A}$ *be an object formula. Then* $\mathtt{P} \models \mathtt{A}$ *if and only if for some list* $\bar{n}$ *such that* $\mathrm{fn}(\mathtt{P}, \mathtt{A}) \subseteq \{\bar{n}\}$ *and some* $\mathcal{X} \subseteq_f \mathcal{E}$*, the sequent* $\mathcal{X} \vdash \nabla\bar{n}.([\![\mathtt{P}]\!] \models [\![\mathtt{A}]\!])$ *is provable in* $FO\lambda^{\Delta\nabla}$ *with definition* $\mathcal{DA}$*.*

The adequacy result stated in Theorem 26 subsumes the adequacy for the specifications of the sublogics of $\mathcal{A}$. Note that we quantify free names in the process-formula pair in the above theorem since we do not assume any constants of type n. Of course, such constants can be introduced without affecting the provability of the satisfaction judgments, but for simplicity in the meta-theory we consider the more uniform approach using ∇-quantified variables to encode names in process and object formulas, just like what we did with the specification of late bisimulation.

Notice that the list of names $\bar{n}$ in Theorem 26 can contain more than just the free names of $\mathtt{P}$ and $\mathtt{A}$. This is important for the adequacy of the specification, since in the modal logics for π-calculus, we can specify a modal formula $\mathtt{A}$ and a process $\mathtt{P}$ such that the assertion $\mathtt{P} \models \mathtt{A}$ is true only if there exists a new name which is not among the free names of both $\mathtt{P}$ and $\mathtt{A}$. Consider, for example, the assertion

$$a(x).0 \models [a(x)]^L[x = a]\text{false}$$

and its encoding in $FO\lambda^{\Delta\nabla}$ as the formula

$$in\ a\ (\lambda x.0) \models [\downarrow a]^l(\lambda x.[x \doteq a]\text{false}).$$

If we do not allow extra new names in the quantifier prefix in Theorem 26, then we would have to prove the formula

$$\nabla a.\big(in\ a\ (\lambda x.0) \models [\downarrow a]^l(\lambda x.[x \doteq a]\text{false})\big).$$

It is easy to see that provability of this formula reduces to provability of

$$\nabla a \exists x.\big(0 \models [x \doteq a]\text{false}\big).$$

Since we do not assume any constants of type n, the only way to prove this would be to instantiate x with a, hence,

$$\nabla a.(0 \models [a \doteq a]\text{false}) \quad \text{and} \quad \nabla a.(a = a) \supset 0 \models \text{false}.$$

must be provable. This is in turn equivalent to $\nabla a.0 \models \text{false}$ which should not be provable for the adequacy result to hold. The key step here is the instantiation of $\exists x$. For the original formula to be provable, x has to be instantiated with a name that is distinct from a. This can be done only if we allow extra names in the quantifier prefix: for example, the following formula is provable.

$$\nabla a \nabla b.\big(in\ a\ (\lambda x.0) \models [\downarrow a]^l(\lambda x.[x \doteq a]\text{false})\big)$$

In [Milner et al. 1993], late bisimulation was characterized by the sublogic $\mathcal{LM}$ of $\mathcal{A}$ that arises by restricting the formulas to contain only the propositional connectives and the following modalities: $\langle\tau\rangle$, $\langle\bar{x}y\rangle$, $\langle\bar{x}(y)\rangle$, $[x = y]$, $\langle x(y)\rangle^L$, and their duals. We shall now show a similar characterization for open bisimulation.

The following theorem states that by dropping the excluded middle and changing the quantification of free names from ∇ to $\forall$, we get exactly a characterization of open bisimulation by the encoding of the sublogic $\mathcal{LM}$.

THEOREM 27. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes. Then* $\mathtt{P} \sim_o^{\emptyset} \mathtt{Q}$ *if and only if for every* $\mathcal{LM}$*-formula* $\mathtt{A}$*, it holds that* $\vdash \forall\bar{n}([\![\mathtt{P}]\!] \models [\![\mathtt{A}]\!])$ *if and only if* $\vdash \forall\bar{n}([\![\mathtt{Q}]\!] \models [\![\mathtt{A}]\!])$*, where* $\bar{n}$ *is the list of free names in* $\mathtt{P}$*,* $\mathtt{Q}$ *and* $\mathtt{A}$*.*

$$!P \xrightarrow{A} P'|!P \triangleq P \xrightarrow{A} P'$$
$$!P \xrightarrow{X} \lambda y(My|!P) \triangleq P \xrightarrow{X} M$$
$$!P \xrightarrow{\tau} (P' \mid M\ Y)|!P \triangleq \exists X.P \xrightarrow{\uparrow XY} P' \wedge P \xrightarrow{\downarrow X} M$$
$$!P \xrightarrow{\tau} \nu z.(Mz \mid Nz)|!P \triangleq \exists X.P \xrightarrow{\uparrow X} M \wedge P \xrightarrow{\downarrow X} N$$

Fig. 5. Definition clauses for the π-calculus with replication

7. ALLOWING REPLICATION IN PROCESS EXPRESSIONS

We now consider an extension to the finite π-calculus which will allow us to represent non-terminating processes. There are at least two ways to encode non-terminating processes in the π-calculus; *e.g.*, via recursive definitions or replications [Sangiorgi and Walker 2001]. We consider here the latter approach since it leads to a simpler presentation of the operational semantics. To the syntax of finite π-calculus we add the process expression $!P$. The processes $!P$ can be understood as the infinite parallel composition of P, i.e., $P|P|\cdots|P|\cdots$. Thus it is possible to have a process that retains a copy of itself after making a transition; *e.g.*, $!P \xrightarrow{\alpha} P'|!P$. The operational semantics for one-step transitions of the π-calculus with replication is given as the definition clauses Figure 5, adapted to the λ-tree syntax from the original presentation in [Sangiorgi and Walker 2001]. We use the same symbol to encode replication in λ-tree syntax, *i.e.*, $! : p \to p$.

In order to reason about bisimulation of processes involving !, we need to move to a stronger logic which incorporates both induction and co-induction proof rules. We consider the logic Linc [Tiu 2004], which is an extension of $FO\lambda^{\Delta\nabla}$ with induction and co-induction proof rules. We first need to extend the notion of definitions to include inductive and co-inductive definitions.

Definition 28. An inductive definition clause is written

$$\forall \bar{x}.p\bar{x} \stackrel{\mu}{=} B\ p\ \bar{x}$$

where B is a closed term. The symbol $\stackrel{\mu}{=}$ is used to indicate that the definition is inductive. Similarly, a co-inductive definition clause is written

$$\forall \bar{x}.p\bar{x} \stackrel{\nu}{=} B\ p\ \bar{x}.$$

The notion of definition defined in Definition 1 shall be referred to as *regular definition*. An *extended definition* is a collection of regular, inductive, or co-inductive definition clauses.

Notice that the head of the (co-)inductive definition clauses contains a predicate with arguments that are only variables and not more general terms: this restriction simplifies the presentation of the induction and co-induction inference rules. Arguments that are more general terms can be encoded as explicit equalities in the body of the clause. We also adopt a higher-order notation in describing the body of clauses, *i.e.*, we use $B\ p\ \bar{x}$ to mean that B is a top-level abstraction that has no free occurrences of the predicate symbol p and the variables $\bar{x}$. This notation simplifies the presentation of the (co-)induction rules: in particular, it simplifies the presentation of predicate substitutions.

There must be some stratification on the extended definition so as not to introduce inconsistency into the logic. For the details of such stratification we refer the interested readers to [Tiu 2004]. For our current purpose, it should be sufficient to understand that mutual recursive (co-)inductive definitions are not allowed, and negative dependencies through negation is forbidden as it already is in regular definitions.

Let $p\bar{x} \stackrel{\mu}{=} B\ p\ \bar{x}$ be an inductive definition. Its left and right introduction rules are

$$\frac{\bar{x}\,;\, B\ S\ \bar{x} \vdash S\ \bar{x} \quad \Sigma\,;\, \bar{z} \triangleright S\ \bar{t}, \Gamma \vdash \mathcal{C}}{\Sigma\,;\, \bar{z} \triangleright p\ \bar{t}, \Gamma \vdash \mathcal{C}}\ \mu\mathcal{L} \qquad \frac{\Sigma\,;\, \Gamma \vdash \bar{z} \triangleright B\ p\ \bar{t}}{\Sigma\,;\, \Gamma \vdash \bar{z} \triangleright p\ \bar{t}}\ \mu\mathcal{R}$$

where S is the *induction invariant*, and it is a closed term of the same type as p. The introduction rules for co-inductively defined predicates are dual to the inductive ones. In this case, we suppose that p is defined by the co-inductive clause $p\bar{x} \stackrel{\nu}{=} B\ p\ \bar{x}$.

$$\frac{\Sigma\,;\, \bar{z} \triangleright B\ p\ \bar{t}, \Gamma \vdash \mathcal{C}}{\Sigma\,;\, \bar{z} \triangleright p\ \bar{t}, \Gamma \vdash \mathcal{C}}\ \nu\mathcal{L} \qquad \frac{\Sigma\,;\, \Gamma \vdash \bar{z} \triangleright S\ \bar{t} \quad \bar{x}\,;\, S\ \bar{x} \vdash B\ S\ \bar{x}}{\Sigma\,;\, \Gamma \vdash \bar{z} \triangleright p\ \bar{t}}\ \nu\mathcal{R}$$

Here S is a closed term denoting the co-induction invariant or *simulation*. Induction rules cannot be applied to co-inductive predicates and vice versa. The *def*R and *def*$\mathcal{L}$ rules, strictly speaking, are applicable only to regular definitions. But as it is shown in [Tiu 2004], these rules are derivable for (co-)inductive definitions: that is, for these definitions, *def*R can be shown to be a special case of $\nu\mathcal{R}$ and *def*$\mathcal{L}$ a special case of $\mu\mathcal{L}$.

The definitions in $FO\lambda^{\Delta\nabla}$ we have seen so far can be carried over to Linc with some minor bureaucratic changes: *e.g.*, in the case of bisimulations, we now need to indicate explicitly that it is a co-inductive definition. For instance, the definition of lbisim should now be indicated as a co-inductive definition by changing the symbol $\stackrel{\triangle}{=}$ with $\stackrel{\nu}{=}$. We shall now present an example of proving bisimulation using explicit induction and co-induction rules. We shall not go into details of the technical theorems of the adequacy results and we refer to [Tiu 2004] for further details.

Example 29. Let $\mathtt{P} = !(z)(\bar{z}a \mid z(y).\bar{x}y)$ and $\mathtt{Q} = !\tau.\bar{x}a$. The only action P can make is the silent action τ since the channel z is restricted internally within the process. It is easy to see that $\mathtt{P} \stackrel{\tau}{\longrightarrow} \bar{x}a \mid \mathtt{P}$. That is, the continuation of P is capable of outputting a free name a or making a silent transition. Obviously Q can make the same τ action and results in a bisimilar continuation. Let us try to prove *lbisim* P Q. The simple proof strategy of unfolding the lbisim clause via *def*R will not work here since after the first *def*R on lbisim (but before the second *def*R on lbisim) we arrive at the sequent *lbisim* $((z)(0 \mid \bar{x}a) \mid \mathtt{P})\ (\bar{x}a \mid \mathtt{Q})$. Since P and Q still occur in the continuation pair, it is obvious that this strategy is non terminating. We need to use the co-induction proof rules instead.

An informal proof starts by finding a bisimulation (a set of pairs of processes) $\mathcal{S}$ such that $(\mathtt{P}, \mathtt{Q}) \in \mathcal{S}$. Let $\mathtt{R}_i$, for any natural number i, be either $(z)(0 \mid \bar{x}a)$ or $(z)(0 \mid 0)$, and $\mathtt{T}_i$ be either $\bar{x}a$ or 0 and let $\mathcal{S}' = \{(\mathtt{R}_1 \mid \cdots \mid \mathtt{R}_n \mid \mathtt{P}, \mathtt{T}_1 \mid \cdots \mid \mathtt{T}_n \mid \mathtt{Q}) \mid n \geq 0\}$. Define $\mathcal{S}$ to be the symmetric closure of $\mathcal{S}'$. It can be verified that $\mathcal{S}$ is a bisimulation set by showing the set is closed with respect to one-step transitions. To prove this

formally in Linc we need to represent the set $\mathcal{S}$. We code the set $\mathcal{S}$ as the following inductive definition (we allow ourselves to put general terms in the head of this definition and to have more than one clause: it is straightforward to translate this definition to the restricted one give above).

$$\begin{aligned}
\mathit{inv}\ \mathtt{P}\ \mathtt{Q} &\stackrel{\mu}{=} \top. \quad \mathit{inv}\ \mathtt{Q}\ \mathtt{P} \stackrel{\mu}{=} \top.\\
\mathit{inv}\ ((z)(0 \mid 0) \mid M)\ (0 \mid N) &\stackrel{\mu}{=} \mathit{inv}\ M\ N.\\
\mathit{inv}\ (0 \mid N)\ ((z)(0 \mid 0) \mid M) &\stackrel{\mu}{=} \mathit{inv}\ N\ M.\\
\mathit{inv}\ ((z)(0 \mid \bar{x}a) \mid M)\ (\bar{x}a \mid N) &\stackrel{\mu}{=} \mathit{inv}\ M\ N.\\
\mathit{inv}\ (\bar{x}a \mid N)\ ((z)(0 \mid \bar{x}a) \mid M) &\stackrel{\mu}{=} \mathit{inv}\ N\ M.
\end{aligned}$$

The set of pairs encoded by *inv* can be shown to be symmetric, i.e., the formula $\forall R \forall T.\mathit{inv}\ R\ T \supset \mathit{inv}\ T\ R$ is provable inductively (using the same formula as the induction invariant).

To now prove the sequent $\vdash \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$, we can use the $\nu\mathcal{R}$ rule with the predicate *inv* as the invariant. The premises of the $\nu\mathcal{R}$ rule are the two sequents $\vdash \mathit{inv}\ \mathtt{P}\ \mathtt{Q}$ and $R, T\,;\ \mathit{inv}\ R\ T \vdash B\,R\,T$, where $B\,R\,T$ is the following large conjunction

$$\begin{aligned}
&\forall A \forall P'\ [(R \xrightarrow{A} R') \supset \exists Q'.(T \xrightarrow{A} T') \wedge \mathit{inv}\ R'\ T'] \wedge\\
&\forall A \forall T'\ [(T \xrightarrow{A} T') \supset \exists R'.(R \xrightarrow{A} R') \wedge \mathit{inv}\ T'\ R'] \wedge\\
&\forall X \forall R'\ [(P \xrightarrow{\downarrow X} R') \supset \exists T'.(T \xrightarrow{\downarrow X} T') \wedge \forall w.\mathit{inv}\ (R'w)\ (T'w)] \wedge\\
&\forall X \forall T'\ [(T \xrightarrow{\downarrow X} T') \supset \exists R'.(R \xrightarrow{\downarrow X} R') \wedge \forall w.\mathit{inv}\ (R'w)\ (R'w)] \wedge\\
&\forall X \forall R'\ [(R \xrightarrow{\uparrow X} R') \supset \exists T'.(T \xrightarrow{\uparrow X} T') \wedge \nabla w.\mathit{inv}\ (R'w)\ (T'w)] \wedge\\
&\forall X \forall T'\ [(T \xrightarrow{\uparrow X} T') \supset \exists R'.(R \xrightarrow{\uparrow X} R') \wedge \nabla w.\mathit{inv}\ (T'w)\ (R'w)].
\end{aligned}$$

The sequent reads, intuitively, that the set defined by *inv* is symmetric and is closed under one-step transitions. This is proved by induction on *inv*. Formally, this is done by applying $\mu\mathcal{L}$ to $\mathit{inv}\ R\ T$, using the invariant

$$\lambda R \lambda T.\mathit{inv}\ R\ T \supset B\,R\,T.$$

The sequents corresponding to the base cases of the induction are

$$\mathit{inv}\ \mathtt{P}\ \mathtt{Q} \vdash B\,\mathtt{P}\,\mathtt{Q} \quad \text{and} \quad \mathit{inv}\ \mathtt{Q}\ \mathtt{P} \vdash B\,\mathtt{Q}\,\mathtt{P}$$

and the inductive cases are given by

$$\begin{aligned}
&\mathit{inv}\ R\ T \supset B\,R\,T \vdash \mathit{inv}\ ((z)(0 \mid 0) \mid R)\ (0 \mid T) \supset B((z)(0 \mid 0) \mid R)(0 \mid T),\\
&\mathit{inv}\ R\ T \supset B\,R\,T \vdash \mathit{inv}\ ((z)(0 \mid \bar{x}a) \mid R)\ (\bar{x}a \mid T) \supset B((z)(0 \mid \bar{x}a) \mid R)(\bar{x}a \mid T)
\end{aligned}$$

and their symmetric variants. The full proof involves a number of cases of which we show one here: the other cases can be proved similarly.

