Model Checking Mobile Processes[1]

Mads Dam[2]
Swedish Institute of Computer Science
Box 1263
S-164 28 Kista
Sweden

[1]Preliminary version published as "Model Checking Mobile Processes", Lecture Notes in Computer Science 715 (1993) pp. 22-36

[2]Work supported by ESPRIT BRA project 6454 "CONFER"

Abstract

We introduce a temporal logic for the polyadic π-calculus based on fixed point extensions of Hennessy-Milner logic. Features are added to account for parametrisation, generation, and passing of names, including the use, following Milner, of dependent sum and product to account for (unlocalised) input and output, and explicit parametrisation on names using lambda-abstraction and application. The latter provides a single name binding mechanism supporting all parametrisation needed. A proof system and decision procedure is developed based on Stirling and Walker's approach to model checking the modal μ-calculus using constants. One difficulty, for both conceptual and efficiency-based reasons, is to avoid the explicit use of the ω-rule for parametrised processes. A key idea, following Hennessy and Lin's approach to deciding bisimulation for certain types of value-passing processes, is the relativisation of correctness assertions to conditions on names. Based on this idea a proof system and decision procedure is obtained for arbitrary π-calculus processes with finite control, π-calculus correlates of CCS finite-state processes, avoiding the use of parallel composition in recursively defined processes.

1 Introduction

The propositional μ-calculus has recently emerged as a powerful instrument for specifying temporal properties of processes (c.f. [17, 4]), and model checkers for checking propositional μ-calculus properties against finite-state (CCS) processes have been developed and implemented (c.f. [8, 18, 2]). For most practical applications, however, mechanisms for parameter passing and quantification are invaluable. Based on CCS the π-calculus of Milner, Parrow, and Walker [13] has recently been proposed as a way of formally describing mobility in process structures such as mobile telephone networks (Orava, Parrow [14]). In fact the π-calculus can well be viewed as a prototypical value passing calculus, a view being reinforced by the capacity of the π-calculus to encode data types [10], lambda calculus [11], and higher order processes [15].

As a temporal logic for the π-calculus, however, the propositional μ-calculus is not directly suitable, lacking, as it does, mechanisms for parametrisation, passing, generation, and quantification of names. In this paper we demonstrate

1. how such facilities can be added to the propositional μ-calculus, resulting in a very expressive temporal logic for the π-calculus, and

2. how a proof system and tableau based model checking algorithm for this richer logic can be built, based, concretely, on Stirling and Walker's approach to model checking the modal μ-calculus [18].

Note that (2) is far from trivial, since there is no prior reason to believe that the mechanisms for parameter handling and those for fixed points do not interfere. Indeed the contrary, if anything, should be supposed, since name passing causes even the simplest processes to be infinite state.

A number of problems must be addressed. The first concerns the choice of base modalities. Our work is based on the logic of Milner [10] for the polyadic π-calculus, an extension of the π-calculus to support the communication of tuples. A key feature of this logic is the use of dependent sum (Σ) and product (here $\forall$) to handle (un-localised) output and input of name-parameters.

The second hurdle concerns the need for fixed points to be parametrised on names. To see the necessity of this consider the following single element memory cell (in CCS-like notation)

$$\mathrm{MEM}(x) \triangleq \overline{\mathrm{out}}x.\mathrm{MEM}(x) + \mathrm{in}(y).\mathrm{MEM}(y)$$

A characteristic property of $\mathrm{MEM}(x)$ is, informally, that it always outputs the last element input, or, rephrased without reference to pasttime modalities, that whenever an element is input then that same element is output until some new element is input. Trying to formalise this property using the ideas of the propositional μ-calculus results in the following parametrised fixed point

$$\phi = \nu X(x).[\mathrm{in}(y)]X(y) \wedge [\overline{\mathrm{out}}x'](x = x' \wedge X(x')).$$

This example illustrates the extent to which name-parametrisation pervades the syntax of formulas. By using explicit name-parametrisation and instantiation by λ-abstraction and application all parametrisation needed can be handled by a single name-binding mechanism. Thus, as an example, we replace ϕ by the formula

$$\nu X.\lambda x.[\mathrm{in}]\forall(\lambda y.(Xy)) \wedge [\overline{\mathrm{out}}]\Sigma(\lambda x'.x = x' \wedge (Xx')).$$

In this manner a large degree of orthogonality is revealed between propositional connectives, modal connectives, fixed points, abstraction and application, and quantifiers.

A third hurdle concerns the doubling of names in the π-calculus as both variables and constants. This makes a standard version of the rule of generalisation for correctness assertions $A : \phi$ such as

$$\forall\text{-INTRO:} \quad \frac{Ax : \phi x}{A : \forall\phi} \quad (x \text{ not free in } A \text{ or } \phi)$$

unsound. For instance it will license the inference

$$\frac{y.\mathbf{0} \mid \overline{z}.\mathbf{0} : [\tau]\mathrm{false}}{(\lambda x)(y.\mathbf{0} \mid \overline{x}.\mathbf{0}) : \forall\lambda x.[\tau]\mathrm{false}}$$

which is clearly invalid. An alternative is to use an ω-rule for $A : \forall\phi$, perhaps restricted to names free in A or ϕ plus one to serve as a representative of names free in neither. While sound, such an approach, however, has some disadvantages: Its schematic form makes it somewhat unattractive from a proof-theoretic point of view, but more seriously it is inefficient, forcing names to be treated distinctly even where this may not be necessary. An alternative which has been pursued in the context of value-passing calculi by Hennessy and Lin [5] for bisimulation checking, and by Hennessy and Liu [6] for modal logics, is to explicitly relativise correctness assertions to conditions c on names. Such name conditions are expressions in the first-order language of names with equality. The problem with $\forall$-INTRO is that by taking x to be fresh it is thereby implicitly assumed to be distinct from all names that are not fresh. If relativised correctness assertions are written $c \vdash A : \phi$ the rule of generalisation is regained in the following form:

$$\text{RELATIVISED-}\forall\text{-INTRO:} \quad \frac{c \vdash Ax : \phi x}{c \vdash A : \forall\phi} \quad (x \text{ not free in } A,\ \phi, \text{ or } c)$$

where by requiring x to be not free in c ensuring that no prior assumptions about x are made neither explicitly nor implicitly. Name conditions are expressions in the first-order language of names with equality.

A fourth hurdle concerns model checking and how to deal with fixed points. We adopt the approach of Stirling and Walker [18] using constants to keep track of the way fixed point occurrences are unfolded during model checking. The use of constants allows alternating fixed points, crucial for the expression of many liveness and fairness related properties, to be handled in an elegant fashion. The approaches to model checking in the propositional μ-calculus applies only to finite-state processes. For the π-calculus restricting to true finite-state processes is far too restrictive since even the simplest π-calculus processes exhibit infinite-state behaviour. A much more liberal notion is obtained as a direct generalisation of the notion of finite-state process in CCS by disallowing just processes which have occurrences of the parallel combinator $|$ within recursive definitions. Processes which adhere to this restriction are termed *finite control*. What is surprising is that this condition turns out to be the only one needed for model checking to work and be decidable. We present as the main result of the paper a proof-, or tableau system for relativised correctness assertions for finite control processes which is sound and complete, and use it as the basis for a decision procedure.