We consider a case for free output, where we have the sequent (after applying some right-introduction rules)

$$\left\{\begin{array}{r} \mathit{inv}\ R\ T \supset B\,R\,T \\ \mathit{inv}\ ((z)(0 \mid \bar{x}a) \mid R)\ (\bar{x}a \mid T) \\ ((z)(0 \mid \bar{x}a) \mid R) \xrightarrow{A} R' \end{array}\right\} \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge \mathit{inv}\ R'\ T' \qquad (1)$$

$$\dfrac{\dfrac{\dfrac{}{\cdots \vdash \top}\,\top\mathcal{R}}{\cdots \vdash (\bar{x}a \mid T) \xrightarrow{\bar{x}a} (0 \mid T)}\,def\,R \qquad \dfrac{\dfrac{}{\cdots, inv\ R\ T \vdash inv\ R\ T}\,init}{\cdots, inv\ R\ T \vdash inv\ ((z)(0 \mid 0) \mid R)\ (0 \mid T)}\,def\,R}{B\,R\,T, inv\ R\ T \vdash (\bar{x}a \mid T) \xrightarrow{\bar{x}a} (0 \mid T) \wedge inv\ ((z)(0 \mid 0) \mid R)\ (0 \mid T)}\,\wedge\mathcal{R}$$
$$\dfrac{B\,R\,T, inv\ R\ T \vdash (\bar{x}a \mid T) \xrightarrow{\bar{x}a} (0 \mid T) \wedge inv\ ((z)(0 \mid 0) \mid R)\ (0 \mid T)}{B\,R\,T, inv\ R\ T \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{\bar{x}a} T' \wedge inv\ ((z)(0 \mid 0) \mid R)\ T'}\,\exists\mathcal{R}$$

Fig. 6. A derivation in Linc

$$\dfrac{\dfrac{\dfrac{}{R \xrightarrow{A} R'' \vdash R \xrightarrow{A} R''}\,init \qquad \Pi}{R \xrightarrow{A} R'' \supset \exists V.T \xrightarrow{A} V \wedge inv\ R''\ V \vdash \cdots}\,\supset\mathcal{L}}{\dfrac{R \xrightarrow{A} R'' \supset \exists V.T \xrightarrow{A'} V \wedge inv\ R''\ V, R \xrightarrow{A} R'' \vdash \cdots}{\forall U \forall A'\ R \xrightarrow{A} U \supset \exists V.T \xrightarrow{A'} V \wedge inv\ U\ V, R \xrightarrow{A} R'' \vdash \cdots}\,\wedge\mathcal{L}}\,\forall\mathcal{L};\forall\mathcal{L}$$
$$B\,R\,T, R \xrightarrow{A} R'' \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge inv\ (z)(0 \mid \bar{x}a) \mid R'')\ T'$$

where Π is

$$\dfrac{\dfrac{\dfrac{}{T \xrightarrow{A} V \vdash T \xrightarrow{A} V}\,init}{T \xrightarrow{A} V \vdash (\bar{x}a \mid T) \xrightarrow{A} (\bar{x}a \mid V)}\,def\,R \qquad \dfrac{\dfrac{}{inv\ R''\ V \vdash inv\ R''\ V}\,init}{inv\ R''\ V \vdash inv\ ((z)(0 \mid \bar{x}a) \mid R'')\ (\bar{x}a \mid V)}\,def\,R}{T \xrightarrow{A} V, inv\ R''\ V \vdash (\bar{x}a \mid T) \xrightarrow{A} (\bar{x}a \mid V) \wedge inv\ (z)(0 \mid \bar{x}a) \mid R'')\ (\bar{x}a \mid V)}\,\wedge\mathcal{R}$$
$$\dfrac{T \xrightarrow{A} V, inv\ R''\ V \vdash (\bar{x}a \mid T) \xrightarrow{A} (\bar{x}a \mid V) \wedge inv\ (z)(0 \mid \bar{x}a) \mid R'')\ (\bar{x}a \mid V)}{T \xrightarrow{A} V, inv\ R''\ V \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge inv\ (z)(0 \mid \bar{x}a) \mid R'')\ T'}\,\exists\mathcal{R}$$
$$\dfrac{T \xrightarrow{A} V, inv\ R''\ V \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge inv\ (z)(0 \mid \bar{x}a) \mid R'')\ T'}{\exists V.T \xrightarrow{A} V \wedge inv\ R''\ V \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge inv\ (z)(0 \mid \bar{x}a) \mid R'')\ T'}\,\exists\mathcal{L};\wedge\mathcal{L}$$

Fig. 7. A derivation in Linc given in two parts

to prove. Its symmetric case can be proved by using cut, since the predicate *inv* is symmetric. The sequent (1) can be simplified by applying *def*$\mathcal{L}$ to the *inv* predicate, followed by an instance of $\supset\mathcal{L}$. The resulting sequent is

$$\left\{\begin{array}{r} B\,R\,T,\ inv\ R\ T \\ ((z)(0 \mid \bar{x}a) \mid R) \xrightarrow{A} R' \end{array}\right\} \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge inv\ R'\ T' \qquad (2)$$

There are three ways in which the one-step transition in the left-hand side of the sequent (1) can be inferred (via *def*$\mathcal{L}$), *i.e.*, either A is $\bar{x}a$ and R' is $((z)(0 \mid 0) \mid R)$, or $R \xrightarrow{A} R''$ and R' is $(z)(0 \mid \bar{x}a) \mid R'')$, or A is τ and $R \xrightarrow{\downarrow X} M$, R' is $((z)(0 \mid 0)|Ma)$ for some X and M. These three cases correspond to the following sequents.

$$B\,R\,T, inv\ R\ T \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{\bar{x}a} T' \wedge inv\ ((z)(0 \mid 0) \mid R)\ T'$$
$$B\,R\,T, inv\ R\ T, R \xrightarrow{A} R'' \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{A} T' \wedge inv\ (z)(0 \mid \bar{x}a) \mid R'')\ T'$$
$$B\,R\,T, inv\ R\ T, R \xrightarrow{\downarrow X} M \vdash \exists T'.(\bar{x}a \mid T) \xrightarrow{\tau} T' \wedge inv\ ((z)(0 \mid 0)|Ma)\ T'$$

The proof of the first sequent is given in Figure 6 and of the second sequent is given in Figure 7. The proof for the third sequent is not given but it is easy to see that it has similar structure as the proof for the second one.

8. AUTOMATION OF PROOF SEARCH

The above specifications for one-step transitions and for late and open bisimulation are not only declarative and natural, an implementation of proof search using them can provide effective and *symbolic* implementation of both one-step transitions and bisimulations. In this section we outline general approaches to the automation of proof search in $FO\lambda^{\Delta\nabla}$.

8.1 Focused proof search

Since the cut-elimination theorem holds for $FO\lambda^{\Delta\nabla}$, the search for a proof can be restricted to *cut-free proofs*. It is possible to significantly constrain cut-free proofs to *focused proofs* while still preserving completeness. The search for focused proofs has a simple structure, which is structured into two phases. One phase is the *asynchronous* phase and it involves the application of invertible inference rules: in this phase, such inference rules are applied in any order and until no additional invertible rule can be applied. The second phase, called the *synchronous* phase, involves the selection of a non-invertible inference rule and the hereditary (focused) application of non-invertible rules until invertible rules are possible again. Andreoli [Andreoli 1992] provided such a focused proof system for linear logic and proved its completeness. Eventually, other focusing systems for intuitionistic and classical logic have been developed, *c.f.* [Liang and Miller 2007]. Focused proof systems are generally the basis for the automation of logic programming languages and they generalize *uniform proofs* [Miller et al. 1991]. A focusing result for a useful subset of the $FO\lambda^{\Delta\nabla}$ logic has been developed in [Baelde and Miller 2007] and an implementation of that subset has been written in the Bedwyr system [Baelde et al. 2007].

8.2 Unification

Unification can be used in the implementation of $FO\lambda^{\Delta\nabla}$ proof search in two different ways. One way involves the implementation of the $def\mathcal{L}$ inference rule and the other way involves the determination of appropriate terms for instantiating the $\exists$ quantifier in the $\exists\mathcal{R}$ inference rule and the $\forall$ quantifier in the $\forall\mathcal{L}$ inference rule. In the specifications presented here, unification only requires the decidable and determinate subset of higher-order unification called *higher-order pattern* (or L_λ) unification [Miller 1991]. This style of unification, which can be described as first-order unification extended to allow for bound variables and their mobility within terms and proofs, is known to have efficient and practical unification algorithms that compute most general unifiers whenever unifiers exist [Nipkow 1993; Nadathur and Linnell 2005]. The Teyjus implementation [Nadathur and Mitchell 1999; Nadathur 2005] of λProlog provides an effective implementation of such unification, as does Isabelle [Paulson 1990] and Twelf [Pfenning and Schürmann 1999].

8.3 Proof search for one-step transitions.

Computing one-step transitions can be done entirely using a conventional, higher-order logic programming language, such as λProlog: since the definition $\mathbf{D}_\pi$ for one-step transitions is Horn, we can use Proposition 4 to show that for the purposes of computing one-step transitions, all occurrences of ∇ in $\mathbf{D}_\pi$ can be changed

to $\forall$. The resulting definition is then a λProlog logic program for which Teyjus provides an effective implementation. In particular, after loading that definition, we would simply ask the query $P \xrightarrow{A} P'$, where P is the encoding of a particular π-calculus expression and A and P' are free variables. Standard logic programming interpreters would then systematically bind these two variables to the actions and continuations that P can make. Similarly, if the query was $P \xrightarrow{A}\!\!\!\!\rightarrow P'$, logic programming search would systematically return all bound actions (here, A has type $n \to \mathrm{a}$) and corresponding bound continuations (here, P' has type $n \to p$).

8.4 Proof search for open bisimulation.

Theorem proving establishing a bisimulation goal is not done via a conventional logic programming system like λProlog since such systems do not implement the ∇-quantifier and the case analysis and unification of eigenvariables that is required for the *def*$\mathcal{L}$ inference rule. None-the-less, the implementation of proof search for open bisimulation is easy to specify using the following key steps. (Sequents missing from this outline are trivial to address.) In the following, we use the quantifier prefix $\mathcal{Q}$ to denote either $\forall x$ or ∇x or the empty quantifier prefix.

(1) When searching for a proof of $\Sigma\,;\vdash \sigma \triangleright \mathcal{Q}.\mathit{lbisim}\ P\ Q$ apply right-introduction rules: *i.e.*, simply introduce the quantifier $\mathcal{Q}$ (if it is non-empty) and then open the definition of *lbisim*.

(2) If the sequent has a formula on its left-hand sides, then that formula is $\sigma \triangleright P \xrightarrow{A} P'$, where P denotes a particular term where all its non-ground subterms are of type n, and A and P' are terms, possibly containing eigenvariables. In this case, select the *def*$\mathcal{L}$ inference rule: the premises of this inference rule will then be either (i) the empty-set of premises (which represents the only way that proof search terminates), or (ii) a set of premises that are all again of the form of one-step judgments, or (iii) the premise contains $\top$ instead of an atom on the left, in which case, we must consider the remaining case that follows (after using the weakening $w\mathcal{L}$ inference rule).

(3) If the sequent has the form $\Sigma\,;\vdash \sigma \triangleright \exists Q'[Q \xrightarrow{A} Q' \wedge B(P', Q')]$, where $B(P', Q')$ involves a recursive call to *lbisim* and where P' is a closed term, then we must instantiate the existential quantifier with an appropriate substitution. Standard logic programming techniques can be used to find a substitution for Q' such that $Q \xrightarrow{A} Q'$ is provable (during this search, eigenvariables and locally scoped variables are treated as constants and P and A denote particular closed terms). There might be several ways to prove such a formula and, as a result, there might be several different substitutions for Q'. If one chooses the term T to instantiate Q', then one proceeds to prove the sequent $\Sigma\,;\vdash \sigma \triangleright \mathcal{Q}.\mathit{lbisim}\ P'\ T$.

If the sequent has instead the form $\Sigma\,;\vdash \sigma \triangleright \exists Q'[Q \xrightarrow{A}\!\!\!\!\rightarrow Q' \wedge B(P', Q')]$, then one proceeds in an analogous manner.

Proof search for the first two cases is invertible (no backtracking is needed for those cases). On the other hand, the third case is not invertible and backtracking on possibly all choices of substitution term T might be necessary to ensure completeness.

8.5 Proof search for late bisimulation.

The main difference between doing proof search for open bisimulation and late bisimulation is that in the latter we need to select and instantiate formulas from the set $\mathcal{E}x$ and explore the cases generated by the resulting $\vee\mathcal{L}$ rule. For example, consider a sequent of the form $\Sigma, x\,;\, \mathcal{E}x, \Gamma_x \vdash C_x$, where $\Gamma_x \cup \{C_x\}$ is a set of formulas which may have x free. One way to proceed with the search for a proof would be to instantiate $\forall z(x = z \vee x \neq z)$ twice with the constants a and b. We would then need to consider proofs of the sequent $\Sigma, x\,;\, x = a \vee x \neq a, x = b \vee x \neq b, \Gamma_x \vdash C_x$. Using the $\vee\mathcal{L}$ rule twice, we are left with four sequents to prove:

(1) $\Sigma, x\,;\, x = a, x = b, \Gamma_x \vdash C_x$ which is proved trivially since the equalities are contradictory;
(2) $\Sigma, x\,;\, x = a, x \neq b, \Gamma_x \vdash C_x$, which is equivalent to $\Sigma\,;\, \Gamma_a \vdash C_a$;
(3) $\Sigma, x\,;\, x \neq a, x = b, \Gamma_x \vdash C_x$, which is equivalent to $\Sigma\,;\, \Gamma_b \vdash C_b$; and
(4) $\Sigma, x\,;\, x \neq a, x \neq b, \Gamma_x \vdash C_x$.

In this way, the excluded middle can be used with a set of n items to produce $n+1$ sequents: one for each member of the set and one extra sequent to handle all other cases (if there are any).

The main issue for implementing proof search with this specification of late bisimulation is to determine what instances of the excluded middle are needed: answering this question would then reduce proof search to one similar to open bisimulation. There seems to be two extreme approaches to take. At one extreme, we can take instances for all possible names that are present in our process expressions: determining such instances is simple but might lead to many more cases to consider than is necessary. The other extreme would be more lazy: an instance of the excluded middle is suggested only when there seems to be a need to consider that instance. The failure of a $defR$ rule because of a mismatch between an eigenvariable and a constant would, for example, suggest that excluded middle should be invoked for that eigenvariable and that constant. The exact details of such schemes is left for future work.

9. RELATED AND FUTURE WORK

There are many papers on topics related to the encoding of the operational semantics of the π-calculus into formal systems. An encoding of one-step transitions for the π-calculus using Coq was presented in [Despeyroux 2000] but the problem of computing bisimulation was not considered. Honsell, Miculan, and Scagnetto [Honsell et al. 2001] give a more involved encoding of the π-calculus in Coq and assume that there are an infinite number of global names. They then build formal mechanisms to support notions such as "freshness" within a scope, substitution of names, occurrences of names in expressions, etc. Gabbay [Gabbay 2003] does something similar but uses the set theory developed in [Gabbay and Pitts 2001] to help develop his formal mechanisms. This formalism is later given a first-order axiomatizations by Pitts [Pitts 2003], resulting in an extension of first-order logic called *nominal logic*. Aspects of nominal reasoning has been incorporated into the proof assistant Isabelle [Urban and Tasson 2005] and there has been some recent work in formalizing the meta theory of the π-calculus in this framework [Bengtson and Parrow

2007]. Hirschkoff [Hirschkoff 1997] also used Coq but employed deBruijn numbers [Bruijn 1972] instead of explicit names. In the papers that address bisimulation, formalizing names and their scopes, occurrences, freshness, and substitution is considerable work. In our approach, much of this same work is required, of course, but it is available in rather old technology, particularly, via Church's Simple Theory of Types (where bindings in terms and formulas were put on a firm foundation via λ-terms), Gentzen's sequent calculus, Huet's unification procedure for λ-terms [Huet 1975], etc. More modern work on proof search in higher-order logics is also available to make our task easier and more declarative.