In sections 2 and 3 we present our version of the polyadic π-calculus and its operational semantics. In order to support the relativisation of correctness assertions to name conditions the operational semantics is modified by similarly relativising the structural congruence and commitment relations to name partitions. These are partitions of the name spaces determining the identifications and distinctions assumed. Distinctions alone, as introduced by Milner et al in [13], are too weak since both positive and negative assertions about the identity of names are needed. Interestingly, name partitions provides machinery to include into the polyadic π-calculus the conditional bAB where b is a boolean expression, behaving like A when b is true and like B when b is false. In section 4 the extended μ-calculus is introduced, and in section 5 the proof system for relativised correctness assertions is given. The remainder of the paper are devoted to proofs of soundness, completeness, and decidability of this proof system. These proofs extend corresponding proofs for the modal μ-calculus due to Stirling and Walke [18], and Streett and Emerson [19]. In section 6 an alternative semantics, called *symbolic* following Hennessy and Lin [5], of the extended μ-calculus is given which relativise formulas to general name conditions rather than just name partitions. The symbolic semantics provides to a large extent the purely local parts of the soundness, completeness, and decidability proofs. Soundness is proved in section 7 and the decision procedure is given in section 8. In section 9 the decision procedure is proved terminating and well-defined, and then completeness and decidability is proved in section 10. Finally section 11 contains the conclusion and discussions of related work.

2 The Polyadic π-calculus

The version of the π-calculus used here is a version of Milner's polyadic π-calculus [10], somewhat modified to involve conditionals and an operational semantics relativised to name partitions. The letters $x, y, z, \ldots$ are used to range over *names* of which there is a countably infinite supply, A, B over *agents*, and D over *agent identifiers*. *Actions*, α, β, are either names, co-names of the form $\overline{x}$, or the distinguished constant τ. We assume a countable infinity of distinct names. If α is a name x then $\mathrm{n}(\alpha)$ (the name of α) is x. and $\mathrm{p}(\alpha)$ (the polarity of α) is $-$. Otherwise if $\alpha = \overline{x}$ then $\mathrm{n}(\alpha) = x$ and $\mathrm{p}(x) = +$. The syntax of agents is given as follows:

Boolean expressions:

$$b ::= x = y \,\big|\, \neg b \,\big|\, b \wedge b$$

Agents:

$$A ::= \mathbf{0} \,\big|\, A + A \,\big|\, \alpha.A \,\big|\, A \mid A \,\big|\, bAA \,\big|\, (\lambda x)A \,\big|\, Ax \,\big|\, (\nu x)A \,\big|\, D \,\big|\, \mathrm{fix}D.A \,\big|\, [x]A$$

For most connectives the intended meaning is familiar from CCS and the π-calculus [9, 13]. Conditionals are agents of the form bAB, and (λx) and $[x]$ are used for unlocalised input and output, to be localised by a prefixing operator $\alpha.\cdot$. In CCS terms $x.(\lambda y)A$ is $x(y).A$ and $\overline{x}.[y]A$ is $\overline{x}y.A$. The restriction operator is ν. We use recursively defined agents rather than replication as in [10] as we are interested in the subcalculus of the polyadic π-calculus which arises from disallowing uses of $\mid$ in recursively defined agents, mirroring the notion of finite state process in CCS. Agents in this subcalculus are termed *finite control*. For technical reasons we assume that recursions $\mathrm{fix}D.A$ are *guarded* in the sense that each occurrence of D in A is within the scope of a prefix operator $\alpha.-$, and that they are *fully parametrised* in the sense that recursive agents $\mathrm{fix}D.A$ have no free occurrences of names. Furthermore we generally presuppose agents not to contain free occurrences of agent identifiers.

The syntax as given here is flat: No distinctions are made between processes, abstractions, and concretions as in [10]. To recover these distinctions we assign to well-formed agents A an integer *arity* n, written $A : n$. The set of all well-formed agents is denoted $\mathcal{A}$. *Processes* are agents of arity 0, *abstractions* are agents of negative arity, and *concretions* are agents of positive arity. The following assignment of arities is relative to an assignment $D : n$ of arities to agent identifiers:

$$\frac{}{\mathbf{0} : 0} \qquad \frac{A : 0 \quad B : 0}{A + B : 0} \qquad \frac{A : n \quad n \leq 0}{x.A : 0} \qquad \frac{A : n \quad n \geq 0}{\overline{x}.A : 0} \qquad \frac{A : 0}{\tau.A : 0}$$

$$\frac{A : 0 \quad B : 0}{A \mid B : 0} \qquad \frac{A : n \quad B : n}{bAB : n} \qquad \frac{A : n \quad n \leq 0}{(\lambda x)A : n-1} \qquad \frac{A : n-1 \quad n \leq 0}{Ax : n}$$

$$\frac{A : n}{(\nu x)A : n} \qquad \frac{D : n \quad A : n}{\mathrm{fix}D.A : n} \qquad \frac{A : n \quad n \geq 0}{[x]A : n+1}$$

Example 2.1 *The agent $(\mathit{fix}D.(\lambda x)(x.(\lambda y)Dy))x$ is a well-formed process under the assumption $D : -1$. The agents $x.(\lambda y)[y]\mathbf{0}$ and $x.(\lambda y)[y]\mathbf{0}$ are ill-formed.*

The operators $(\lambda x)A$ and $(\nu x)A$ introduce binding of the free occurrences of x and $\overline{x}$ in A. For an agent A, $\mathrm{fn}(A)$ is the set of names occurring freely in A, and $A\{y/x\}$ is A with all free occurrences of x substituted for y. In general this involves alpha-conversion of A to avoid capture of names.

1. $\equiv_\varepsilon$ is an equivalence relation preserved by all non-binding operators
2. If $A \equiv_{(\nu x)\varepsilon} B$ then $(\nu x)A \equiv_\varepsilon (\nu x)B$.
3. $A \equiv_\varepsilon B$ if A and B are alpha-convertible.
4. Abelian monoid laws for $+$ and $\mathbf{0}$, i.e. $A_1 + (A_2 + A_3) \equiv_\varepsilon (A_1 + A_2) + A_3$, $A_1 + A_2 \equiv_\varepsilon A_2 + A_1$, and $A + \mathbf{0} \equiv_\varepsilon A$.
5. Abelian monoid laws for $\mid$ and $\mathbf{0}$.
6. $bAB \equiv_\varepsilon (\neg b)BA$.
7. If $\varepsilon \models b$ then $bAB \equiv_\varepsilon A$.
8. $((\lambda x)A)y \equiv_\varepsilon A\{y/x\}$.
9. $\mathrm{fix}D.A \equiv_\varepsilon A\{\mathrm{fix}D.A/D\}$.
10. $(\nu x)\mathbf{0} \equiv_\varepsilon \mathbf{0}$, $(\nu x)(\nu y)A \equiv_\varepsilon (\nu y)(\nu x)A$, $(\nu x)(\nu x)A \equiv_\varepsilon (\nu x)A$.
11. If $x \notin \mathrm{fn}(B)$ then $((\nu x)A) \mid B \equiv_\varepsilon (\nu x)(A \mid B)$.
12. If $x \neq y$ then $(\nu y)(\lambda x)A \equiv_\varepsilon (\lambda x)(\nu y)A$ and $(\nu y)[x]A \equiv_\varepsilon [x](\nu y)A$.