The encoding of transitions for the π-calculus into logics and type systems have been studied in a number of previous works [Briais and Nestmann 2007; Honsell et al. 1998; Despeyroux 2000; Honsell et al. 2001; Röckl et al. 2001]. Our encoding, presented as a definition in Figure 2, has appeared in [Miller and Palamidessi 1999; Miller and Tiu 2003]. The material on proof automation in Section 8 clearly seems related to *symbolic bisimulation* (for example, see [Boreale and Nicola 1996; Hennessy and Lin 1995]) and on using unification and logic programming techniques to compute symbolic bisimulations (for example, see [Basu et al. 2001; Boreale 2001]). Since the technologies used to describe these other approaches are rather different than what is described here, a detailed comparison is left for future work.

It is, of course, interesting to consider the general π-calculus where infinite behaviors are allowed (by including ! or recursive definitions). In such cases, one might be able to still do many proofs involving bisimulation if the proof system included induction and co-induction inference rules. Inference rules for induction and co-induction appropriate for the sequent calculus have been presented in [Momigliano and Tiu 2003] and a version of these rules that also involves the ∇ quantifier has been presented in the first author's PhD [Tiu 2004]. Open bisimulation, however, has not been studied in this setting. We plan to investigate further how these stronger proof systems can be used establish properties about π-calculus expressions with infinite behaviors.

Specifications of operational semantics using a logic should make it possible to formally prove properties concerning that operational semantics. This was the case, for example, with specifications of the evaluation and typing of simple functional and imperative programming languages: a number of common theorems (determinacy of evaluation, subject-reduction, etc) can be naturally inferred using logical specifications [McDowell and Miller 2002]. We plan to investigate using our logic (also incorporating rules for induction and co-induction) for formally proving parts of the theory of the π-calculus. It seems, for example, rather transparent to prove that open bisimilarity is a congruence in our setting (see [Ziegler et al. 2005] for a more general class of congruence relations).

10. CONCLUSION

In this paper we presented a meta-logic that allows for declarative specifications of judgments related to the π-calculus. These specifications are done entirely within the logic and without any additional side conditions. The management of name bindings in the specification of one-step transition, bisimulation, and modal logic is handled completely by the logic's three levels of binding, namely, λ-bindings

within terms, the formula-level binders (quantifiers) $\forall$, $\exists$, and ∇, and the proof-level bindings for eigenvariables and local (generic) contexts.

A significant part of this paper deals with establishing adequacy results that show a formal connection between the "standard" definitions of judgments concerning the π-calculus and the definitions given in logic (see the appendices for the details). These adequacy results are all rather tedious and shallow but seem necessary to ensure that we have not invented our own problems for which we provide good solutions. It would seem, however, that the tediousness of adequacy can be attributed to the "standard" approach used to represent the π-calculus: now that some of these basic adequacy results have been written down, it might be possible to switch entirely from the standard approach to the approach using λ-tree syntax described in this paper. Historically, a similar switch in representations has taken place. For example, Church and Gödel formalized terms and formulas as strings: eventually, this standard treatment of syntax was replaced by more abstract objects such as parse trees. It is on parse trees that most standard syntactic descriptions of the λ-calculus and π-calculus are given. Unfortunately, parse trees do not come equipped with primitive notions of bindings. To fix that problem, for example, Prawitz introduced "discharge functions" [Prawitz 1965] and de Bruijn introduced "nameless dummies" [Bruijn 1972]. We have illustrated here that by making syntax more abstract (moving from parse-trees to λ-trees) and by using a logic able to deal intimately with syntactic abstractions, specifications can become fully declarative and avoid the morass of dealing with too many concrete aspects of the encoding of bindings.

Acknowledgments. We are grateful to the reviewers of an earlier draft of this paper for their detailed and useful comments. We also benefited from support from INRIA through the "Equipes Associées" Slimmer, and from the Australian Research Council through the Discovery Project "Proof Theoretical Methods for Reasoning about Process Equivalence".

REFERENCES

Andreoli, J.-M. 1992. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation 2,* 3, 297–347.

Baelde, D., Gacek, A., Miller, D., Nadathur, G., and Tiu, A. 2007. The Bedwyr system for model checking over syntactic expressions. In *21th Conference on Automated Deduction (CADE)*, F. Pfenning, Ed. Number 4603 in LNAI. Springer, 391–397.

Baelde, D. and Miller, D. 2007. Least and greatest fixed points in linear logic. In *International Conference on Logic for Programming and Automated Reasoning (LPAR)*, N. Dershowitz and A. Voronkov, Eds. Lecture Notes in Computer Science, vol. 4790. 92–106.

Basu, S., Mukund, M., Ramakrishnan, C. R., Ramakrishnan, I. V., and Verma, R. M. 2001. Local and symbolic bisimulation using tabled constraint logic programming. In *International Conference on Logic Programming (ICLP)*. LNCS, vol. 2237. Springer, Paphos, Cyprus, 166–180.

Bengtson, J. and Parrow, J. 2007. Formalising the π-calculus using nominal logic. In *Proceedings of FOSSACS 2007*. LNCS, vol. 4423. Springer, 63–77.

Boreale, M. 2001. Symbolic trace analysis of cryptographic protocols. In *Proceedings of ICALP 2001*. LNCS, vol. 2076. Springer-Verlag, 667–681.

Boreale, M. and Nicola, R. D. 1996. A symbolic semantics for the π-calculus. *Information and Computation 126,* 1 (Apr.), 34–52.

Briais, S. and Nestmann, U. 2007. Open bisimulation, revisited. *Theoretical Computer Science 386,* 3, 236–271.

Bruijn, N. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser Theorem. *Indag. Math. 34,* 5, 381–392.

Church, A. 1940. A formulation of the simple theory of types. *J. of Symbolic Logic 5*, 56–68.

Despeyroux, J. 2000. A higher-order specification of the π-calculus. In *Proc. of the IFIP International Conference on Theoretical Computer Science, IFIP TCS'2000, Sendai, Japan, August 17-19, 2000.* 425–439.

Eriksson, L.-H. 1991. A finitary version of the calculus of partial inductive definitions. In *Proceedings of the Second International Workshop on Extensions to Logic Programming*, L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, Eds. LNAI, vol. 596. Springer-Verlag, 89–134.

Fiore, M. P., Plotkin, G. D., and Turi, D. 1999. Abstract syntax and variable binding. In *14th Symp. on Logic in Computer Science*. IEEE Computer Society Press, 193–202.

Gabbay, M. J. 2003. The π-calculus in FM. In *Thirty-five years of Automath*, F. Kamareddine, Ed. Kluwer, 80–149.

Gabbay, M. J. and Pitts, A. M. 2001. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing 13*, 341–363.

Gentzen, G. 1969. Investigations into logical deductions. In *The Collected Papers of Gerhard Gentzen*, M. E. Szabo, Ed. North-Holland, Amsterdam, 68–131.

Girard, J.-Y. 1992. A fixpoint theorem in linear logic. An email posting to the mailing list linear@cs.stanford.edu.

Hallnäs, L. and Schroeder-Heister, P. 1991. A proof-theoretic approach to logic programming. II. Programs as definitions. *J. of Logic and Computation 1,* 5 (Oct.), 635–660.

Hennessy, M. and Lin, H. 1995. Symbolic bisimulations. *Theoretical Computer Science 138,* 2 (Feb.), 353–389.

Hirschkoff, D. 1997. A full formalization of pi-calculus theory in the Calculus of Constructions. In *Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'97)*, E. Gunter and A. Felty, Eds. Number 1275 in LNCS. Murray Hill, New Jersey, 153–169.

Hofmann, M. 1999. Semantical analysis of higher-order abstract syntax. In *14th Symp. on Logic in Computer Science*. IEEE Computer Society Press, 204–213.

Honsell, F., Lenisa, M., Montanari, U., and Pistore, M. 1998. Final semantics for the π-calculus. In *Proc. of PROCOMET'98*.

Honsell, F., Miculan, M., and Scagnetto, I. 2001. π-calculus in (co)inductive type theories. *Theoretical Computer Science 2,* 253, 239–285.

Huet, G. 1975. A unification algorithm for typed λ-calculus. *Theoretical Computer Science 1*, 27–57.

Huet, G. and Lang, B. 1978. Proving and applying program transformations expressed with second-order patterns. *Acta Informatica 11*, 31–55.

Liang, C. and Miller, D. 2007. Focusing and polarization in intuitionistic logic. In *CSL 2007: Computer Science Logic*, J. Duparc and T. A. Henzinger, Eds. LNCS, vol. 4646. Springer, 451–465.

McDowell, R. and Miller, D. 2000. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science 232*, 91–119.

McDowell, R. and Miller, D. 2002. Reasoning with higher-order abstract syntax in a logical framework. *ACM Trans. on Computational Logic 3,* 1, 80–136.

McDowell, R., Miller, D., and Palamidessi, C. 2003. Encoding transition systems in sequent calculus. *Theoretical Computer Science 294,* 3, 411–437.

Miller, D. 1991. A logic programming language with lambda-abstraction, function variables, and simple unification. *J. of Logic and Computation 1,* 4, 497–536.

Miller, D. 1992. Unification under a mixed prefix. *Journal of Symbolic Computation 14,* 4, 321–358.

Miller, D. 2000. Abstract syntax for variable binders: An overview. In *Computational Logic - CL 2000*, J. Lloyd and et. al., Eds. Number 1861 in LNAI. Springer, 239–253.

Miller, D. and Nadathur, G. 1986. Some uses of higher-order logic in computational linguistics. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, New Jersey, 247–255.

Miller, D. and Nadathur, G. 1987. A logic programming approach to manipulating formulas and programs. In *IEEE Symposium on Logic Programming*, S. Haridi, Ed. San Francisco, 379–388.

Miller, D., Nadathur, G., Pfenning, F., and Scedrov, A. 1991. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic 51*, 125–157.

Miller, D. and Palamidessi, C. 1999. Foundational aspects of syntax. *ACM Computing Surveys 31*.

Miller, D. and Tiu, A. 2003. A proof theory for generic judgments: An extended abstract. In *18th Symp. on Logic in Computer Science*. IEEE, 118–127.

Miller, D. and Tiu, A. 2005. A proof theory for generic judgments. *ACM Trans. on Computational Logic 6,* 4 (Oct.), 749–783.

Milner, R., Parrow, J., and Walker, D. 1992. A calculus of mobile processes, Part II. *Information and Computation*, 41–77.

Milner, R., Parrow, J., and Walker, D. 1993. Modal logics for mobile processes. *Theoretical Computer Science 114,* 1, 149–171.

Mitchell, J. C. and Moggi, E. 1991. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic 51,* 1-2, 99–124.

Momigliano, A. and Tiu, A. 2003. Induction and co-induction in sequent calculus. In *Post-proceedings of TYPES 2003*, M. C. S. Berardi and F. Damiani, Eds. Number 3085 in LNCS. 293–308.

Nadathur, G. 2005. A treatment of higher-order features in logic programming. *Theory and Practice of Logic Programming 5,* 3, 305–354.

Nadathur, G. and Linnell, N. 2005. Practical higher-order pattern unification with on-the-fly raising. In *ICLP 2005: 21st International Logic Programming Conference*. LNCS, vol. 3668. Springer, Sitges, Spain, 371–386.

Nadathur, G. and Miller, D. 1988. An Overview of λProlog. In *Fifth International Logic Programming Conference*. MIT Press, Seattle, 810–827.

Nadathur, G. and Mitchell, D. J. 1999. System description: Teyjus—A compiler and abstract machine based implementation of λProlog. In *Proceedings of the 16th International Conference on Automated Deduction*, H. Ganzinger, Ed. Springer-Verlag LNCS, Trento, Italy, 287–291.

Nipkow, T. 1993. Functional unification of higher-order patterns. In *Proc. 8th IEEE Symposium on Logic in Computer Science (LICS 1993)*, M. Vardi, Ed. IEEE, 64–74.

Paulson, L. C. 1990. Isabelle: The next 700 theorem provers. In *Logic and Computer Science*, P. Odifreddi, Ed. Academic Press, 361–386.

Pfenning, F. and Elliott, C. 1988. Higher-order abstract syntax. In *Proceedings of the ACM-SIGPLAN Conference on Programming Language Design and Implementation*. ACM Press, 199–208.

Pfenning, F. and Schürmann, C. 1999. System description: Twelf — A meta-logical framework for deductive systems. In *16th Conference on Automated Deduction (CADE)*, H. Ganzinger, Ed. Number 1632 in LNAI. Springer, Trento, 202–206.

Pitts, A. M. 2003. Nominal logic, A first order theory of names and binding. *Information and Computation 186,* 2, 165–193.

Plotkin, G. 1981. A structural approach to operational semantics. DAIMI FN-19, Aarhus University, Aarhus, Denmark. Sept.

Prawitz, D. 1965. *Natural Deduction*. Almqvist & Wiksell, Uppsala.

Röckl, C., Hirschkoff, D., and Berghofer, S. 2001. Higher-order abstract syntax with induction in Isabelle/HOL: Formalizing the pi-calculus and mechanizing the theory of contexts. In *Proc. FOSSACS'01*, F. Honsell and M. Miculan, Eds. LNCS, vol. 2030. Springer, 364–378.

Sangiorgi, D. 1996. A theory of bisimulation for the π-calculus. *Acta Informatica 33,* 1, 69–97.

Sangiorgi, D. and Walker, D. 2001. *π-Calculus: A Theory of Mobile Processes*. Cambridge University Press.

Schroeder-Heister, P. 1993. Rules of definitional reflection. In *Eighth Annual Symposium on Logic in Computer Science*, M. Vardi, Ed. IEEE Computer Society Press, IEEE, 222–232.

Stärk, R. F. 1994. Cut-property and negation as failure. *International Journal of Foundations of Computer Science 5,* 2, 129–164.

Tiu, A. 2004. A logical framework for reasoning about logical specifications. Ph.D. thesis, Pennsylvania State University.

Tiu, A. 2005. Model checking for π-calculus using proof search. In *CONCUR*, M. Abadi and L. de Alfaro, Eds. LNCS, vol. 3653. Springer, 36–50.

Tiu, A. and Miller, D. 2004. A proof search specification of the π-calculus. In *3rd Workshop on the Foundations of Global Ubiquitous Computing*. ENTCS, vol. 138. 79–101.

Urban, C. and Tasson, C. 2005. Nominal techniques in Isabelle/HOL. In *20th Conference on Automated Deduction (CADE)*, R. Nieuwenhuis, Ed. LNCS, vol. 3632. Springer, 38–53.

Ziegler, A., Miller, D., and Palamidessi, C. 2005. A congruence format for name-passing calculi. In *Proceedings of SOS 2005: Structural Operational Semantics*. Electronic Notes in Theoretical Computer Science. Elsevier Science B.V., Lisbon, Portugal, 169–189.

A. PROPERTIES OF ONE-STEP TRANSITIONS

To prove the adequacy results for the encodings of bisimulation and modal logics, we shall consider some derived rules which allow us to enumerate all possible next states from a given process. In the following, we use the notation $\alpha^n \to \beta$ to denote the type $\underbrace{\alpha \to \cdots \to \alpha}_{n} \to \beta$, and we write $\alpha^* \to \beta$ to denote $\alpha^n \to \beta$ for some $n \geq 0$.

Due to space limits, some results in this section are stated without proofs, but they can be found in the electronic appendix of the paper.

DEFINITION 30. A one-step transition judgment $\sigma \triangleright P \xrightarrow{A} Q$ (respectively, $\sigma \triangleright P \xrightarrow{A}\!\!\!\!\rightarrow Q$) is a *higher-order patterned judgment*, or patterned judgment for short, if

(1) every occurrence of the free variables in the judgment is applied to distinct names in σ or internally bound names, i.e., $M\ a_1 \cdots a_n$, where $a_i \in \sigma$ or it is bound by some λ-abstraction, and $a_1, \ldots, a_n$ are pairwise distinct,

(2) the only occurrences of free variables in P are those of type $n^n \to n$ where $n \geq 0$, and the only occurrences of free variables in A are those of type $n^n \to n$ or $n^n \to \mathrm{a}$,

(3) and Q is of the form $(M\ \vec{\sigma})$ for some variable M.