Figure 1: Structural congruence relation

3 Operational Semantics

The operational semantics of agents is, following Milner [10], given in terms of a structural congruence relation $\equiv$ together with a commitment relation $\succ$. This style of semantics was introduced by Milner in [11] to which the reader is referred for justification of many of the clauses given below. Here the structural congruence and commitment relations are parametrised on *name partitions*, partitions ε on the set of names. This provides the strengthening of the notion of distinctions [13] needed to deal with general name conditions rather than just the positive match operator of [13]. A name partition ε identifies the names x and y if and only if x and y are members of the same partition. Thus name partitions provide models for boolean expressions and first-order conditions on names, and we write $\varepsilon \models c$ if ε is a model for c. Name partitions extend to actions in the obvious way by $\varepsilon \models \alpha_1 = \alpha_2$ iff either $\alpha_1 = x_1$, $\alpha_2 = x_2$, and $\varepsilon \models x_1 = x_2$; or $\alpha_1 = \overline{x_1}$, $\alpha_2 = \overline{x_2}$, and $\varepsilon \models x_1 = x_2$; or $\alpha_1 = \alpha_2 = \tau$. In addition to interpreting booleans and first-order name conditions we need an operation for the generation of new names:

$$(\nu x)\varepsilon = \{S - \{x\} \mid S \in \varepsilon\} \cup \{\{x\}\}.$$

The conditions governing the relativised structural congruence relation $\equiv_\varepsilon$ are shown on fig. 1. Note that for the structural congruence relation (but not for the commitment relation) relativisation to name partitions is needed only because of conditionals. An unrelativised structural congruence relation $\equiv$ can be derived from the relativised one by $A\varepsilon \equiv B\varepsilon$ whenever $A \equiv_\varepsilon B$. This congruence relation is closely related to the one considered by Milner in [10]. The difference is that we do not here in general assume conversion under λ, i.e. a rule such as

$$\text{If } A \equiv_{\varepsilon'} B \text{ for all } \varepsilon' \text{ such that } \{S - \{x\} \mid S \in \varepsilon'\} = \{S - \{x\} \mid S \in \varepsilon\} \text{ then}$$
$$(\lambda x)A \equiv_\varepsilon (\lambda x)B.$$

Thus the term "congruence" for the structural congruence relation is actually misplaced, and for the remainder of the paper we refer to $\equiv_\varepsilon$ as the structural *equivalence* relation instead.

Another justification for $\equiv_\varepsilon$ is in terms of an appropriate normal form theorem. Say an agent A is in *normal form* if it is either an abstraction of the form $(\lambda x)A$, a concretion of the form $[x]A$ or $(\nu x)[x]A$, or a process P generated by the abstract syntax

$$P ::= \mathbf{0} \,\Big|\, P + P \,\Big|\, \alpha.A \,\Big|\, P \mid P \,\Big|\, (\nu x)P$$

Proposition 3.1 (Normal forms) *Given any well-formed agent A and any name partition ε there is a normal form B such that $A \equiv_\varepsilon B$.*

Proof We prove a somewhat more general statement. Say that A is ε-*admissible*, if

1. A is well-formed,
2. there is a normal form B such that $A \equiv_\varepsilon B$, and
3. if $A : n$ and $n < 0$ then for all x, Ax is ε-admissible.

We show for all well-formed agents A and all name partitions ε that A is ε-admissible. First we need to show that both arities and ε-admissibility is preserved by structural equivalence.

Lemma 3.2 *Let A be any agent.*

1. *If $A : n$ and $A \equiv_\varepsilon B$ then $B : n$.*
2. *If A is ε-admissible and $A \equiv_\varepsilon B$ then B is ε-admissible.*

Proof 1: An easy induction in the structure of proof of $A \equiv_\varepsilon B$. 2: Induction in $|n|$ where n is the arity of A, using 1. □ (Lemma 3.2)

Let now ε' be any name partition and A any well-formed agent. A is allowed to contain free guarded occurrences of identifiers, and identifiers are assumed to be assigned an arity. We use induction in the structure of A to show that if A' is any instance of A obtained by substituting names for names and agents of arity n for free guarded occurrences of identifiers of arity n then A' is ε-admissible thus completing the proof. We consider the cases for the conditional, lambda abstraction, application, and recursive definition. The remaining cases are similar.

$A = bA_1A_2$. Either $\varepsilon \models b$ or $\varepsilon \models \neg b$. Assume without loss of generality the first. Then $A' \equiv_\varepsilon A'_1$ where A'_1 is the corresponding substitution instance of A_1. By the induction hypothesis A'_1 is ε-admissible. By Lemma 3.2.2 so is A'.

$A = (\lambda x)B$. A is well-formed by assumption, and A is in normal form. Let y be any name. By the induction hypothesis $B'\{y/x\}$ is ε-admissible where B' is the appropriate substitution instance of B. Then by Lemma 3.2.2 $(A')y$ is ε-admissible too. Thus A' is ε-admissible.

$A = Bx$. By the induction hypothesis B' is ε-admissible where B' is the expected substitution instance of B. Then by definition so is A'.

$A = \mathrm{fix}D.B$. Since A is well-formed by assumption, and all occurrences of D in B are guarded, B' is ε-admissible where B' is the substitution instance of B that corresponds to A', and which substitutes A for D. But then by Lemma 3.2.2 A' is also ε-admissible. $\square$ (Lemma 3.1)

In fact the proof of Proposition 3.1 can be used to show that B can be found of size not greater than that of A where size is measured in e.g. depth of parse tree.

We proceed to define the relativised commitment relation $A \succ_\varepsilon \alpha.B$. The definition uses the operation of pseudo-application, and the extension of parallel composition to pairs of abstractions and concretions as in [10]. The pseudo-application of A to B, $A \cdot B$, is defined only when $A : -n$ and $B : n$ for some (positive or negative) n. If $n = 0$ then $A \cdot B = A \mid B$. If $n > 0$, $A = (\lambda x)A'$, and $B = [y]B'$ then $A \cdot B = A'\{y/x\} \cdot B'$, and if instead $B = (\nu y)[y]B'$ then $A \cdot B = (\nu y)(A'\{y/x\} \cdot B')$. The case for $n < 0$ is defined symmetrically. Secondly $A \mid B$ is extended to the case when only one of A, B is a process by (in case B is a process) $((\lambda x)A) \mid B = (\lambda x)(A \mid B)$ where $x \notin \mathrm{fn}(B)$, $([x]A) \mid B = [x](A \mid B)$, and $((\nu x)[x]A) \mid B = (\nu x)[x](A \mid B)$ where $x \notin \mathrm{fn}(B)$. The case for A is defined symmetrically.

The commitment relation is now given in fig. 2. Note that although this is not necessary since $\mid$ is assumed to be commutative, we have chosen to include symmetrical versions of the rules SUM, COMM and PAR. This is merely a technical convenience. As for the structural equivalence relation we can derive an unrelativised version by $A\varepsilon \succ (\alpha.B)\varepsilon$ whenever $A \succ_\varepsilon \alpha.B$. In the absence of conditionals, $\succ$ is exactly the commitment relation of [10].