The process term P in the transition predicate $P \xrightarrow{A} Q$ and $P \xrightarrow{A}\!\!\!\!\rightarrow Q$ is called a *primary* process term. The notion of patterned judgments extends to non-atomic judgments, which are defined inductively as follows:

—$\sigma \triangleright \top$ is a patterned judgment,

—if $\sigma \triangleright B$ and $\sigma \triangleright C$ are patterned judgments such that both judgments have no free variables in common which are of type $n^* \to p$, then $\sigma \triangleright B \wedge C$ is a patterned judgment,

—if $\sigma x \triangleright B$ is a patterned judgment, then $\sigma \triangleright \nabla x.B$ is a patterned judgment,

—and if $\sigma \triangleright B[h\,\vec{\sigma}/y]$ is a patterned judgment then $\sigma \triangleright \exists y.B$ is a patterned judgment, provided that h is of type $n^n \to \mathrm{a}$ or $n^n \to p$, and h is not free in $\exists y.B$.

Two patterned judgments $\mathcal{A}$ and $\mathcal{B}$ are *p-compatible* if they do not have variables in common which are of type $n^* \to p$.

The restrictions on the occurences of free variables in patterned judgments are similar to the restrictions used in higher-order pattern unification. This is to ensure that proof search for patterned judgments involves only higher-order pattern unification.

Let ρ be a substitution and let Σ be a signature. We write $\Sigma \vdash \rho$ if for every $x \in \mathrm{dom}(\rho)$ of type τ, we have $\Sigma \vdash \rho(x) : \tau$. Two signatures Σ and Σ' are said to be compatible if whenever $x : \tau_1 \in \Sigma$ and $y : \tau_2 \in \Sigma'$, $x = y$ implies $\tau_1 = \tau_2$. Given two signature-and-substitution pairs (Σ_1, ρ_1) and (Σ_2, ρ_2) such that Σ_1 and Σ_2 are compatible, and $\Sigma_1 \vdash \rho_1$ and $\Sigma_2 \vdash \rho_2$, we write $(\Sigma_1, \rho_1) \circ (\Sigma_2, \rho_2)$ to denote the pair

$$(\Sigma_1 \rho_2 \cup \Sigma_2, \rho_1 \circ \rho_2).$$

This definition of composition extends straightforwardly to composition between a pair and a set or a list of pairs.

Let us call a signature-substitution pair (Σ, ρ) a *solution* for a patterned judgment $\mathcal{C}$ if $\Sigma \vdash \rho$ and $\Sigma\,;\, . \vdash \mathcal{C}\rho$ is provable. In proving the adequacy of the encoding of bisimulation and modal logics for the π calculus, we often want to find all possible solutions to a given transition relation, which corresponds to enumerating all possible continuations of a given process. For this purpose, we define a construction of "open" derivation trees for a given list of patterned judgments Δ. Open derivation trees are trees made of nodes which are instances of certain inference rules. This construction gives us a set of derivation trees for the sequent $\Delta \vdash \bot$, following a certain order of rule applications. As we shall see, the construction of the trees basically amounts to application of left-introduction rules to Δ. We are interested in collecting all the substitutions generated by the $def\mathcal{L}$ rule in these trees, which we will show to correspond to the solutions for the patterned judgments in Δ.

Definition 31. Let Δ be a list of patterned judgments such that its elements are pairwise p-compatible, and let (Σ, θ) be a pair such that $\Sigma \vdash \theta$, and that the free variables of Δ are in Σ. An *open inference rule* is an inference on triples of the form $(\Sigma', \Delta', \theta')$ where Σ' is a signature, Δ' is a list of patterned judgments and θ' is a substitution such that $\Sigma' \vdash \theta'$. We will use the notation $(\Sigma', \theta') \vdash \Delta'$ to denote such a triple. *Open derivation trees* are derivations constructed using the following open inference rules:

$$\frac{}{(\Sigma, \theta) \vdash [\,]}\ open \qquad \frac{(\Sigma, \theta) \vdash \Delta'}{(\Sigma, \theta) \vdash \bar{n} \triangleright \top, \Delta'}\ \top$$

$$\frac{(\Sigma, \theta) \vdash \bar{n} \triangleright A, \bar{n} \triangleright B, \Delta'}{(\Sigma, \theta) \vdash \bar{n} \triangleright A \wedge B, \Delta'}\ \wedge \qquad \frac{(\Sigma \cup \{h\}, \theta) \vdash \bar{n} \triangleright B\,(h\,\bar{n}), \Delta'}{(\Sigma, \theta) \vdash \bar{n} \triangleright \exists x.B\,x, \Delta'}\ \exists$$

$$\frac{\{(\Sigma\rho, \theta \circ \rho) \vdash \mathcal{B}\rho, \Delta\rho \mid \rho \in CSU(\mathcal{A}, \mathcal{H}),\ \mathcal{H} \stackrel{\triangle}{=} \mathcal{B}\}}{(\Sigma, \theta) \vdash \mathcal{A}, \Delta}\ def$$

In the $\exists$-rule, the eigenvariable h is new, i.e., it is not in Σ. In the def-rule, we require that for every $\rho \in CSU(\mathcal{A}, \mathcal{H})$, the judgments $\mathcal{B}\rho, \Delta\rho$ are patterned judgments. That is, we restrict the CSU's to those that preserves the pattern restrictions on judgments. The instances of the *open*-rule in an open derivation are called *open leaves* of the derivation. Given an open derivation Π, we denote with $\mathcal{L}(\Pi)$ the set of signature-substitution pairs in the open leaves of Π.

Definition 32. The measure of a patterned judgment $\sigma \triangleright B$, written $|\sigma \triangleright B|$, is defined as the number of process constructors occuring in the primary terms in B. The measure of a list of judgments Δ is the multiset of measures of the judgments in Δ.

Lemma 33. *Let Δ be a list of patterned judgments such that its elements are pairwise p-compatible, and whose variables are in a given signature Σ. Let θ be a substitution such that $\Sigma \vdash \theta$. Then there exists an open derivation Π of $(\Sigma, \theta) \vdash \Delta$.*

Lemma 34. *Let Σ_1, Σ_2, θ_1 and θ_2 be signatures and substitutions such that $\Sigma_1 \vdash \theta_1$ and $\Sigma_2 \vdash \theta_2$. Let Δ be a list of patterned jugdments such that its elements*

are pairwise p-compatible, and all its free variables are in Σ_2. *If there exists an open derivation* Π_1 *of* $(\Sigma_1\theta_2 \cup \Sigma_2, \theta_1 \circ \theta_2) \vdash \Delta$, *then there exists an open derivation* Π_2 *of* $(\Sigma_2, \theta_2) \vdash \Delta$ *of the same height such that*

$$\mathcal{L}(\Pi_1) = (\Sigma_1, \theta_1) \circ \mathcal{L}(\Pi_2)$$

and vice versa.

The following lemma states that the open leaves in an open derivation are solutions of the patterned judgments on the root of the derivation tree. This can be proved by induction on the height of derivation and case analysis on the definition clauses of one-step transitions.

LEMMA 35. *Let* Δ *be a list of patterned judgments such that its elements are pairwise p-compatible and whose variables are in a given signature* Σ. *Let* Π *be an open derivation of* $(\Sigma, \epsilon) \vdash \Delta$. *Then for every element* $\mathcal{C} \in \Delta$ *and every pair* $(\Sigma', \theta) \in \mathcal{L}(\Pi)$, *the sequent* $\Sigma' ; . \vdash \mathcal{C}\theta$ *is provable.*

We are now ready to define the following derived rules. The rule one_f enumerates all possible free-actions that a process can perform:

$$\frac{\{\Sigma' ; \Gamma\theta \vdash \mathcal{C}\theta \mid (\Sigma', \theta) \in \mathcal{L}(\Pi)\}}{\Sigma ; \bar{n} \triangleright P \xrightarrow{A} Q, \Gamma \vdash \mathcal{C}} \; one_f$$

where $\bar{n} \triangleright P \xrightarrow{A} Q$ is a patterned judgment and Π is an open derivation of $(\Sigma, \epsilon) \vdash \bar{n} \triangleright P \xrightarrow{A} Q$. The corresponding rule for bound input or bound output transition is defined analogously, i.e.,

$$\frac{\{\Sigma' ; \Gamma\theta \vdash \mathcal{C}\theta \mid (\Sigma', \theta) \in \mathcal{L}(\Pi)\}}{\Sigma ; \bar{n} \triangleright P \xrightarrow{X} M, \Gamma \vdash \mathcal{C}} \; one_b.$$

where Π is an open derivation of $(\Sigma, \epsilon) \vdash \bar{n} \triangleright P \xrightarrow{X} M$. Since open inference rules are essentially invertible left-rules of $FO\lambda^{\Delta\nabla}$, these derived rules are sound and invertible.

LEMMA 36. *The rules* one_f *and* one_b *are derivable in* $FO\lambda^{\Delta\nabla}$ *and are both invertible.*

We can now prove Proposition 10.

PROOF. Suppose that $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ does not hold in the π-calculus. We show that the sequent $\neg\nabla\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ is derivable in $FO\lambda^{\Delta\nabla}$. This is equivalent to proving the sequent

$$; \vec{n} \triangleright [\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!] \vdash \bot.$$

We apply either one_f or one_b to the sequent (bottom-up), depending on whether α is a free or a bound action. In both cases, if the premise of the one_f or one_b is empty, then we are done. Otherwise, there exists a substitution θ such that $(\nabla\vec{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!])\theta$ is derivable in $FO\lambda^{\Delta\nabla}$. Since the transition judgment is ground, this would

mean that $\nabla\vec{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ is derivable, and by Proposition 9, the transition $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ holds in the π-calculus, contradicting our assumption.

Conversely, suppose that $\neg\nabla\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ is derivable in $FO\lambda^{\Delta\nabla}$. Then $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ cannot be a transition in the π-calculus, for otherwise, we would have $\vdash \nabla\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ by Proposition 9, and by cut, we would have a proof of $\bot$, which is impossible. □

B. ADEQUACY OF THE SPECIFICATIONS OF BISIMULATIONS

We need some auxiliary lemmas that concern the structures of cut free proofs. The next three lemmas can be proved straightforwardly by permutation of inference rules.

LEMMA 37. *Let Π be a cut-free derivation of $\cdot\,;\Gamma \vdash \mathcal{C}$, where $\mathcal{C}$ contains a non-equality atomic formula and every judgment in Γ is in one of the following forms:*

$$\bar{n} \triangleright \forall x\forall y.x = y \vee x \neq y \qquad \bar{n} \triangleright \forall y.a = y \vee a \neq y \qquad \bar{n} \triangleright a = b \vee a \neq b$$

$$\bar{n} \triangleright a = a \vee a \neq a \qquad \bar{n} \triangleright a = a \qquad \bar{n} \triangleright a \neq b$$

for some $\bar{n}$ and distinct names a, b in $\bar{n}$. Then there exists a derivation of the sequent which ends with a right-introduction rule on $\mathcal{C}$.

LEMMA 38. *The def$R$ rule, applied to lbisim P Q, for any P and Q, is invertible.*

LEMMA 39. *The def$R$ rule, applied to ebisim P Q, for any P and Q, is invertible.*

B.1 Adequacy of the specification of late bisimulation

In the following, we use the notation $x_1 \neq x_2 \neq \cdots \neq x_{n-1} \neq x_n$ to abbreviate the conjunction

$$\bigwedge\{x_i \neq x_j \mid i, j \in \{1, \ldots, n\}, i \neq j\}.$$

With a slight abuse of notation, we shall write $\mathcal{X} \supset B$, where $\mathcal{X}$ is a finite set of formula $\{B_1, \ldots, B_n\}$, to mean $B_1 \wedge \cdots \wedge B_n \supset B$, and we shall write $\nabla y.\mathcal{X}$ to mean the formula $\nabla y.B_1 \wedge \cdots \wedge \nabla y.B_2$.

LEMMA 40. *Let $\mathtt{P}$ and $\mathtt{Q}$ be two late-bisimilar finite π-processes and let $n_1, \ldots, n_k$ be the free names in $\mathtt{P}$ and $\mathtt{Q}$. Then for some finite set $\mathcal{X} \subset \mathcal{E}$, we have*

$$\vdash \forall n_1 \cdots \forall n_k.(\mathcal{X} \wedge n_1 \neq n_2 \neq \cdots \neq n_k \supset \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}). \tag{3}$$

PROOF. We construct a proof of formula (3) by induction on the size of $\mathtt{P}$ and $\mathtt{Q}$, i.e., the number of action prefixes in $\mathtt{P}$ and $\mathtt{Q}$. It can be easily shown that the number of prefixes in a process is reduced by transitions, for finite processes. By applying the introduction rules for $\forall$, $\supset$ and unfolding the definition of *lbisim* (bottom up) to the formula (3), we get the following three sequents:

(1) $n_1, \cdots, n_k, A, P'\,; \mathcal{X}, n_1 \neq \cdots \neq n_k, \mathtt{P} \xrightarrow{A} P' \vdash \exists Q'.\mathtt{Q} \xrightarrow{A} Q' \wedge \mathit{lbisim}\ P'\ Q'$

(2) $n_1, \cdots, n_k, X, P'\,; \mathcal{X}, n_1 \neq \cdots \neq n_k, \mathtt{P} \xrightarrow{\downarrow X} P' \vdash \exists Q'.\ \mathtt{Q} \xrightarrow{\downarrow X} Q' \wedge \forall w.\mathit{lbisim}\ (P'w)\ (Q'w)$

(3) $n_1, \cdots, n_k, X, P' ; \mathcal{X}, n_1 \neq \cdots \neq n_k, \mathtt{P} \xrightarrow{\uparrow X} P' \vdash \exists Q'. \mathtt{Q} \xrightarrow{\uparrow X} Q' \wedge \nabla w. lbisim\ (P'w)\ (Q'w)$

and their symmetric counterparts (obtained by exchanging the role of P and Q). The set $\mathcal{X}$ is left unspecified above, since it will be constructed by induction hypothesis (in the base case, where both P and Q are deadlocked processes, define $\mathcal{X}$ to be the empty set). We show here how to construct proofs for these three sequents; their symmetric counterparts can be proved similarly. In all these three cases, we apply either the one_f rule (for sequent 1) or the one_b rule (for sequent 2 and 3). If this application of one_f (or one_b) results in two distinct name-variables, say n_1 and n_2, to be identified, then the sequent is proved by using the assumption $n_1 \neq n_2$. Therefore the only interesting cases are when the name-variables $n_1, \cdots, n_k$ are instantiated to distinct name-variables, say, $m_1, \cdots, m_k$. In the following we assume that the substitution in the premises of one_f or one_b are non-trivial, meaning that they do not violate the assumption on name-distinction above.