4 Adding Name Passing to the Propositional μ-calculus

In this section we extend the propositional μ-calculus with name-parametrisation and dependent sum and product as in [10]. The result is a powerful temporal logic for the polyadic π-calculus characterising late strong bisimulation equivalence [10, 12]. By explicitly introducing lambda-abstraction and application of names all parametrisation issues for fixed points and dependent types are catered for in a uniform way. Formulas, ranged over by ϕ, ψ, are thus interpreted as sets of agents parametrised on names. The letters X, Y, Z range over propositional variables each assigned an *arity* $n \in \omega$, written $X : n$. The syntax of formulas is given as

$$\textsc{act}: \quad \frac{}{\alpha.A \succ_\varepsilon \alpha.A} \qquad \textsc{sum}: \quad \frac{A_1 \succ_\varepsilon B}{A_1 + A_2 \succ_\varepsilon B}$$

$$\textsc{comm}: \quad \frac{A_1 \succ_\varepsilon x.B_1 \quad A_2 \succ_\varepsilon \overline{y}.B_2}{A_1 \mid A_2 \succ_\varepsilon \tau.(B_1 \cdot B_2)} \quad (\varepsilon \models x = y)$$

$$\textsc{par}: \quad \frac{A_1 \succ_\varepsilon \alpha.B}{A_1 \mid A_2 \succ_\varepsilon \alpha.(B \mid A_2)} \qquad \textsc{res-1}: \quad \frac{A \succ_{(\nu x)\varepsilon} \tau.B}{(\nu x)A \succ_\varepsilon \tau.(\nu x)B}$$

$$\textsc{res-2}: \quad \frac{A \succ_{(\nu x)\varepsilon} \alpha.B}{(\nu x)A \succ_\varepsilon \alpha.(\nu x)B} \quad (x \neq \mathrm{n}(\alpha))$$

$$\textsc{struct}: \quad \frac{A_1 \equiv_\varepsilon A_2 \quad A_2 \succ_\varepsilon \alpha.B_1 \quad B_1 \equiv_\varepsilon B_2}{A_1 \succ_\varepsilon \alpha.B_2}$$

+ symmetrical versions of rules SUM, COMM and PAR

Figure 2: Commitment relation

follows:

$$\begin{array}{rcl} \phi & ::= & x = y \;\big|\; x \neq y \;\big|\; \phi \wedge \phi \;\big|\; \phi \vee \phi \;\big|\; <\alpha>\phi \;\big|\; [\alpha]\phi \;\big| \\ & & X \;\big|\; \nu X.\phi \;\big|\; \mu X.\phi \;\big|\; \lambda x.\phi \;\big|\; \phi x \;\big|\; \Sigma\phi \;\big|\; \forall\phi \;\big|\; \exists\phi \end{array}$$

Briefly the logical connectives can be understood as follows: $\wedge$ and $\vee$ are the usual boolean connectives; $<\alpha>$ and $[\alpha]$ are the labelled modal connectives; ν (not to be confused with the π-calculus ν-operator) is the greatest fixed point operator used, typically, for invariant properties; μ is the least fixed point operator used for eventualities; λ and application is used for name-parametrisation; Σ is dependent sum used for concretions, for instance $\Sigma\phi$ is satisfied by a concretion $[x]A$ for which A satisfies ϕx; and finally $\forall$ and $\exists$ are quantifiers expressing properties of abstractions. For instance $\forall\phi$ is satisfied by an abstraction A for which Ax satisfies ϕx for all x, and $\exists\phi$ is satisfied by an abstraction A for which Ax satisfies ϕx for some x. Thus the logical correlate of (agent) abstraction is quantification. We use σ as a meta-variable ranging over $\{\nu, \mu\}$. As for agents we assume for technical convenience that recursive (ν or μ) formulas have no free occurrences of names. The only binder of names is λ, and ν and μ are binders of propositional variables. Formulas are generally identified up to renaming of bound names or variables.

As for the π-calculus attention is restricted to well-formed formulas by extending the assignment of arities to variables to arbitrary well-formed formulas by letting $x = y : 0$, $x \neq y : 0$, and closing under the rules:

$$\frac{\phi : 0 \quad \psi : 0}{\phi \wedge \psi : 0} \qquad \frac{\phi : 0 \quad \psi : 0}{\phi \vee \psi : 0} \qquad \frac{\phi : 0}{<\alpha>\phi : 0} \qquad \frac{\phi : 0}{[\alpha]\phi : 0}$$

$$\frac{X : n \quad \phi : n}{\sigma X.\phi : n} \qquad \frac{\phi : n}{\lambda x.\phi : n+1} \qquad \frac{\phi : n+1}{\phi x : n}$$

$$\frac{\phi : n+1}{\Sigma\phi : n} \quad \frac{\phi : n+1}{\forall\phi : n} \quad \frac{\phi : n+1}{\exists\phi : n}$$

A simple generalisation is to extend nonzero arities to boolean and modal formulas by pointwise extensions as for instance for conjunction:

$$\frac{\phi : n \quad \psi : n}{\phi \wedge \psi : n}$$

No expressive power is gained by this modification.

We proceed to define the semantics of formulas. First machinery is introduced to account for free occurrences of propositional variables. A *proposition environment* is a mapping ρ which given a propositional variable X of arity m, an m-vector of names $y_1, \ldots, y_m$, and a name partition ε gives a set $\rho X y_1 \cdots y_m \varepsilon \subseteq \mathcal{A}$. Let now $\phi : n$. Given a proposition environment ρ, an n-vector $x_1 \cdots x_n$ of names, and a name partition ε, the "standard" interpretation of ϕ produces a set $\|\phi\|\rho x_1 \cdots x_n \varepsilon \subseteq \mathcal{A}$. If ϕ does not contain free occurrences of propositional variables then ϕ is said to be *propositionally closed*. For such ϕ, $\|\phi\|\rho x_1 \cdots x_n \varepsilon$ does not depend on ρ and is thus abbreviated $\|\phi\| x_1 \cdots x_n \varepsilon$. The standard interpretation is shown in fig. 3. Here the complete boolean algebra structure of $2^{\mathcal{A}}$ is inherited pointwise to proposition environments and interpretations. The symbols $\sqsubseteq$, $\sqcap$, and $\sqcup$ are used to denote the induced lattice ordering, infimum, and supremum, respectively. Notice that for formulas in positive form (i.e. with negations applied to propositional variables only) the modal μ-calculus can be viewed as a sublanguage of the language considered here, and that the semantics assigned by fig. 3 to this sublanguage is the usual one (c.f. [18]).

5 Proof System

In this section we introduce a proof system for relativised correctness assertions $c \vdash A : \phi$. The intended interpretation of such assertions is that $A \in \|\phi\|\varepsilon$ whenever $\varepsilon \models c$. A complication, however, concerns the need to handle fixed point formulas. For this we adopt the approach of Stirling and Walker [18] by including into the syntax of formulas *constants* U to denote occurrences of fixed point formulas. A *definition list* is a sequence $\Delta = (U_1 \mapsto \phi_1), \ldots, (U_m \mapsto \phi_m)$, associating to each U_i the propositionally closed formula $\Delta(U_i) = \phi_i$. Here Δ is required to satisfy the conditions:

1. each U_i is unique, and
2. each $\Delta(U_i)$ mentions only constants among $\{U_1, \ldots, U_{i-1}\}$.

For Δ as above, $\mathrm{dom}(\Delta) \triangleq \{U_1, \ldots, U_m\}$, and if $U \notin \mathrm{dom}(\Delta)$ and each constant occurring in ϕ is included in $\mathrm{dom}(\Delta)$ then $\Delta \cdot (U \mapsto \phi)$ is the update of Δ associating ϕ to U. If Δ is *admissible for* ϕ in the sense that each constant occurring in ϕ

$$
\begin{array}{lcl}
\|x = y\|\rho\varepsilon & = & \begin{cases} \mathcal{A} & \text{if } \varepsilon \models x = y \\ \emptyset & \text{otherwise} \end{cases} \\
\|x \neq y\|\rho\varepsilon & = & \begin{cases} \mathcal{A} & \text{if } \varepsilon \models x \neq y \\ \emptyset & \text{otherwise} \end{cases} \\
\|\phi \wedge \psi\| & = & \|\phi\| \sqcap \|\psi\| \\
\|\phi \vee \psi\| & = & \|\phi\| \sqcup \|\psi\| \\
\|{<}\alpha{>}\phi\|\rho\varepsilon & = & \{A \mid \exists\beta, B.\ A \succ_\varepsilon \beta.B, \varepsilon \models \alpha = \beta, B \in \|\phi\|\rho\varepsilon\} \\
\|[\alpha]\phi\|\rho\varepsilon & = & \{A \mid \forall\beta, B.\ \text{if } A \succ_\varepsilon \beta.B \text{ and } \varepsilon \models \alpha = \beta \\
 & & \text{then } B \in \|\phi\|\rho\varepsilon\} \\
\|X\|\rho & = & \rho X \\
\|\nu X.\phi\|\rho & = & \sqcup\{f \mid f \sqsubseteq \|\phi\|\rho[X \mapsto f]\} \\
\|\mu X.\phi\|\rho & = & \sqcap\{f \mid \|\phi\|\rho[X \mapsto f] \sqsubseteq f\} \\
\|\lambda x.\phi\|\rho x_1 \cdots x_n\varepsilon & = & \|\phi\{x1/x\}\|\rho x_2 \cdots x_n\varepsilon \\
\|\phi x\|\rho x_1 \cdots x_n\varepsilon & = & \|\phi\|\rho x x_1 \cdots x_n\varepsilon \\
\|\Sigma\phi\|\rho x_1 \cdots x_n\varepsilon & = & \{A \mid A \equiv_\varepsilon [x]B, \text{ and } B \in \|\phi\|\rho x x_1 \cdots x_n\varepsilon\} \cup \\
 & & \{A \mid A \equiv_\varepsilon (\nu x)[x]B, x \notin \mathrm{fn}(\phi) \cup \{x_1, \ldots, x_n\}, \text{ and} \\
 & & B \in \|\phi\|\rho x x_1 \cdots x_n((\nu x)\varepsilon)\} \\
\|\forall\phi\|\rho x_1 \cdots x_n\varepsilon & = & \{A \mid \forall x.Ax \in \|\phi\|\rho x x_1 \cdots x_n\varepsilon\} \\
\|\exists\phi\|\rho x_1 \cdots x_n\varepsilon & = & \{A \mid \exists x.Ax \in \|\phi\|\rho x x_1 \cdots x_n\varepsilon\},
\end{array}
$$

Figure 3: Standard semantics

is in dom(Δ) then ϕ_Δ is constant-free formula resulting from recursively replacing each occurrence of a constant in ϕ by its definition. Note that, as fixed point formulas are required to be fully parametrised, formulas ϕ and ϕ_Δ have identical sets of free names.

Thus relativised correctness assertions, or *sequents*, have the form $c \vdash_\Delta A : \phi$ where A is a well-formed process, Δ is admissible for ϕ, and ϕ_Δ is propositionally closed and of arity 0. The sequent $c \vdash_\Delta A : \phi$ is then *true*, if $A \in \|\phi_\Delta\|\varepsilon$ whenever $\varepsilon \models c$. We present a proof, or tableau system for sequents. The proof system consists of a collection of axioms and proof rules which describe the local properties of the logical connectives, plus an additional rule to deal with properties which depend on the infinite behaviour of agents. The following abbreviations are used in the local proof rules shown in fig. 4:

1. α and β *c-match*: Either $\alpha = \beta = \tau$, or else $\models c \supset \mathrm{n}(\alpha) = \mathrm{n}(\beta)$, and $\mathrm{p}(\alpha) = \mathrm{p}(\beta)$.
2. x fresh: Relative to a proof rule $\frac{c' \vdash_{\Delta'} A' : \phi'}{c \vdash_\Delta A : \phi}$, x fresh means that $x \notin \mathrm{fn}(c) \cup \mathrm{fn}(A) \cup \mathrm{fn}(\phi)$.
3. $A \equiv_c B$: For all ε, if $\varepsilon \models c$ then $A \equiv_\varepsilon B$.
4. $A \succ_c B$: For all ε, if $\varepsilon \models c$ then $A \succ_\varepsilon B$.

The rules should be fairly uncontroversial given the semantics of formulas and our previous comments.

In addition to the local rules the proof system is equipped with the following single rule for discharging hypotheses:

$$\textsc{dis}: \quad \frac{\begin{array}{c}[c' \vdash_{\Delta'} A : U\ x_1 \cdots x_n] \\ \vdots \\ c \vdash_\Delta A : U\ x_1 \cdots x_n\end{array}}{c \vdash_\Delta A : U\ x_1 \cdots x_n} \quad (\models c' \supset c)$$

where it is required that $\Delta(U)$ is a formula of the form $\nu X.\phi$, and that the given derivation of $c \vdash_\Delta A : U\ x_1 \cdots x_n$ is nontrivial, in the sense that it contains an application of an introduction rule. The following example shows that the side-condition $\models c' \supset c$ is indeed necessary: Let

$$\begin{array}{rcl} B & = & \mathrm{fix} D.\overline{x_2}.[y]x_1.(\lambda y)(Dy) \\ A & = & x_1.(\lambda y)(y = z)(B)(\mathbf{0}). \end{array}$$

Then, if the side-condition on DIS is absent, the following false sequent is derivable:

$$\mathrm{true} \vdash_\Delta A : [x_1]\forall\lambda y.(y \neq z) \vee (\nu X.<\overline{x_2}>\Sigma\lambda y.(y = z) \wedge ([x_1]\forall\lambda y.X)).$$

There is a close relationship between the proof system of fig. 4 and the tableau system of Stirling and Walker [18]. For the fragment of closed positive modal

Introduction rules:

$$\text{EQ:} \quad \frac{}{c \vdash_\Delta A : x = y} \quad (\models c \supset x = y)$$

$$\text{INEQ:} \quad \frac{}{c \vdash_\Delta A : x \neq y} \quad (\models c \supset x \neq y)$$

$$\text{AND:} \quad \frac{c \vdash_\Delta A : \phi \quad c \vdash_\Delta A : \psi}{c \vdash_\Delta A : \phi \wedge \psi}$$

$$\text{OR-1:} \quad \frac{c \vdash_\Delta A : \phi}{c \vdash_\Delta A : \phi \vee \psi} \quad \text{OR-2:} \quad \frac{c \vdash_\Delta A : \psi}{c \vdash_\Delta A : \phi \vee \psi}$$

$$\text{DIA:} \quad \frac{c \vdash_\Delta B : \phi}{c \vdash_\Delta A : <\alpha>\phi} \quad (A \succ_c \alpha.B)$$

$$\text{BOX:} \quad \frac{\{c' \vdash_\Delta B : \phi \mid A \succ_{c'} \beta.B,\ \models c' \supset c, \alpha \text{ and } \beta\ c'\text{-match}\}}{c \vdash_\Delta A : [\alpha]\phi}$$

$$\text{FIX:} \quad \frac{c \vdash_{\Delta \cdot (U \mapsto \sigma X.\phi)} A : U\ x_1 \cdots x_n}{c \vdash_\Delta A : \sigma X.\phi\ x_1 \cdots x_n}$$

$$\text{FOLD:} \quad \frac{c \vdash_\Delta A : \phi[X := U]\ x_1 \cdots x_n}{c \vdash_\Delta A : U\ x_1 \cdots x_n} \quad (\Delta(U) = \sigma X.\phi)$$