Sequent 1. In this case, after applying the one_f rule bottom up and discharging the trivial premises, we need to prove, for each θ associated with the rule, the sequent

$$m_1, \cdots, m_k, \Sigma ; \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \exists Q'. \mathtt{Q}\theta \xrightarrow{A\theta} Q' \wedge lbisim\ (P'\theta)\ Q'$$

for some signature Σ. We give a top-down construction of a derivation of this sequent as follows. By Lemma 35, we know that

$$\vdash m_1, \cdots, m_k, \Sigma ; . \vdash \mathtt{P}\theta \xrightarrow{A\theta} P'\theta.$$

Since $m_1, \ldots, m_k$ are the only free names in $\mathtt{P}\theta$, we can show by induction on proofs that Σ in the sequent is redundant and can be removed, thus

$$\vdash m_1, \cdots, m_k ; . \vdash \mathtt{P}\theta \xrightarrow{A\theta} P'\theta.$$

By the adequacy of one-step transition (Proposition 9) and Proposition 4, we have $\mathtt{P}\theta \xrightarrow{A\theta} P'\theta$. Notice that P is a renaming of $\mathtt{P}\theta$, since $m_1, \ldots, m_k$ are pairwise distinct. We recall that both one-step transitions and (late) bisimulation are closed under injective renaming (see, *e.g.*, [Milner et al. 1992]). Therefore, there exist α and R such that $\mathtt{P} \xrightarrow{\alpha} \mathtt{R}$, where α and R are obtained from $A\theta$ and $P'\theta$, respectively, under the same injective renaming. Since P and Q are bisimilar, there exists T such that $\mathtt{Q} \xrightarrow{\alpha} \mathtt{T}$, hence, by injective renaming and the adequacy result for one-step transitions, the sequent $m_1, \cdots, m_k ; . \vdash \mathtt{Q}\theta \xrightarrow{A\theta} \mathtt{T}\theta$ is provable. It remains to show that

$$\vdash m_1, \cdots, m_k ; \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash lbisim\ (P'\theta)\ (\mathtt{T}\theta)$$

By induction hypothesis (note that the size of $(\mathtt{R}, \mathtt{T})$ is smaller than $(\mathtt{P}, \mathtt{Q})$), we have

$$\vdash \forall x_1 \cdots \forall x_j. \mathcal{X}' \wedge x_1 \neq \cdots \neq x_j \supset lbisim\ \mathtt{R}\ \mathtt{T}$$

where $\{x_1, \ldots, x_j\}$ is a subset of $\{n_1, \ldots, n_k\}$. We can weaken the formula with extra variables and assumptions to get

$$\vdash \forall n_1 \cdots \forall n_k. \mathcal{X}' \wedge n_1 \neq \cdots \neq n_k \supset lbisim\ \mathtt{R}\ \mathtt{T}.$$

Now since the $\forall R$ and $\supset R$ rules are invertible, this means

$$\vdash n_1, \ldots, n_k \,;\, \mathcal{X}', n_1 \neq \cdots \neq n_k \vdash \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}.$$

Now define $\mathcal{X}$ to be $\mathcal{X}'$ and apply a renaming substitution which maps each n_i to m_i, we get a derivation of

$$m_1, \ldots, m_k \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \mathit{lbisim}\ (P'\theta)\ (\mathtt{T}\theta).$$

Since provability is closed under weakening of signature, we have

$$\vdash m_1, \ldots, m_k, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \mathit{lbisim}\ (P'\theta)\ (\mathtt{T}\theta),$$

and together with provability of $m_1, \ldots, m_k \,;\, . \vdash \mathtt{Q}\theta \xrightarrow{A\theta} \mathtt{T}\theta$, we get

$$\vdash m_1, \cdots, m_k, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \mathtt{Q}\theta \xrightarrow{A\theta} \mathtt{T}\theta \wedge \mathit{lbisim}\ (P'\theta)\ \mathtt{T}\theta.$$

Finally, applying an $\exists R$ to this sequent, we get

$$\vdash m_1, \cdots, m_k, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \exists Q'.\mathtt{Q}\theta \xrightarrow{A\theta} Q' \wedge \mathit{lbisim}\ (P'\theta)\ Q'.$$

Sequent 2.. In this case, we need to prove the sequent

$$(*)\ m_1, \cdots, m_k, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \exists Q'.\mathtt{Q}\theta \xrightarrow{\downarrow X\theta} Q' \wedge \forall w.\mathit{lbisim}\ ((P'\theta)w)\ (Q'w)$$

for each non-trivial θ in the premises of one_b rule. By the same reasoning as in the previous case, we obtain, for every transition $P\theta \xrightarrow{x(w)} \mathtt{R}$, where $\mathtt{R} = (P'\theta)\, w$, another transition $Q\theta \xrightarrow{x(w)} \mathtt{T}$ such that for all name z $\mathtt{R}[z/w] \sim_l \mathtt{T}[z/w]$. It is enough to consider $k{+}1$ cases for z, i.e., those in which z is one of $m_1, \ldots, m_k$ and another where z is a new name, say m_{k+1}. By induction hypothesis, we have, for each $i \in \{1, \ldots, k\}$, a provable formula F_i

$$\forall m_1 \cdots \forall m_k.\mathcal{X}_1 \wedge m_1 \neq \cdots \neq m_k \supset \mathit{lbisim}\ (\mathtt{R}[m_i/w])\ (\mathtt{T}[m_i/w])$$

and a provable formula F_{k+1}:

$$\forall m_1 \cdots \forall m_{k+1}.\mathcal{X}_{k+1} \wedge m_1 \neq \cdots \neq m_{k+1} \supset \mathit{lbisim}\ (\mathtt{R}[m_{k+1}/w])\ (\mathtt{T}[m_{k+1}/w]).$$

Let $\mathcal{X}$ be the set

$$\mathcal{X} = \{\forall x \forall y.x = y \vee x \neq y\} \cup \{\mathcal{X}_i \mid i \in \{1, \ldots, k+1\}\}.$$

Then the sequent (*) is proved, in a bottom-up fashion, by instantiating Q' to $\lambda w.\mathtt{T}$, followed by an $\wedge\mathcal{R}$-rule, resulting in the sequents:

$$m_1, \ldots, m_k, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \mathtt{Q}\theta \xrightarrow{A\theta} \lambda w.\mathtt{T}$$

and

$$m_1, \ldots, m_k, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \forall w.\mathit{lbisim}\ \mathtt{R}\ \mathtt{T}$$

The first sequent is provable following the adequacy of one-step transition. For the second sequent, we apply the $\forall R$-rule to get the sequent

$$m_1, \ldots, m_k, m_{k+1}, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \mathit{lbisim}\ (\mathtt{R}[m_{k+1}/w])\ (\mathtt{T}[m_{k+1}/w]).$$

We then do a case analysis on the name m_{k+1}, using the assumption $\forall x \forall y. x = y \vee x \neq y$ in $\mathcal{X}$. Let $\mathtt{R}_{k+1} = \mathtt{R}[m_{k+1}/w]$ and let $\mathtt{T}_{k+1} = \mathtt{T}[m_{k+1}/w]$. We consider k instantiations, each instantiation compares m_{k+1} with m_i, for $i \in \{1, \ldots, k\}$. We thus get the following sequents:

$$
\begin{array}{ll}
(S_1) & \Sigma' \,;\, \Delta, m_1 = m_{k+1} \vdash \mathit{lbisim}\ \mathtt{R}_{k+1}\ \mathtt{T}_{k+1} \\
(S_2) & \Sigma' \,;\, \Delta, m_1 \neq m_{k+1}, m_2 = m_{k+1} \vdash \mathit{lbisim}\ \mathtt{R}_{k+1}\ \mathtt{T}_{k+1} \\
 & \vdots \\
(S_k) & \Sigma' \,;\, \Delta, m_1 \neq m_{k+1}, \ldots, m_{k-1} \neq m_{k+1}, m_k = m_{k+1} \vdash \mathit{lbisim}\ \mathtt{R}_{k+1}\ \mathtt{T}_{k+1} \\
(S_{k+1}) & \Sigma', m_{k+1} \,;\, \Delta, m_1 \neq m_2, \cdots, m_{k-1} \neq m_k, m_k \neq m_{k+1} \vdash \mathit{lbisim}\ \mathtt{R}_{k+1}\ \mathtt{T}_{k+1}
\end{array}
$$

Here Σ' denotes the set $\{m_1, \ldots, m_{k+1}\} \cup \Sigma$ and Δ denotes the set $\{\mathcal{X}, m_1 \neq \cdots \neq m_k\}$. Provability of these sequents follow from provability of $F_1, \ldots, F_{k+1}$.

Sequent 3. In this case, we need to prove the sequent

$$
(**) \qquad m_1, \cdots, m_k, \Sigma \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \begin{array}{l} \exists Q'. \mathtt{Q}\theta \xrightarrow{A\theta} Q' \wedge \\ \nabla w. \mathit{lbisim}\ ((P'\theta)w)\ (Q'w) \end{array}
$$

for each non-trivial θ in the premises of one_b rule. As in the previous case, we obtain $\mathtt{R}$ and $\mathtt{T}$ such that $\mathtt{P}\theta \xrightarrow{\bar{x}(w)} \mathtt{R}$ and $\mathtt{Q}\theta \xrightarrow{\bar{x}(w)} \mathtt{T}$ where $\lambda w.\mathtt{R} = P'\theta$. We assume, without loss of generality, that w is fresh, therefore by induction hypothesis we have that $\mathtt{R} \sim_l \mathtt{T}$ and

$$
\forall m_1 \cdots \forall m_k \forall w. \mathcal{X}' \wedge m_1 \neq \cdots \neq m_k \neq w \supset \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}.
$$

Now apply Proposition 3 to replace $\forall w$ with ∇w,

$$
\forall m_1 \cdots \forall m_k \nabla w. \mathcal{X}' \wedge m_1 \neq \cdots \neq m_k \neq w \supset \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}.
$$

And since ∇ distributes over all propositional connectives, we also have

$$
\forall m_1 \cdots \forall m_k. (\nabla w \mathcal{X}') \wedge \nabla w. (m_1 \neq \cdots \neq m_k) \wedge \nabla w (\bar{m} \neq w) \supset \nabla w. \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}.
$$

Let $\mathcal{X} = \nabla w \mathcal{X}'$. Now, since the right-introduction rules for $\forall$, ∇ and $\supset$ are all invertible, we have that the sequent

$$
(i) \qquad m_1, \ldots, m_k \,;\, \mathcal{X}, \nabla w. (m_1 \neq \cdots \neq m_k), \nabla w (\bar{m} \neq w) \vdash \nabla w. \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}
$$

is provable. It can be easily checked that the following sequents are provable:

$$
\begin{array}{l}
\nabla w. m_i \neq m_j \vdash m_i \neq m_j, \text{ for any } i \text{ and } j. \\
\vdash \nabla w. m_i \neq w, \text{ for any } i \text{ (since } w \text{ is in the scope of } m_i).
\end{array}
$$

Therefore, by applying the cut rules to these sequents and sequent (i) above, we obtain

$$
(ii) \qquad m_1, \ldots, m_k \,;\, \mathcal{X}, m_1 \neq \cdots \neq m_k \vdash \nabla w. \mathit{lbisim}\ \mathtt{R}\ \mathtt{T},
$$

Provability of sequent $(**)$ then follows from provability of sequent (ii) above and the adequacy of the one-step transition (i.e., by instantiating Q' with $\lambda w.\mathtt{T}$). □

The following lemma shows that *lbisim* is symmetric. Its proof is straightforward by induction on derivations.

LEMMA 41. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two* π*-processes and let* $\bar{n}$ *be the list of all free names in* $\mathtt{P}$ *and* $\mathtt{Q}$*. If* $\vdash \mathcal{X} \supset \nabla \bar{n}. \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$*, for some* $\mathcal{X} \subset \mathcal{E}$*, then* $\vdash \mathcal{X} \supset \nabla \bar{n}. \mathit{lbisim}\ \mathtt{Q}\ \mathtt{P}$.

B.2 Proof for Theorem 16 (adequacy of late bisimulation specification)

Soundness. We define a set $\mathcal{S}$ as follows

$$\mathcal{S} = \{(\mathtt{P}, \mathtt{Q}) \mid\vdash \mathcal{X} \supset \nabla \bar{n}\ \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}, \text{ where } \mathrm{fn}(\mathtt{P}, \mathtt{Q}) \subseteq \{\bar{n}\} \text{ and } \mathcal{X} \subset \mathcal{E}\},$$

and show that $\mathcal{S}$ is a bisimulation set, that is, it is symmetric and closed with respect to the condition 1, 2 and 3 in Definition 12. The symmetry of $\mathcal{S}$ follows from Lemma 41.

Suppose that $(\mathtt{P}, \mathtt{Q}) \in \mathcal{S}$, that is, $\vdash \mathcal{X} \supset \nabla \bar{n}\ \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$ for some $\mathcal{X}$. Since $\mathit{def}R$ on lbisim is invertible (Lemma 38), and since $\wedge\mathcal{R}$, $\supset\ \mathcal{R}$, $\nabla\mathcal{R}$ and $\forall\mathcal{R}$ are also invertible, there is a proof of the formula that ends with applications of these invertible rules. From this and the definition of lbisim, we can infer that provability of $\mathcal{X} \supset \nabla \bar{n}\ \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$ implies provability of six other sequents, three of which are given in the following (the other three are symmetric counterparts of these sequents):

$$(a)\ \ P', A\,;\, \mathcal{X}, \bar{n} \triangleright \mathtt{P} \xrightarrow{A\bar{n}} (P'\bar{n}) \vdash \bar{n} \triangleright \exists Q'.\mathtt{Q} \xrightarrow{A\bar{n}} Q' \ \wedge\ \mathit{lbisim}\ (P'\bar{n})\ Q'$$

$$(b)\ \ M, X\,;\, \mathcal{X}, \bar{n} \triangleright \mathtt{P} \xrightarrow{\downarrow(X\bar{n})} (M\bar{n}) \vdash \bar{n} \triangleright \exists N.\mathtt{Q} \xrightarrow{\downarrow(X\bar{n})} N \ \wedge\ \forall y.\mathit{lbisim}\ (M\bar{n}y)\ (Ny)$$

$$(c)\ \ M, X\,;\, \mathcal{X}, \bar{n} \triangleright \mathtt{P} \xrightarrow{\uparrow(X\bar{n})} (M\bar{n}) \vdash \bar{n} \triangleright \exists N.\mathtt{Q} \xrightarrow{\uparrow(X\bar{n})} N \ \wedge\ \nabla y.\mathit{lbisim}\ (M\bar{n}y)\ (Ny)$$

By examing the structure of proofs of these three sequents, we show that $\mathcal{S}$ is closed under all possible transitions from $\mathtt{P}$ and $\mathtt{Q}$. We examine the three cases in Definition 12:

(1) Suppose $\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'$ for some free action α. Since $\mathtt{P} \xrightarrow{\alpha} \mathtt{P}'$, by the adequacy result for one-step transitions, we have that $\bar{n} \triangleright \mathtt{P} \xrightarrow{\alpha} \mathtt{P}'$ is derivable. Let $\rho = [\lambda\bar{n}.\alpha/A, \lambda\bar{n}.\mathtt{P}'/P']$. Applying ρ to the derivation of sequent (a), we get

$$\vdash \cdot\,;\, \mathcal{X}, \bar{n} \triangleright \mathtt{P} \xrightarrow{\alpha} \mathtt{P}' \vdash \bar{n} \triangleright \exists Q'.\mathtt{Q} \xrightarrow{\alpha} Q' \ \wedge\ \mathit{lbisim}\ \mathtt{P}'\ Q'.$$

By a cut between $\bar{n} \triangleright \mathtt{P} \xrightarrow{\alpha} \mathtt{P}'$ and this sequent, we obtain a derivation of

$$\cdot\,;\, \mathcal{X} \vdash \bar{n} \triangleright \exists Q'.\mathtt{Q} \xrightarrow{\alpha} Q' \ \wedge\ \mathit{lbisim}\ \mathtt{P}'\ Q'.$$

By Lemma 37, we know that there exists a derivation of this sequent which ends with a right-rule, hence, there exists a process $\mathtt{Q}'$ such that

$$\vdash \cdot\,;\, \mathcal{X} \vdash \bar{n} \triangleright \mathtt{Q} \xrightarrow{\alpha} \mathtt{Q}' \qquad\qquad \vdash \cdot\,;\, \mathcal{X} \vdash \bar{n} \triangleright \mathit{lbisim}\ \mathtt{P}'\ \mathtt{Q}'$$

It is easy to show that $\mathcal{X}$ plays no part in the proof of the first sequent, so it can be removed from the sequent. Hence by the adequacy of one-step transition, we have $\mathtt{Q} \xrightarrow{\alpha} \mathtt{Q}'$. Provability of the second sequent implies that $(\mathtt{P}', \mathtt{Q}')$ is in the set $\mathcal{S}$. Therefore we have shown that $\mathcal{S}$ is indeed closed under the α-transition.