$$\text{LAMBDA:} \quad \frac{c \vdash_\Delta A : \phi\{x_1/x\}\ x_2 \cdots x_n}{c \vdash_\Delta A : (\lambda x.\phi)\ x_1 \cdots x_n} \quad \text{APP:} \quad \frac{c \vdash_\Delta A : \phi\ x\ x_1 \cdots x_n}{c \vdash_\Delta A : (\phi x)\ x_1 \cdots x_n}$$

$$\text{SIGMA-1:} \quad \frac{c \vdash_\Delta A : \phi\ x_1 \cdots x_n}{c \vdash_\Delta [x_1]A : \Sigma\phi\ x_2 \cdots x_n}$$

$$\text{SIGMA-2:} \quad \frac{c \wedge \bigwedge\{z \neq y \mid y \text{ not fresh}\} \vdash_\Delta A\{z/x\} : \phi\ z\ x_1 \cdots x_n}{c \vdash_\Delta (\nu x)[x]A : \Sigma\phi\ x_1 \cdots x_n} \quad (z \text{ fresh})$$

$$\text{FORALL:} \quad \frac{c \vdash_\Delta Ay : \phi\ y\ x_1 \cdots x_n}{c \vdash_\Delta A : \forall\phi\ x_1 \cdots x_n} \quad (y \text{ fresh})$$

$$\text{EXISTS:} \quad \frac{c \vdash_\Delta Ay : \phi\ y\ x_1 \cdots x_n}{c \vdash_\Delta A : \exists\phi\ x_1 \cdots x_n}$$

Structural rules:

$$\text{OR-COND:} \quad \frac{c_1 \vdash_\Delta A : \phi \quad c_2 \vdash_\Delta A : \phi}{c_1 \vee c_2 \vdash_\Delta A : \phi}$$

$$\text{EX-COND:} \quad \frac{c \vdash_\Delta A : \phi}{\exists x.c \vdash_\Delta A : \phi} \quad (x \notin \mathrm{fn}(A) \cup \mathrm{fn}(\phi_\Delta))$$

$$\text{CONS:} \quad \frac{c_1 \vdash_\Delta A : \phi}{c_2 \vdash_\Delta A : \phi} \quad (\models c_2 \supset c_1)$$

$$\text{EQUIV:} \quad \frac{c \vdash_\Delta A : \phi}{c \vdash_\Delta B : \phi} \quad (A \equiv_c B) \quad \text{REN:} \quad \frac{c \vdash_\Delta A : \phi(x)}{c \vdash_\Delta A : \phi(y)} \quad (\models c \supset x = y)$$

Figure 4: Local proof rules

μ-calculus formulas and CCS agents, the two systems coincide in the sense that there is a successful tableau for $A \vdash_\Delta \phi$ in the notation of [18] iff there is a proof of true $\vdash_\Delta A : \phi$ in the system of fig. 4.

Note that BOX causes the proof system to be infinitary. This problem, however, is only superficial, as we proceed to show. While the set of antecedents of BOX $\{c' \vdash_\Delta B : \phi \mid A \succ_{c'} \beta.B, \models c' \supset c, \alpha$ and β c'-match$\}$ is infinite, only a finite number of name conditions c' and $\equiv_{c'}$-equivalence classes need actually be considered. The key is to apply the box-rules only when A is in normal form, and then disregarding the structural equivalence relation. Thus let $A \succ_\varepsilon^- B$ if $A \succ_\varepsilon B$ is derivable using $\equiv_\varepsilon$ only for alpha conversions. The following finitary version of BOX results:

$$\text{FIN-BOX:} \quad \frac{\{c' \vdash_\Delta B : \phi \mid \mathcal{C}_1, \mathcal{C}_2\}}{c \vdash_\Delta A : [\alpha]\phi}$$

where $\mathcal{C}_1$ and $\mathcal{C}_2$ are the following conditions:

- $\mathcal{C}_1$: $A \succ_{c'}^- \beta.B$, $\models c' \supset c$, α and β c'-match, and A is in normal form.
- $\mathcal{C}_2$: c' is minimal in the sense that if c'' is any other name condition such that $\mathcal{C}_1$ holds with c'' in place of c', and if $\models c' \supset c''$, then $\models c'' \supset c'$.

Similarly we can replace the rule DIA by the rule FIN-DIA where the side-condition $A \succ_c \alpha.B$ is replaced by the condition $A \succ_c^- \alpha.B$.

Proposition 5.1 (Finitary box-rules) *A sequent $c \vdash_\Delta A : \phi$ is derivable using* BOX *and* DIA *iff it is derivable using* FIN-BOX *and* FIN-DIA.

PROOF This is a consequence of the following standardisation property: If $A \succ_\varepsilon \alpha.B$ then there are A', B' such that A' is in normal form, $A \equiv_\varepsilon A'$, $A' \succ_\varepsilon^- B'$, and $B' \equiv_\varepsilon B$. □

In the remainder of the paper we tacitly assume that the rules FIN-BOX and FIN-DIA are being used in place of BOX and DIA. Note that strictly speaking FIN-BOX remains infinitary due to the fact that name conditions range over syntactical name conditions rather than sets of names. This, however, can easily be overcome, for instance by using normal forms. We obtain the following soundness, completeness, and decidability results for finite control processes:

Theorem 5.2 (Soundness, Completeness, Decidability) *Let $c \vdash_\Delta A : \phi$ be a sequent with A of finite control.*

1. *The following conditions are equivalent:*
   (a) *$c \vdash_\Delta A : \phi$ is derivable.*
   (b) *$c \vdash_\Delta A : \phi$ is true.*
2. *Derivability of $c \vdash_\Delta A : \phi$ is decidable.* □

The remaining part of the paper is devoted to a proof of Theorem 5.2. First we give a direct characterisation of true sequents in terms of a symbolic semantics. Using this semantics we proceed to prove soundness. For decidability and completeness we then present the model checking algorithm, show its termination, and, using this, finally establish completeness and decidability.

6 Symbolic Semantics

The point of the symbolic semantics is to replace the relativisation of the standard semantics to name partitions with relativisation to more general name conditions, thus providing a direct semantical correlate of the notion of true sequent. Thus the symbolic semantics assigns to each ϕ of arity n a set $\|\phi\|_s \delta x_1 \cdots x_n c \subseteq \mathcal{A}$ where δ is a *symbolic* environment. Such environments differ from proposition environments ρ only in that they depend on general name conditions instead of name partitions.

For technical reasons attention needs to be restricted to name-condition maps $f : c \mapsto S \subseteq \mathcal{A}$ which are *well-behaved* (abbreviated w-b) in the sense that

$$f(c_1 \vee c_2) = (fc_1) \cap (fc_2)$$

for all c_1, c_2. A symbolic environment δ is then well-behaved if $\delta X y_1 \cdots y_k$ is well-behaved for all $X : k$ and $y_1, \ldots, y_k$. Well-behaved maps are closed under arbitrary infima and suprema. Note, however, that while suprema of chains of well-behaved maps *can* be computed pointwise, this is not generally true for arbitrary suprema.

As in section 4, $\|\phi\|_s \delta x_1 \cdots x_n c$ is abbreviated to $\|\phi\|_s x_1 \cdots x_n c$ when ϕ is propositionally closed. To define $\|\cdot\|_s$ it is convenient first by mutual recursion to define its specialisation $\|\cdot\|_{nf}$ to normal forms, and then derive $\|\cdot\|_s$ itself in the following manner:

$$\begin{array}{l} \|\phi\|_s \delta x_1 \cdots x_n c = \\ \qquad \{A \mid \forall \varepsilon, B \in \mathrm{NF},\ \text{if } A \equiv_\varepsilon B \text{ and } \varepsilon \models c \text{ then } B \in \|\phi\|_{nf} \delta x_1 \cdots x_n c_\varepsilon\} \end{array}$$

where c_ε abbreviates the condition

$$\bigwedge(\{x = y \mid x, y \in N, \varepsilon \models x = y\} \cup \{x \neq y \mid x, y \in N, \varepsilon \models x \neq y\}),$$

and $N = \mathrm{fn}(A) \cup \mathrm{fn}(\phi) \cup \{x_1, \ldots, x_n\}$. The same abbreviation is used in the definition of $\|\cdot\|_{nf}$ shown in fig 5. The correctness of the symbolic semantics is expressed in the following Lemma:

Lemma 6.1 *Let $\phi : n$ and ϕ propositionally closed. Then $A \in \|\phi\|_s x_1 \cdots x_n c$ iff for all ε, $A \in \|\phi\| x_1 \cdots x_n \varepsilon$ whenever $\varepsilon \models c$.*

PROOFWe show:

$$
\begin{array}{lcl}
\|x = y\|_{nf}\delta c & = & \begin{cases} \mathrm{NF} & \text{if } \models c \supset x = y \\ \emptyset & \text{otherwise} \end{cases} \\
\|x \neq y\|_{nf}\delta c & = & \begin{cases} \mathrm{NF} & \text{if } \models c \supset x \neq y \\ \emptyset & \text{otherwise} \end{cases} \\
\|\phi \wedge \psi\|_{nf} & = & \|\phi\|_{nf} \sqcap \|\psi\|_{nf} \\
\|\phi \vee \psi\|_{nf}\delta c & = & \cup\{\|\phi\|_{nf}\delta c_1 \cap \|\psi\|_{nf}\delta c_2 \mid \models c \supset c_1 \vee c_2\} \\
 & & \cup(\|\phi\|_{nf}\delta c) \cup (\|\psi\|_{nf}\delta c) \\
\|{<}\alpha{>}\phi\|_{nf}\delta c & = & \{A \in \mathrm{NF} \mid \forall\varepsilon,\ \text{if } \varepsilon \models c \text{ then } \exists\beta, B,\ A \succ_\varepsilon \beta.B, \\
 & & \varepsilon \models \alpha = \beta, \text{ and } B \in \|\phi\|_s\delta c_\varepsilon\} \\
\|[\alpha]\phi\|_{nf}\delta c & = & \{A \in \mathrm{NF} \mid \forall\varepsilon, \beta, B,\ \text{if } \varepsilon \models c,\ A \succ_\varepsilon \beta.B, \text{ and} \\
 & & \varepsilon \models \alpha = \beta, \text{ then } B \in \|\phi\|_s\delta c_\varepsilon\} \\
\|X\|_{nf}\delta x_1 \cdots x_n c & = & \{A \in \mathrm{NF} \mid A \in \delta X x_1 \cdots x_n c\} \\
\|\nu X.\phi\|_{nf}\delta x_1 \cdots x_n c & = & \{A \in \mathrm{NF} \mid A \in (\sqcup\{f \mid f \sqsubseteq \|\phi\|_{nf}\delta[X \mapsto f]\})x_1 \cdots x_n c\} \\
\|\mu X.\phi\|_{nf}\delta x_1 \cdots x_n c & = & \{A \in \mathrm{NF} \mid A \in (\sqcap\{f \mid \|\phi\|_{nf}\delta[X \mapsto f] \sqsubseteq f\})x_1 \cdots x_n c\} \\
\|\lambda x.\phi\|_{nf}\delta x_1 \cdots x_n c & = & \|\phi\{x_1/x\}\|_{nf}\delta x_2 \cdots x_n c \\
\|\phi x_1\|_{nf}\delta x_2 \cdots x_n c & = & \|\phi\|_{nf}\delta x_1 \cdots x_n c \\
\|\Sigma\phi\|_{nf}\delta x_2 \cdots x_n c & = & \{A = [x_1]A' \mid A' \in \|\phi\|_s\delta x_1 \cdots x_n c\} \cup \\
 & & \{A = (\nu x)[x]A' \mid \exists \text{ fresh } x_1.\ A\{x_1/x\} \in \\
 & & \|\phi\|_s\delta x_1 \cdots x_n(c \wedge \bigwedge\{x_1 \neq y \mid y \text{ not fresh}\})\} \\
\|\forall\phi\|_{nf}\delta x_2 \cdots x_n c & = & \{A = (\lambda x)A' \mid \exists \text{ fresh } x_1.\ A'\{x_1/x\} \in \|\phi\|_s\delta x_1 \cdots x_n c\} \\
\|\exists\phi\|_{nf}\delta x_2 \cdots x_n c & = & \{A = (\lambda x)A' \mid \exists x_1.A'\{x_1/x\} \in \|\phi\|_s\delta x_1 \cdots x_n c'\} \\
 & & \text{where } c' = c \wedge \bigwedge\{x_1 \neq z \mid z \text{ not fresh}\} \text{ if } x_1 \text{ fresh}, \\
 & & \text{and } c' = c \text{ otherwise}
\end{array}
$$

Figure 5: Symbolic semantics

1. For all name substitutions ε,

$$A \in \|\phi\|_s \rho' x_1 \cdots x_n c_\varepsilon \text{ iff } A \in \|\phi\| \rho x_1 \cdots x_n \varepsilon$$

   where ρ' is derived from ρ by

$$\rho' X y_1 \cdots y_m c = \bigcap \{\rho X y_1 \cdots y_m \varepsilon \mid \varepsilon \models c\}.$$

2. Whenever δ is well-behaved then

$$\|\phi\|_s \delta x_1 \cdots x_n (c_1 \vee c_2) = (\|\phi\|_s \delta x_1 \cdots x_n c_1) \wedge (\|\phi\|_s \delta x_1 \cdots x_n c_2).$$

Together (1) and (2) implies the desired conclusion. We first prove that (1) follows from (2). Observe first that

$$\|\phi\|_s \rho' x_1 \cdots x_n c_\varepsilon = \{A \mid \forall B \in \mathrm{NF}, \text{if } A \equiv_\varepsilon B, \text{ then } B \in \|\phi\|_{nf} \rho' x_1 \cdots x_n c_\varepsilon\}.$$

It thus suffices to show that $A \in \|\phi\|_{nf} \rho' x_1 \cdots x_n c_\varepsilon$ iff $A \in \|\phi\| \rho x_1 \cdots x_n \varepsilon$, assuming $A \in \mathrm{NF}$. The following Lemma expresses the contravariance of $\|\cdot\|_{nf}$ in its last argument, and allows unused names to be projected out.

Lemma 6.2 *1. If* $\models c_1 \supset c_2$ *then* $\|\phi\|_{nf} \delta x_1 \cdots x_n c_2 \subseteq \|\phi\|_{nf} \delta x_1 \cdots x_n c_1$.

2. Suppose that $x \notin N$. *Then* $A \in \|\phi\|_{nf} \delta x_1 \cdots x_n c$ *iff* $A \in \|\phi\|_{nf} \delta x_1 \cdots x_n (\exists x.c)$.