(2) Suppose that $\mathtt{P} \xrightarrow{a(y)} \mathtt{P}'$. Applying a similar argument as in the previous case to Sequent (b), with substitution $\rho = [\lambda\bar{n}.a/X, \lambda\bar{n}\lambda y.\mathtt{P}'/M]$, we obtain a provable sequent

$$\cdot\,;\, \mathcal{X} \vdash \bar{n} \triangleright \exists N.\mathtt{Q} \xrightarrow{\downarrow a} N \ \wedge\ \forall y.\mathit{lbisim}\ \mathtt{P}'\ (Ny).$$

Again, as in the previous case, using Lemma 37, we can show that $\mathtt{Q} \xrightarrow{a(y)} \mathtt{Q}'$ for some process $\mathtt{Q}'$ such that $\vdash \cdot ; \mathcal{X} \vdash \bar{n} \triangleright \forall y.\mathit{lbisim}\ \mathtt{P}'\ \mathtt{Q}'$. This implies that

(i) $\mathcal{X} \supset \nabla\bar{n}\forall y.\mathit{lbisim}\ \mathtt{P}'\ \mathtt{Q}'$,
(ii) $\mathcal{X} \supset \nabla\bar{n}.\mathit{lbisim}\ (\mathtt{P}'[w/y])\ (\mathtt{Q}'[w/y])$, where $w \in \{\bar{n}\}$,
(iii) $(\nabla y.\mathcal{X}) \supset \nabla y\nabla\bar{n}.\mathit{lbisim}\ \mathtt{P}'\ \mathtt{Q}'$

are all provable. The formula (ii) is obtained from (i) by instantiating y with one of $\bar{n}$. The formula (iii) is obtained from (i) as follows: Since

$$\nabla x\forall y.P\,x\,y \supset \forall y\nabla x.P\,x\,y \quad \text{and} \quad (A \supset \forall y.B) \supset (\forall y.(A \supset B)),$$

where y is not free in A, are theorems of $FO\lambda^{\Delta\nabla}$, we can enlarge the scope of y in (i) to the outermost level: hence, we have that $\forall y(\mathcal{X} \supset \nabla\bar{n}.\mathit{lbisim}\ \mathtt{P}'\ \mathtt{Q}')$ is provable. Now apply Proposition 3 to turn $\forall y$ into ∇y, then distribute the ∇y over the implication $\supset$ and conjunction $\wedge$, and we have (iii).
It remains to show that for every name w, the pair $(\mathtt{P}'[w/y], \mathtt{Q}'[w/y])$ is in $\mathcal{S}$. There are two cases to consider: The case where w is among $\bar{n}$ follows straightforwardly from (ii), the other case, where w is a new name, follows from (iii).

(3) Suppose $\mathtt{P} \xrightarrow{\bar{a}(y)} \mathtt{P}'$. Using the same argument as in the previous case, we can show that there exists a process $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{\bar{a}(y)} \mathtt{Q}'$ and such that

$$\vdash \cdot ; \mathcal{X} \vdash \bar{n} \triangleright \nabla y.\mathit{lbisim}\ \mathtt{P}'\ \mathtt{Q}'.$$

The latter entails that $(\mathtt{P}', \mathtt{Q}') \in \mathcal{S}$, as required.

B.3 Completeness

We are given $\mathtt{P} \sim_l \mathtt{Q}$ and we need to show that $\vdash \mathcal{X} \supset \nabla\bar{n}.\mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$, where $\mathcal{X} \subset \mathcal{E}$ and $\bar{n} = \{n_1, \ldots, n_k\}$ include all the free names in $\mathtt{P}$ and $\mathtt{Q}$. From Lemma 40 we have that

$$\vdash \forall n_1 \cdots \forall n_k(\mathcal{X}' \wedge n_1 \neq \cdots \neq n_k \supset \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q})$$

for some $\mathcal{X}' \subset \mathcal{E}$. By Proposition 3, we can turn all the $\forall$ into ∇, hence

$$\vdash \nabla n_1 \cdots \nabla n_k(\mathcal{X}' \wedge n_1 \neq \cdots \neq n_k \supset \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}).$$

Since ∇ distributes over all propositional connectives, we have

$$\vdash (\nabla\bar{n}.\mathcal{X}') \wedge \nabla\bar{n}.(n_1 \neq \cdots \neq n_k) \supset \nabla\bar{n}.\mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}.$$

Now, $\nabla\bar{n}.n_1 \neq \cdots \neq n_k$ is a theorem of $FO\lambda^{\Delta\nabla}$ (since any two distinct ∇-quantified names are not unifiable), therefore by modus ponens we have

$$\vdash \nabla\bar{n}.\mathcal{X}' \supset \nabla\bar{n}.\mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}.$$

Let $\mathcal{X} = \nabla\bar{n}.\mathcal{X}'$, then we have $\mathcal{X} \supset \nabla\bar{n}.\mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$ as required. □

B.4 Adequacy of the specification of early bisimulation

The proof for the adequacy of the specification of early bisimulation follows a very similar outline as that of late bisimulation. The proof is rather tedious and is not very enlightening. We therefore omit the proof and refer interested readers to the electronic appendix of the paper for more details.

B.5 Adequacy of the specification of open bisimulation

Proof of Lemma 19: The proof proceeds by induction on the length of the quantifier prefix $\mathcal{Q}\bar{x}$. At each stage of the induction, we construct a quantifier prefix $\mathcal{Q}\bar{y}$ such that $\mathcal{Q}\bar{x}.P \supset \mathcal{Q}\bar{y}.P\theta$ and $D\theta$ corresponds to the $\mathcal{Q}\bar{y}$-distinction. In the base case, where the quantifier prefix $\mathcal{Q}\bar{x}$ is empty, the quantifier $\mathcal{Q}\bar{y}$ is also the empty prefix. In this case we have $P\theta = P$, therefore $P \supset P\theta$ holds trivially. The inductive cases are as follows:

—Suppose $\mathcal{Q}\bar{x}.P = \mathcal{Q}'\bar{u}\nabla z.P$. Let D' be the distinction that corresponds to $\mathcal{Q}'\bar{u}$. Note that by definition, we have

$$D = D' \cup \{(z, v), (x, v) \mid v \in D'\},$$

Let θ' be the substitution θ with domain restricted to $\{\bar{u}\}$. Since θ respects D, obviously θ' respects D' and $\theta(z) \neq \theta(v)$ for all $v \in D'$. By induction hypothesis, we have a proof of the formula

$$\mathcal{Q}\bar{u}(\nabla z.P) \supset \mathcal{Q}\bar{m}(\nabla z.P)\theta'$$

for some quantifier prefix $\mathcal{Q}\bar{m}$ such that $D'\theta'$ is the $\mathcal{Q}\bar{m}$-distinction. Note that by since z is not in the domain of θ', we have $(\nabla z.P)\theta' = \nabla z.(P\theta')$. Let $w = \theta(z)$. Since w is distinct from all other free names in $D'\theta'$, we can rename z with w, thus,

$$\vdash \mathcal{Q}\bar{u}\nabla z.P \supset \mathcal{Q}\bar{m}\nabla w.P(\theta' \circ [w/z])$$

But $\theta' \circ [w/z]$ is exactly θ. Let $\mathcal{Q}\bar{y}$ be the prefix $\mathcal{Q}\bar{m}\nabla w$. It then follows that

$$\vdash \mathcal{Q}\bar{x}.P \supset \mathcal{Q}\bar{y}.P\theta.$$

Moreover, $D\theta$ can be easily shown to be the $\mathcal{Q}\bar{y}$-distinction.

—Suppose $\mathcal{Q}\bar{x} = \mathcal{Q}'\bar{u}\forall z.P$. Note that in this case, the $\mathcal{Q}\bar{x}$-distinction and $\mathcal{Q}'\bar{u}$-distinction co-incide, i.e., both are the same distinction D. Moreover, $z \notin \mathrm{fv}(D)$. Let θ' be the substitution θ restricted to the domain $\{\bar{u}\}$. By induction hypothesis, we have that

$$\vdash \mathcal{Q}\bar{u}(\forall z.P) \supset \mathcal{Q}\bar{m}(\forall z.P)\theta',$$

for some quantifier prefix $\mathcal{Q}\bar{m}$ such that $\mathcal{Q}\bar{m}$ corresponds to $D\theta'$. Note that $D\theta' = D\theta$, because $z \notin \mathrm{fv}(D)$. There are two cases to consider when constructing $\mathcal{Q}\bar{y}$. The first case is when z is identified, by θ, with some name in $\{\bar{u}\}$. In this case, by the property of universal quantification, we have that

$$\vdash \mathcal{Q}\bar{u}\forall z.P \supset \mathcal{Q}\bar{m}.P\theta.$$

In this case, we let $\mathcal{Q}\bar{y} = \mathcal{Q}\bar{m}$. Note that $D\theta'$ is the same as $D\theta$ in this case. Therefore $D\theta$ is the $\mathcal{Q}\bar{y}$-distinction.
For the second case, we have that z is instantiated by θ to a new name, say w. Then following the same argument as the case with ∇, we have that

$$\vdash \mathcal{Q}\bar{u}\forall z.P \supset \mathcal{Q}\bar{m}\forall w.P\theta.$$

In this case, we let $\mathcal{Q}\bar{y} = \mathcal{Q}\bar{m}\forall w$. Note that in this case the $\mathcal{Q}\bar{y}$-distinction also coincides with $\mathcal{Q}\bar{m}$-distinction, i.e., both are the same set $D\theta$. □

In the proof of soundness of open bisimulation to follow, we make use of a property of the structure of proofs of certain sequents. The following three lemmas state some meta-level properties of $FO\lambda^{\Delta\nabla}$. Their proofs are easy and are omited.

LEMMA 42. *Suppose the sequent* $\Sigma\,;\,\Delta \vdash \mathcal{C}$ *is provable, where* C *is an existential judgment and* Δ *is a set of inequality between distinct terms, i.e., every element of* Δ *is of the form* $\bar{n} \triangleright s \neq t$*, for some* $\bar{n}$*,* s *and* t*. Then there exists a proof of the sequent ending with* $\exists\mathcal{R}$ *applied to* C*.*

LEMMA 43. *For any positive formula context* $C[]$*,* $\vdash C[\forall x.B] \supset C[B[t/x]]$*.*

LEMMA 44. *Let* $\mathcal{Q}\bar{x}$ *be a quantifier prefix. Suppose* $\mathcal{Q}\bar{x}.P$ *and* $\mathcal{Q}\bar{x}.P \supset Q$ *are provable. Then* $\mathcal{Q}\bar{x}.Q$ *is provable.*

LEMMA 45. *Let* D *be a conjunction of inequality between terms. If* $\vdash \mathcal{Q}\bar{x}.D \supset \nabla y.P$*, where* y *is not free in* D*, then* $\vdash \mathcal{Q}\bar{x}\nabla y.D \supset P$*.*

The following lemma is a simple corollary of Proposition 9 and Proposition 4.

LEMMA 46. $\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}$ *if and only if* $\mathcal{Q}\bar{n}.[\![\mathtt{P} \xrightarrow{\alpha} \mathtt{Q}]\!]$ *is provable, where* $\mathcal{Q}\bar{n}$ *is a quantifier prefix and* $\bar{n}$ *are the free names of* $\mathtt{P}$*.*

To prove soundness of open bisimulation specification, we define a family of set $\mathcal{S}$ in the following, and show that it is indeed an open bisimulation.

$$\begin{aligned}\mathcal{S}_D = \{(\mathtt{P},\mathtt{Q}) \mid\ & \vdash \mathcal{Q}\bar{n}.[D'] \supset \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q} \text{ and} \\ & \mathrm{fn}(\mathtt{P},\mathtt{Q},D') = \{\bar{n}\}, D = D' \cup D'', \\ & \text{where } D'' \text{ is the } \mathcal{Q}\bar{n}\text{-distinction. } \}\end{aligned}$$

Suppose $(\mathtt{P},\mathtt{Q}) \in \mathcal{S}_D$. That is, $\vdash \mathcal{Q}\bar{n}.[D'] \supset \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$. Let D'' be the distinction that corresponds to the prefix $\mathcal{Q}\bar{n}$. We have to show that for every name substitution θ which respects D, the set $\mathcal{S}$ is closed under conditions 1, 2, and 3 in Definition 15. Since θ respects D, it also respects D'' (since D'' is a subset of D). Therefore, it follows from Lemma 19 that there exists a prefix $\mathcal{Q}\bar{x}$ such that $D''\theta$ is the $\mathcal{Q}\bar{x}$-distinction, and

$$\vdash \mathcal{Q}\bar{x}.[D'\theta] \supset \mathit{lbisim}\ (\mathtt{P}\theta)\ (\mathtt{Q}\theta).$$

By the invertibility of $defR$ on $lbisim$ and the right-introduction rules for $\forall$, ∇, $\supset$ and $\wedge$, we can infer that provability of the above formula implies provability of six other formulas, three of which are given in the following (the other three are symmetric variants of these formulas):

(a) $\mathcal{Q}\bar{x}.[D'\theta] \supset \forall P'\forall A.\mathtt{P}\theta \xrightarrow{A} P' \supset \exists Q'.\mathtt{Q}\theta \xrightarrow{A} Q' \wedge \mathit{lbisim}\ P'\ Q'$

(b) $\mathcal{Q}\bar{x}.[D'\theta] \supset \forall M\forall X.\mathtt{P}\theta \xrightarrow{\downarrow X} M \supset \exists N.\mathtt{Q}\theta \xrightarrow{\downarrow X} N \wedge \forall w.\mathit{lbisim}\ (Mw)\ (Nw)$

(c) $\mathcal{Q}\bar{x}.[D'\theta] \supset \forall M\forall X.\mathtt{P}\theta \xrightarrow{\uparrow X} M \supset \exists N.\mathtt{Q}\theta \xrightarrow{\uparrow X} N \wedge \nabla w.\mathit{lbisim}\ (Mw)\ (Nw)$

Using provability of these formulas, we show that $\mathcal{S}$ is closed under free actions, bound input actions and bound output actions.

—Suppose $\mathtt{P}\theta \xrightarrow{\alpha} \mathtt{R}$ where α is a free action. By Lemma 46, we have that

$$\vdash \mathcal{Q}\bar{x}.\mathtt{P}\theta \xrightarrow{\alpha} \mathtt{R}. \tag{4}$$

From formula (a) and Lemma 43, we have that

$$\vdash \mathcal{Q}\bar{x}.[D'\theta] \supset \mathtt{P}\theta \xrightarrow{\alpha} \mathtt{R} \supset \exists Q'.\mathtt{Q}\theta \xrightarrow{\alpha} Q' \wedge \mathit{lbisim}\ \mathtt{R}\ Q'. \tag{5}$$

Applying Lemma 44 to formula (4) and (5) above, we have that

$$\vdash \mathcal{Q}\bar{x}.[D'\theta] \supset \exists Q'.\mathtt{Q}\theta \xrightarrow{\alpha} Q' \wedge \mathit{lbisim}\ \mathtt{R}\ Q'.$$

The latter implies, by the invertibility of the right rules for ∇ and $\forall$, provability of the sequent

$$\Sigma\,;\, D_1 \vdash \bar{m} \triangleright \exists Q'.\mathtt{Q}' \xrightarrow{\alpha'} Q' \wedge \mathit{lbisim}\ \mathtt{R}'\ Q'$$

where Σ are the eigenvariables corresponding to the universally quantified variables in $\mathcal{Q}\bar{x}$ (with appropriate raising) and $\bar{m}$ corresponds to the ∇-quantified variables in the same prefix. The terms $\mathtt{Q}'$, $\mathtt{R}'$, D_1 and α' are obtained from, respectively, $\mathtt{Q}\theta$, $\mathtt{R}$, $[D'\theta]$ and α by replacing their free names with their raised counterparts. Note that since θ respects D', the inequality in D_1 are those that relate distinct terms, hence, by Lemma 42, provability of the above sequent implies the existence of a term T such that

$$\vdash \Sigma\,;\, D_1 \vdash \bar{m} \triangleright \mathtt{Q}' \xrightarrow{\alpha'} T' \tag{6}$$

and

$$\vdash \Sigma\,;\, D_1 \vdash \bar{m} \triangleright \mathit{lbisim}\ \mathtt{R}'\ T'. \tag{7}$$

It can be shown by induction on the height of derivations that D_1 in the first sequent can be removed, hence we have that

$$\vdash \Sigma\,;\, . \vdash \bar{m} \triangleright \mathtt{Q}' \xrightarrow{\alpha'} T'.$$

Applying the appropriate introduction rules to this sequent (top down), we "unraise" the variables in Σ and obtain $\vdash \mathcal{Q}\bar{x}.\mathtt{Q}\theta \xrightarrow{\alpha} \mathtt{T}$, where $\mathtt{T}$ corresponds to T'. By Lemma 46, this means that $\mathtt{Q}\theta \xrightarrow{\alpha} \mathtt{T}$. It remains to show that $(\mathtt{R}, \mathtt{Q}) \in \mathcal{S}$. This is obtained from the sequent (7) above as follows. We apply the introduction rules for quantifiers and implication (top down) to sequent (7), hence unraising the variables in Σ and obtain the following provable formulas:

$$\mathcal{Q}\bar{x}.[D'\theta] \supset \mathit{lbisim}\ \mathtt{R}\ \mathtt{T},$$

from which it follows that $(\mathtt{R}, \mathtt{T}) \in \mathcal{S}_{D\theta}$.

—Suppose $\mathtt{P}\theta \xrightarrow{a(y)} \mathtt{R}$. As in the previous case, using Lemma 46, Lemma 43, Lemma 44 and formula (b) we can show that

$$\vdash \mathcal{Q}\bar{x}.[D'\theta] \supset \exists Q'.\mathtt{Q}\theta \xrightarrow{\downarrow a} N \wedge \forall w.\mathit{lbisim}\ (\mathtt{R}[w/y])\ (N\,y).$$

From this formula, we can show that there exists $\mathtt{T}$ such that $\mathcal{Q}\bar{x}.\mathtt{Q}\theta \xrightarrow{\downarrow a} \lambda z.\mathtt{T}$, therefore $\mathtt{Q}\theta \xrightarrow{a(z)} \mathtt{T}$, and that

$$\vdash \mathcal{Q}\bar{x}.[D'\theta] \supset \forall w.\mathit{lbisim}\ (\mathtt{R}[w/y])\ (\mathtt{T}[w/z]). \tag{8}$$

We need to show that for a fresh name w, $(\mathtt{R}[w/y], \mathtt{T}[w/z]) \in \mathcal{S}_{D\theta}$. From provability of formula (8), and the fact that $(A \supset \forall x.B) \supset \forall x(A \supset B)$, we obtain

$$\vdash \mathcal{Q}\bar{x}\forall w.[D'\theta] \supset \mathit{lbisim}\ (\mathtt{R}[w/y])\ (\mathtt{T}[w/z]).$$

Since the $\mathcal{Q}\bar{x}\forall w$-distinction is the same as $\mathcal{Q}\bar{x}$-distinction, the overal distinction encoded in the above formula is $D\theta$, therefore, by definition of $\mathcal{S}$, we have $(\mathtt{R}[w/y], \mathtt{T}[w/z]) \in \mathcal{S}_{D\theta}$.

—Suppose $\mathtt{P}\theta \xrightarrow{\bar{a}(y)} \mathtt{R}$. This case is similar to the bound input case. Applying the same arguments shows that there exists a process $\mathtt{T}$ such that $\mathtt{Q}\theta \xrightarrow{a(z)} \mathtt{T}$ and

$$\vdash \mathcal{Q}\bar{x}.[D'\theta] \supset \nabla w.\mathit{lbisim}\ (\mathtt{R}[w/y])\ (\mathtt{T}[w/z]). \tag{9}$$

We have to show that, for a fresh w, $(\mathtt{R}[w/y], \mathtt{T}[w/z]) \in \mathcal{S}_{D_2}$ where $D_2 = D\theta \cup \{w\} \times \mathrm{fn}(D\theta, \mathtt{P}\theta, \mathtt{Q}\theta)$. Note that the free names of $D\theta$, $\mathtt{P}\theta$ and $\mathtt{Q}\theta$ are all in $\bar{x}$ by definition. From formula (9) and Lemma 45, we have that

$$\vdash \mathcal{Q}\bar{x}\nabla w.[D'\theta] \supset \mathit{lbisim}\ (\mathtt{R}[w/y])\ (\mathtt{T}[w/z]).$$

Notice that the $\mathcal{Q}\bar{x}\nabla w$-distinction is $D''\theta \cup \{w\} \times \{\bar{x}\}$, and since $\bar{x}$ is the free names of $D\theta$, $\mathtt{P}\theta$ and $\mathtt{Q}\theta$, the overall distinction encoded by the above formula is exactly D_2, hence $(\mathtt{R}[w/y], \mathtt{T}[w/z]) \in \mathcal{S}_{D_2}$ as required. □

The proof of Theorem 22 is analogous to the completeness proof for Theorem 16. Suppose $\mathtt{P}$ and $\mathtt{Q}$ are open D-bisimilar. We construct a derivation of the formula

$$\forall n_1 \cdots \forall n_k([D] \supset \mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}) \tag{10}$$

by induction on the number of action prefixes in $\mathtt{P}$ and $\mathtt{Q}$. By applying the introduction rules for $\forall$, $\supset$ and unfolding the definition of lbisim (bottom up) to the formula (10), we get the following sequents:

(1) $n_1, \cdots, n_k, A, P'\,;\, [D], \mathtt{P} \xrightarrow{A} P' \vdash \exists Q'.\mathtt{Q} \xrightarrow{A} Q' \wedge \mathit{lbisim}\ P'\ Q'$

(2) $n_1, \cdots, n_k, X, P'\,;\, [D], \mathtt{P} \xrightarrow{\downarrow X} P' \vdash \exists Q'.\mathtt{Q} \xrightarrow{\downarrow X} Q' \wedge \forall w.\mathit{lbisim}\ (P'w)\ (Q'w)$

(3) $n_1, \cdots, n_k, X, P'\,;\, [D], \mathtt{P} \xrightarrow{\uparrow X} P' \vdash \exists Q'.\mathtt{Q} \xrightarrow{\uparrow X} Q' \wedge \nabla w.\mathit{lbisim}\ (P'w)\ (Q'w)$

and their symmetric counterparts. We show here how to construct proofs for these three sequents; the rest can be proved similarly. In all these three cases, we apply either the one_f rule (for sequent 1) or the one_b rule (for sequent 2 and 3). If this application of one_f (or one_b) results in two distinct name-variables, say n_1 and n_2, in D to be identified, then the sequent is proved by using the assumption $n_1 \neq n_2$ in D. Therefore the only interesting cases are when the instantiations of name-variables $n_1, \cdots, n_k$ respect the distinction D. In the following we assume the names $n_1, \ldots, n_k$ are instantiated to $m_1, \ldots, m_l$ and the distinction D is respected. Note that l may be smaller than k, depending on D, i.e., it may allow some names to be identified.

Sequent 1. In this case, after applying the one_f rule bottom up and discharging the trivial premises (*i.e.*, those that violates the distinction D), we need to prove,

for each θ associated with the rule, the sequent

$$m_1, \cdots, m_l, \Sigma\,;\, [D\theta] \vdash \exists Q'.\mathtt{Q}\theta \xrightarrow{A\theta} Q' \wedge \mathit{lbisim}\ (P'\theta)\ Q' \tag{11}$$

for some signature Σ. By Lemma 35, we know that $m_1, \cdots, m_l, \Sigma\,;\, . \vdash \mathtt{P}\theta \xrightarrow{A\theta} P'\theta$ is provable. Since $m_1, \ldots, m_l$ are the only free names in $\mathtt{P}\theta$, we can show by induction on proofs that Σ in the sequent is redundant and can be removed, thus the sequent $m_1, \cdots, m_l\,;\, . \vdash \mathtt{P}\theta \xrightarrow{A\theta} P'\theta$ is also provable. By the adequacy of one-step transition (Proposition 9) and Proposition 4, we have $\mathtt{P}\theta \xrightarrow{\alpha} \mathtt{R}$ for some free action α and $\mathtt{R}$ where $\alpha = A\theta$ and $P'\theta = \mathtt{R}$. Let θ' be θ with domain restricted to $\{n_1, \ldots, n_k\}$. Obviously, θ' respects D and $D\theta' = D\theta$. Since $\mathtt{P}$ and $\mathtt{Q}$ are open D-bisimilar, we have that there exists $\mathtt{T}$ such that $\mathtt{Q}\theta \xrightarrow{\alpha} \mathtt{T}$ and $\mathtt{R} \sim_o^{D\theta'} \mathtt{T}$, hence by induction hypothesis, we have that

$$\vdash \forall m_1 \cdots \forall m_l.[D\theta] \supset \mathit{lbisim}\ P'\theta\ \mathtt{T}. \tag{12}$$

Provability of sequent (11) thus follows from these facts, by instantiating Q' with $\mathtt{T}$.

Sequent 2.. In this case, we need to prove the sequent

$$m_1, \cdots, m_l, \Sigma\,;\, [D\theta] \vdash \exists Q'.\mathtt{Q}\theta \xrightarrow{\downarrow X\theta} Q' \wedge \forall w.\mathit{lbisim}\ ((P'\theta)w)\ (Q'w) \tag{13}$$

for each non-trivial θ in the premises of one_b rule. By the same reasoning as in the previous case, we obtain, for every transition $\mathtt{P}\theta \xrightarrow{x(w)} \mathtt{R}$, where $\mathtt{R} = (P'\theta)\,w$, another transition $\mathtt{Q}\theta \xrightarrow{x(w)} \mathtt{T}$ such that (we assume w.l.o.g. that w is fresh) $\mathtt{R} \sim_o^{D\theta} \mathtt{T}$. The former implies that $\mathtt{Q}\theta \xrightarrow{\downarrow x} \lambda w.\mathtt{T}$ is derivable, and the latter implies, by induction hypothesis, that

$$\forall m_1 \cdots \forall m_l \forall w.[D\theta] \supset \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}$$

is derivable. As in the previous case, from these two facts, we can prove the sequent (13) by instantiating Q' with $\lambda w.\mathtt{T}$.

Sequent 3. In this case, we need to prove the sequent

$$m_1, \cdots, m_l, \Sigma\,;\, [D\theta] \vdash \exists Q'.\mathtt{Q}\theta \xrightarrow{A\theta} Q' \wedge \nabla w.\mathit{lbisim}\ ((P'\theta)w)\ (Q'w) \tag{14}$$

for each non-trivial θ in the premises of one_b rule. As in the previous case, we obtain $\mathtt{R}$ and $\mathtt{T}$ such that $\mathtt{P}\theta \xrightarrow{\bar{x}(w)} \mathtt{R}$ and $\mathtt{Q}\theta \xrightarrow{\bar{x}(w)} \mathtt{T}$ where $\lambda w.\mathtt{R} = P'\theta$. We assume, without loss of generality, that w is fresh, therefore since $\mathtt{P} \sim_o^D \mathtt{Q}$, by definition we have that $\mathtt{R} \sim_o^{D'} \mathtt{T}$, where $D' = D\theta \cup \{x\} \times \mathrm{fn}(D\theta, \mathtt{P}\theta, \mathtt{Q}\theta)$. Note that the free names of $D\theta$, $\mathtt{P}\theta$ and $\mathtt{Q}\theta$ are exactly $m_1, \ldots, m_l$, so $D' = D\theta \cup \{x\} \times \{m_1, \ldots m_l\}$. Thus by induction hypothesis, the formula

$$\forall m_1 \cdots \forall m_l \forall w.[D'] \supset \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}.$$

Now apply Proposition 3 to replace $\forall w$ with ∇w,

$$\forall m_1 \cdots \forall m_k \nabla w.[D'] \supset \mathit{lbisim}\ \mathtt{R}\ \mathtt{T}.$$

And since ∇ distributes over all propositional connectives, we also have

$$\forall m_1 \cdots \forall m_k.(\nabla w.[D']) \supset \nabla w.\mathit{lbisim}\ \mathtt{R}\ \mathtt{T}.$$

It can be shown that $m_1, \ldots, m_l\,;\, . \vdash \nabla w.[D'] \supset [D\theta]$ is provable, since the inequalities between w and $m_1, \ldots, m_k$ trivially true. Therefore we have that

$$\vdash \forall m_1 \cdots \forall m_k.[D\theta] \supset \nabla w.\mathit{lbisim}\ \mathtt{R}\ \mathtt{T}. \tag{15}$$

Now in order to prove sequent (14), we instantiate Q' with $\lambda w.\mathtt{T}$, and the rest of the proof proceeds as in the previous case, i.e., with the help of formula (15). □

B.6 "Early" open bisimulation

The proof of Theorem 24 is by induction on the number of input prefixes in P and Q. We prove a more general result: $\vdash \mathcal{Q}\bar{n}.\mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$ if and only if $\vdash \mathcal{Q}\bar{n}.\mathit{ebisim}\ \mathtt{P}\ \mathtt{Q}$, for any quantifier prefix $\mathcal{Q}\bar{n}$. By Lemma 38 and Lemma 39, and the invertibility of $\nabla\mathcal{R}$ and $\forall\mathcal{R}$ rules, we know that if $\vdash \mathcal{Q}\bar{n}.\mathit{lbisim}\ \mathtt{P}\ \mathtt{Q}$ and $\vdash \mathcal{Q}\bar{n}.\mathit{ebisim}\ \mathtt{P}\ \mathtt{Q}$, then their unfolded instances are also provable. We show that one can construct a derivation for one instance from the other. The non-trivial case is when the bound input transition is involved. That is, given a derivation of

$$\mathcal{Q}\bar{n}.[\forall X \forall P'.\mathtt{P} \xrightarrow{\downarrow X} P' \supset \forall w \exists Q'.\mathtt{Q} \xrightarrow{\downarrow X} Q' \wedge \mathit{ebisim}\ (P'w)\ Q'w)]$$

we can construct a derivation of

$$\mathcal{Q}\bar{n}.[\forall X \forall P'.\mathtt{P} \xrightarrow{\downarrow X} P' \supset \exists Q'.\mathtt{Q} \xrightarrow{\downarrow X} Q' \wedge \forall w.\mathit{ebisim}\ (P'w)\ (Q'w)]$$

and vice versa. Note that we cannot do any analysis on the universally quantified name w in both formulas, since we do not have any assumptions on names (*e.g.*, the excluded middle on names as in the adequacy theorem for late bisimulation). It is then easy to check that the choice of Q' in both cases is independent of the name w, and their correspondence follows straightforwardly from the induction hypothesis. □

C. ADEQUACY OF THE SPECIFICATIONS OF MODAL LOGICS

The completeness proof of the modal logics specification shares similar structures with the completeness proofs for specifications of bisimulation. In particular, we use an analog of Lemma 40, given in the following.

LEMMA 47. *Let* P *be a process and* A *an assertion such that* $\mathtt{P} \models \mathtt{A}$. *Then*

$$\vdash \forall n_1 \cdots \forall n_k.\mathcal{X} \wedge n_1 \neq \cdots \neq n_k \supset [\![\mathtt{P} \models \mathtt{A}]\!]$$

for some $\mathcal{X} \subseteq_f \mathcal{E}$ *and some names* $n_1, \ldots, n_k$ *such that* $\mathrm{fn}(\mathtt{P}, \mathtt{A}) \subseteq \{n_1, \ldots, n_k\}$.

The proof of lemma proceeds by induction on the size of A. The crucial step is when its interpretation in $FO\lambda^{\Delta\nabla}$ contains universal quantification over names, e.g., when $\mathtt{A} = [a(y)]\mathtt{B}$. In this case, we again use the same technique as in the proof of Lemma 40, i.e., using the excluded middle assumptions on names to enumerate all possible instances of the judgments. More detailed proof can be found in the electronic appendix of this paper.

C.1 Proof of Theorem 26 (Adequacy of the modal logic encoding)

First consider proving the soundness part of this theorem. Suppose we have a derivation Π of $\cdot\,;\,\mathcal{X} \vdash \nabla\bar{n}.[\![\mathtt{P} \models \mathtt{A}]\!]$. We want to show that $\mathtt{P} \models \mathtt{A}$. This is proved by induction on the size of $\mathtt{A}$. The proof also uses the property of invertible rules and the fact that applications of the excluded middles in $\mathcal{X}$ in deriving the sequent can be permuted up over all the right introduction rules. The latter is a consequence of Lemma 37. We look at a couple of interesting cases involving bound input and bound output.

out:. Suppose $\mathtt{A}$ is $[\bar{x}(y)]\mathtt{B}$. We need to show that for every $\mathtt{P}'$ such that $\mathtt{P} \xrightarrow{\bar{x}(y)} \mathtt{P}'$, we have $\mathtt{P}' \models \mathtt{B}$. (By α-conversion we can assume without loss of generality that y is not free in $\mathtt{P}$ and $\mathtt{A}$.) Note that here the occurrence of y in $\mathtt{P}'$ is bound in the transition judgment $\mathtt{P} \xrightarrow{\bar{x}(y)} \mathtt{P}'$. By Lemma 37 and the invertibility of certain inference rules, we can show that provability of $\cdot\,;\,\mathcal{X} \vdash \nabla\bar{n}.[\![\mathtt{P} \models \mathtt{A}]\!]$ implies the existence of a derivation Π' of

$$M\,;\,\mathcal{X}, \bar{n} \triangleright [\![\mathtt{P}]\!] \xrightarrow{\uparrow x} M\bar{n} \vdash \bar{n} \triangleright \nabla y.M\bar{n}y \models [\![\mathtt{B}]\!]$$

for some eigenvariable M. By the adequacy of one-step transitions, we have that

$$\vdash \nabla\bar{n}.[\![\mathtt{P}]\!] \xrightarrow{\uparrow x} \lambda y.[\![\mathtt{P}']\!].$$

Let θ be the substitution $[(\lambda\bar{n}\lambda y.[\![\mathtt{P}']\!])/M]$. Applying θ to Π' we get the derivation $\Pi'\theta$ of $\cdot\,;\,\bar{n} \triangleright [\![\mathtt{P}]\!] \xrightarrow{\uparrow x} \lambda y.[\![\mathtt{P}']\!] \vdash \bar{n} \triangleright \nabla y.\mathtt{P}' \models [\![\mathtt{B}]\!]$. By cutting this derivation with the one-step transition judgment above, we obtain a derivation of $\cdot\,;\,. \vdash \bar{n} \triangleright \nabla y.\mathtt{P}' \models [\![\mathtt{B}]\!]$. Hence by induction hypothesis, we have that $\mathtt{P}' \models \mathtt{B}$.

in:. Suppose $\mathtt{A}$ is $[x(y)]^L\mathtt{B}$. We show that there exists a process $\mathtt{P}'$ such that $\mathtt{P} \xrightarrow{x(y)} \mathtt{P}'$ and for all name w, $\mathtt{P}'[w/y] \models \mathtt{B}[w/y]$. It is enough to consider the case where w is a name in $\mathrm{fn}(\mathtt{P}, \mathtt{A})$ and the case where w is a new name not in $\mathrm{fn}(\mathtt{P}, \mathtt{A})$. By Lemma 37 and the invertibility of some inference rules, we can show that provability of $\cdot\,;\,\mathcal{X} \vdash \bar{n} \triangleright [\![\mathtt{P}]\!] \models [\![[x(y)]^L\mathtt{B}]\!]$ implies the existence of two derivations Π_1 and Π_2, of the sequents

$$\cdot\,;\,\mathcal{X} \vdash \bar{n} \triangleright \mathtt{P} \xrightarrow{\downarrow x} N \quad \text{and} \quad \cdot\,;\,\mathcal{X} \vdash \bar{n} \triangleright \forall y.Ny \models [\![\mathtt{B}]\!],$$

respectively, for some closed term N.

By the adequacy result in Proposition 9, there exists a process $\mathtt{P}'$ such that $[\![\mathtt{P}']\!] = Ny$ and $\mathtt{P} \xrightarrow{x(y)} \mathtt{P}'$. By Proposition 6, we can instantiate y with any of the free names occurring in $\mathtt{P}$ or $\mathtt{A}$ (since they are all in the list $\bar{n}$), and hence for any name $w \in \mathrm{fn}(\mathtt{P}, \mathtt{A})$ by induction hypothesis we get $\mathtt{P}'[w/y] \models \mathtt{B}[w/z]$. The case where w is a new name is dealt with as follows. Without loss of generality we assume that $y = w$ (since we can always choose y to be sufficiently fresh). From Π_2 it follows that $\vdash \mathcal{X} \supset \nabla\bar{n}.\forall y.[\![\mathtt{P}']\!] \models [\![\mathtt{B}]\!]$. Using the $FO\lambda^{\Delta\nabla}$ theorems

$$(\nabla x \forall y.P) \supset \forall y \nabla x.P \quad \text{and} \quad (P \supset \forall z.Q) \supset \forall z(P \supset Q)$$

where z is not free in P, we can move the $\forall y$ quantification in $\mathcal{X} \supset \nabla\bar{n}.\forall y.[\![\mathtt{P}']\!] \models [\![\mathtt{B}]\!]$ to the outermost level and get another provable formula:

$$\forall y(\mathcal{X} \supset \nabla\bar{n}.[\![\mathtt{P}']\!] \models [\![\mathtt{B}]\!]).$$

We then apply Proposition 3, to turn $\forall y$ into ∇y, thus obtaining a derivation of $\nabla y(\mathcal{X} \supset \nabla\bar{n}.[\![\mathtt{P}']\!] \models [\![\mathtt{B}]\!])$, and by distributing ∇ over $\supset$, we get $(\nabla y.\mathcal{X}) \supset \nabla y\nabla\bar{n}.[\![\mathtt{P}']\!] \models [\![\mathtt{B}]\!]$. We can now apply the induction hypothesis to get $\mathtt{P}' \models \mathtt{B}$.

Next we consider proving the completeness part of Theorem 26. Given $\mathtt{P} \models \mathtt{A}$, we would like to show that $\cdot; \mathcal{X} \vdash \nabla\bar{n}.[\![\mathtt{P} \models \mathtt{A}]\!]$ is provable. By Lemma 47, there are $m_1, \ldots, m_k$ and $\mathcal{X}'$ such that

$$\vdash \forall m_1 \cdots \forall m_k.\mathcal{X}' \wedge m_1 \neq m_2 \cdots \neq m_k \supset [\![\mathtt{P} \models \mathtt{A}]\!].$$

Let $\bar{n} = m_1, \ldots, m_k$ and let $\mathcal{X} = \nabla\bar{n}.\mathcal{X}'$. By Proposition 3, we have a derivation of

$$\nabla m_1 \cdots \nabla m_k.\mathcal{X}' \wedge m_1 \neq m_2 \cdots \neq m_k \supset [\![\mathtt{P} \models \mathtt{A}]\!].$$

By distributing the ∇'s over implication and conjunction we obtain

$$(\mathcal{X}) \wedge (\nabla\bar{n}.m_1 \neq m_2 \cdots \neq m_k) \supset \nabla\bar{n}.[\![\mathtt{P} \models \mathtt{A}]\!].$$

But since $\nabla\bar{n}.m_1 \neq m_2 \cdots \neq m_k$ is provable, by cut we obtain a derivation of

$$; \mathcal{X} \vdash \nabla\bar{n}.[\![\mathtt{P} \models \mathtt{A}]\!].$$

□

D. CHARACTERISATION OF OPEN BISIMULATION

LEMMA 48. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes. If for all* $\mathtt{A} \in \mathcal{LM}$, $\vdash (\mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A})$ *if and only if* $\vdash (\mathcal{Q}\bar{n}.\mathtt{Q} \models \mathtt{A})$, *where* $\mathrm{fn}(\mathtt{P}, \mathtt{Q}, \mathtt{A}) \subseteq \{\bar{n}\}$, *then* $\mathtt{P} \sim_o^D \mathtt{Q}$, *where* D *is the* $\mathcal{Q}\bar{n}$*-distinction.*

PROOF. Let $\mathcal{S}$ be the following family of relations

$$\mathcal{S}_D = \{(\mathtt{P}, \mathtt{Q}) \mid \text{ for all } \mathtt{A}, \vdash (\mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A}) \text{ iff } \vdash (\mathcal{Q}\bar{n}.\mathtt{Q} \models \mathtt{A}), \text{ where } \mathrm{fn}(\mathtt{P}, \mathtt{Q}, \mathtt{A}) \subseteq \{\bar{n}\} \text{ and } D \text{ is the } \mathcal{Q}\bar{n}\text{-distinction}\}$$

We then show that $\mathcal{S}$ is an open bisimulation. $\mathcal{S}$ is obviously symmetric, so it remains to show that it is closed under one-step transitions. We show here a case involving bound output; the rest are treated analogously.

Suppose $(\mathtt{P}, \mathtt{Q}) \in \mathcal{S}_D$. Then we have that for all $\mathtt{A}$, $\vdash \mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A}$ iff $\vdash \mathcal{Q}\bar{n}.\mathtt{Q} \models \mathtt{A}$, for some prefix $\mathcal{Q}\bar{n}$. Let θ be a substitution that respects D. Suppose $\mathtt{P}\theta \xrightarrow{\bar{x}(y)} \mathtt{P}'$. We need to show that there exists a $\mathtt{Q}'$ such that $\mathtt{Q}\theta \xrightarrow{\bar{x}(y)} \mathtt{Q}'$ and $\mathtt{P}' \sim_o^{D'} \mathtt{Q}'$ where $D' = D\theta \cup \{y\} \times \mathrm{fn}(\mathtt{P}, \mathtt{Q}, D)$. (Here we assume w.l.o.g. that y is chosen to be sufficiently fresh.) Suppose θ identifies the following pairs of names in $\mathtt{P}$ and $\mathtt{Q}$: $(x_1, y_1), \ldots, (x_k, y_k)$, and suppose that $\theta(z) = x$. Then by the definition of $\mathcal{S}_D$:

$$\vdash \mathcal{Q}\bar{n}.\mathtt{P} \models [x_1 = y_2][x_2 = y_2] \cdots [x_k = y_k]\langle\bar{z}(y)\rangle\mathtt{B}$$

if and only if

$$\vdash \mathcal{Q}\bar{n}.\mathtt{Q} \models [x_1 = y_2][x_2 = y_2] \cdots [x_k = y_k]\langle\bar{z}(y)\rangle\mathtt{B}$$

for all $\mathtt{B}$.

Note that the statement cannot hold vacuously, since for at least one instance of $\mathtt{B}$, i.e., $\mathtt{B}$ = true, both judgments must be true. By analysis on the (supposed) cut-free proofs of both judgments, for any $\mathtt{B}$, the above statement reduces to

$$\vdash \mathcal{Q}\bar{m}.\mathtt{P}\theta \models \langle \bar{x}(y)\rangle \mathtt{B}\theta \qquad \text{iff} \qquad \vdash \mathcal{Q}\bar{m}.\mathtt{Q}\theta \models \langle \bar{x}(y)\rangle \mathtt{B}\theta,$$

for some prefix $\mathcal{Q}\bar{m}$ such that $\mathcal{Q}\bar{m}$-distinction is the result of applying θ to the $\mathcal{Q}\bar{n}$-distinction.

Now let $\{\mathtt{Q}_i\}_{i\in I}$ be the set of all $\mathtt{Q}'$ such that $\mathtt{Q}\theta \xrightarrow{\bar{x}(y)} \mathtt{Q}'$, and suppose that for all $i \in I$, $\mathtt{P}' \not\sim_o^{D'} \mathtt{Q}_i$. That means that there exists an $\mathtt{A}_i$, for each $i \in I$, that separates $\mathtt{P}'$ and $\mathtt{Q}_i$, i.e., $\vdash (\mathcal{Q}\bar{m}\nabla y.\mathtt{P}' \models \mathtt{A}_i)$ but $\not\vdash (\mathcal{Q}\bar{m}\nabla y.\mathtt{Q}' \models \mathtt{A}_i)$. Note that we can assume w.l.o.g. that $\bar{m}$ include all the free names of $\mathtt{A}_i$ (recall that $\bar{n}$ is really a schematic list of names, dependent on the choice of $\mathtt{A}$ in the first place). Let $\mathtt{B}\theta$ be $\bigwedge_{i\in I} \mathtt{A}_i$. Then, by analysis of cut-free proofs, we can show that $\vdash (\mathcal{Q}\bar{m}.\mathtt{P}\theta \models \langle \bar{x}(y)\rangle \mathtt{B}\theta)$ but $\not\vdash (\mathcal{Q}\bar{m}.\mathtt{Q} \models \langle \bar{x}(y)\rangle \mathtt{B}\theta)$, which contradicts our initial assumption. Therefore, there must be one $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{\bar{x}(y)} \mathtt{Q}'$ and $\mathtt{P}' \sim_o^{D'} \mathtt{Q}'$. □

Lemma 49. *Let* $\mathtt{P}$ *and* $\mathtt{Q}$ *be two processes such that* $\mathtt{P} \sim_o^D \mathtt{Q}$ *for some distinction* D*. Then for all* $\mathtt{A} \in \mathcal{LM}$ *and for all prefix* $\mathcal{Q}\bar{n}$ *such that* D *corresponds to the* $\mathcal{Q}\bar{n}$*-distinction and* $\mathrm{fn}(\mathtt{P}, \mathtt{Q}, D) \subseteq \{\bar{n}\}$*,* $\vdash \mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A}$ *if and only if* $\vdash \mathcal{Q}\bar{n}.\mathtt{Q} \models \mathtt{A}$.

Proof. Suppose that $\mathtt{P} \sim_o^D \mathtt{Q}$ and $\vdash \mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A}$. We show, by induction on the size of $\mathtt{A}$, that $\vdash \mathcal{Q}\bar{n}.\mathtt{Q} \models \mathtt{A}$. The other direction is proved symmetrically, since open bisimulation is symmetric. We look at the interesting cases:

—Suppose $\mathtt{A} = \langle \bar{x}(y)\rangle \mathtt{B}$ for some $\mathtt{B}$. By analysis on the cut free derivations of $\mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A}$, it can be shown that

$$\vdash \mathcal{Q}\bar{n}.\exists M.\mathtt{P} \xrightarrow{\uparrow x} M \wedge \nabla y.(M\, y) \models \mathtt{B}.$$

This entails that there exists a process $\mathtt{P}'$ such that

$$\vdash \mathcal{Q}\bar{n}.\mathtt{P} \xrightarrow{\uparrow x} \lambda y.\mathtt{P}' \wedge \nabla y.\mathtt{P}' \models \mathtt{B}.$$

And by the invertibility of the right-introduction rules for $\forall$, ∇ and $\wedge$, this in turn entails that $\vdash \mathcal{Q}\bar{n}.\mathtt{P} \xrightarrow{\uparrow x} \lambda y.\mathtt{P}'$ and $\vdash \mathcal{Q}\bar{n}\nabla y.\mathtt{P}' \models \mathtt{B}$. The former implies, by the adequacy of one-step transition, that $\mathtt{P} \xrightarrow{\bar{x}(y)} \mathtt{P}'$. Since $\mathtt{P} \sim_o^D \mathtt{Q}$, this means that there exists $\mathtt{Q}'$ such that $\mathtt{Q} \xrightarrow{\bar{x}(y)} \mathtt{Q}'$ and $\mathtt{P}' \sim_o^{D'} \mathtt{Q}'$, where $D' = D \cup \{y\} \times \mathrm{fn}(\mathtt{P}, \mathtt{Q}, D)$. At this point we are almost ready to apply the induction hypothesis to $\mathcal{Q}\bar{n}\nabla y.\mathtt{P}' \models \mathtt{B}$, except that D' may not corresponds to the $\mathcal{Q}\bar{n}\nabla y$-distinction, since the latter may contain more inequal pairs than D'. However, since open bisimulation is closed under extensions of distinctions (see Lemma 6.3. in [Sangiorgi 1996]), we can assume without loss of generality that D' is indeed the $\mathcal{Q}\bar{n}\nabla y$-distinction. Therefore by the adequacy of one-step transition and induction hypothesis, we conclude that $\vdash \mathcal{Q}\bar{n}.\mathtt{Q} \xrightarrow{\uparrow x} \lambda x.\mathtt{Q}'$ and $\vdash \mathcal{Q}\bar{n}\nabla y.\mathtt{Q}' \models \mathtt{B}$, and from these, it follows that $\mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A}$ is also provable.

—Suppose $\mathtt{A} = \langle x(y)\rangle \mathtt{B}$. This case is analogous to the previous case. The only difference is that the bound input is universally quantified, instead of ∇-quantified. So we apply the induction hypothesis to $\mathcal{Q}\bar{n}\forall y.\mathtt{P}' \models \mathtt{B}$, which can be done without resorting to extensions of the distinction D, since in this case the $\mathcal{Q}\bar{n}\forall y$-distinction is exactly D.

—For the cases where $\mathtt{A}$ is prefixed by either $[x(y)]^L$ or $[\bar{x}(y)]$, the proof follows a similar argument as in the completeness proof of open bisimulation (Theorem 22). For instance, for the case where $\mathtt{A} = [x(y)]^L\mathtt{B}$, from the fact that $\vdash \mathcal{Q}\bar{n}.\mathtt{P} \models \mathtt{A}$, it follows that

$$\vdash \mathcal{Q}\bar{n}.\forall M(\mathtt{P} \xrightarrow{\downarrow x} M \supset \exists y.(M\, y) \models \mathtt{B}).$$

As in the proof of Theorem 22, we can further show that there is a derivation of this formula that ends with one_b-rule, such that every θ in this premise is a D-respecting substitution. Since $\mathtt{P} \sim_o^D \mathtt{Q}$, we can show that every bound input action of $\mathtt{P}\theta$, for any D-respecting θ, can be imitated by $\mathtt{Q}\theta$ and vice versa. From this and induction hypothesis, we can therefore obtain a derivation of

$$\mathcal{Q}\bar{n}.\forall N(\mathtt{Q} \xrightarrow{\downarrow x} N \supset \exists y.(N\, y) \models \mathtt{B}),$$

hence $\vdash \mathcal{Q}\bar{n}.\mathtt{Q} \models \mathtt{A}$. □

Finally, the proof of Theorem 27 now follows immediately from Lemma 48 and Lemma 49. □

Received Month Year; revised Month Year; accepted Month Year