Proof1. Structural induction in ϕ. 2. The if-direction follows from 1, and the only-if direction is proved by structural induction. $\Box$(Lemma 6.2)

We now prove (1) by induction in the structure of ϕ, assuming $A \in \mathrm{NF}$:

$$\begin{array}{ll}
\multicolumn{2}{l}{A \in \|\phi \vee \psi\|_{nf} \rho' c_\varepsilon} \\
\text{iff} & A \in \|\phi\|_{nf} \rho' c_\varepsilon, \text{ or } A \in \|\psi\|_{nf} \rho' c_\varepsilon, \text{ or} \\
& \exists c_1, c_2.\ A \in \|\phi\|_{nf} \rho' c_1 \cap \|\psi\|_{nf} \rho' c_2 \text{ and } \models c_\varepsilon \supset c_1 \vee c_2 \\
\text{iff} & A \in \|\phi\|_{nf} \rho' c_\varepsilon, \text{ or } A \in \|\psi\|_{nf} \rho' c_\varepsilon, \text{ or} \\
& \exists c_1, c_2.\ A \in \|\phi\|_{nf} \rho' c_1 \cap \|\psi\|_{nf} \rho' c_2 \text{ and } \models c_\varepsilon \supset c_1, \text{ or} \\
& A \in \|\phi\|_{nf} \rho' c_1 \cap \|\psi\|_{nf} \rho' c_2 \text{ and } \models c_\varepsilon \supset c_2 \\
& \text{(Using 6.2.2 and property of } c_\varepsilon) \\
\text{iff} & A \in \|\phi\|_{nf} \rho' c_\varepsilon \text{ or } A \in \|\psi\|_{nf} \rho' c_\varepsilon \text{ (by 6.2.1)} \\
\text{iff} & A \in \|\phi\| \rho \varepsilon \text{ or } A \in \|\psi\| \rho \varepsilon \text{ (by the induction hypothesis)} \\
\text{iff} & A \in \|\phi \vee \psi\| \rho \varepsilon
\end{array}$$

$$\begin{array}{ll}
\multicolumn{2}{l}{A \in \|[\alpha]\phi\|_{nf} \rho' c_\varepsilon} \\
\text{iff} & \forall \varepsilon', \beta, B. \text{ if } \varepsilon' \models c_\varepsilon,\ A \succ_{\varepsilon'} \beta.B, \text{ and } \varepsilon'(\alpha) = \varepsilon'(\beta) \text{then } B \in \|\phi\|_s \rho' c_{\varepsilon'} \\
\text{iff} & \forall \beta, B. \text{ if } A \succ_\varepsilon \beta.B, \text{ and } \varepsilon(\alpha) = \varepsilon(\beta), \text{ then } B \in \|\phi\|_s \rho' c_\varepsilon
\end{array}$$

iff $\forall\beta, B.$ if $A \succ_\varepsilon \beta.B$, and $\varepsilon(\alpha) = \varepsilon(\beta)$, then $B \in \|\phi\|\rho\varepsilon$

(By the induction hypothesis)

iff $A \in \|[\alpha]\phi\|\rho\varepsilon$

$A = (\nu x)[x]A' \in \|\Sigma\phi\|_{nf}\rho' x_1 \cdots x_n c_\varepsilon$

iff $\exists$ fresh y. $A'\{y/x\} \in \|\phi\|_s \rho' y x_1 \cdots x_n (c_\varepsilon \wedge \bigwedge\{y \neq z \mid z \text{ not fresh}\})$

iff $\exists y \notin N$. $A'\{y/x\} \in \|\phi\|_s \rho' y x_1 \cdots x_n c_{(\nu y)\varepsilon}$

iff $\exists y \notin N$. $A'\{y/x\} \in \|\phi\|\rho y x_1 \cdots x_n ((\nu y)\varepsilon)$

(By the induction hypothesis)

iff $(\nu x)[x]A' = A \in \|\Sigma\phi\|\rho x_1 \cdots x_n \varepsilon$

$A = (\lambda x)A' \in \|\forall\phi\|_{nf}\rho' x_1 \cdots x_n c_\varepsilon$

iff $\exists y \notin N$. $A\{y/x\} \in \|\phi\|_s \rho' y x_1 \cdots x_n c_\varepsilon$

iff $\forall y$. $A'\{y/x\} \in \|\phi\|_s \rho' y x_1 \cdots x_n c_\varepsilon$

(By (2))

iff $\forall y$. $A'\{y/x\} \in \|\phi\|\rho y x_1 \cdots x_n \varepsilon$

(By the induction hypothesis)

iff $(\lambda x)A' = A \in \|\phi\|\rho x_1 \cdots x_n \varepsilon$

The remaining cases, except those for fixed points, are proved by similar methods. For the fixed points it suffices from the assumption that ϕ satisfies (1) and (2) for greatest fixed points to show:

(a) If $g \sqsubseteq \|\phi\|\rho[X \mapsto g]$ then $g' \sqsubseteq \|\phi\|_s \rho'[X \mapsto g']$ where g' is defined by

$$g' y_1 \cdots y_m c = \bigcap\{g y_1 \cdots y_m \varepsilon \mid \varepsilon \models c\}.$$

(b) If $f \sqsubseteq \|\phi\|_s \rho'[X \mapsto f]$ and f is well-behaved then $f^\dagger \sqsubseteq \|\phi\|\rho[X \mapsto f^\dagger]$ where $f^\dagger$ is defined by

$$f^\dagger y_1 \cdots y_m \varepsilon = f y_1 \cdots y_m c_\varepsilon$$

where c_ε is computed relative to $\mathrm{fn}(\phi) \cup \{y_1, \ldots, y_m\}$.

We leave the proof of (a) to the reader. For (b) note first that if f is well-behaved then

$$(\rho[X \mapsto f^\dagger])' = \rho'[X \mapsto f].$$

Suppose that $f \sqsubseteq \|\phi\|_s \rho'[X \mapsto f]$, i.e. for all $x_1, \ldots x_n$ and c, $f x_1 \cdots x_n c \subseteq \|\phi\|_s \rho'[X \mapsto f] x_1 \cdots x_n c$. Then in particular for all ε, $f x_1 \cdots x_n c_\varepsilon \subseteq \|\phi\|_s \rho'[X \mapsto f] x_1 \cdots x_n c_\varepsilon$ where c_ε is computed relative to $\mathrm{fn}(\phi) \cup \{x_1, \ldots x_n\}$. By the above observation it follows that $f x_1 \cdots x_n c_\varepsilon \subseteq \|\phi\|_s (\rho[X \mapsto f^\dagger])' x_1 \cdots x_n c_\varepsilon$, consequently by the induction hypothesis for (1) and the definition of $f^\dagger$, $f^\dagger x_1 \cdots x_n \varepsilon \subseteq$

$\|\phi\|\rho[X \mapsto f^\dagger]x_1 \cdots x_n\varepsilon$ as was to be shown since $x_1, \ldots x_n$ and ε were arbitrary. The checks for least fixed points are dual to the cases for greatest fixed points.

Next for the proof of (2), assuming that δ is well-behaved:

$$
\begin{aligned}
&\|\phi \vee \psi\|_{nf}\delta(c_1 \vee c_2)\\
&\quad= \bigcup\{\|\phi\|_{nf}\delta c_1' \cap \|\psi\|_{nf}\delta c_2' \mid \models c_1 \vee c_2 \supset c_1' \vee c_2'\}\\
&\qquad \cup(\|\phi\|_{nf}\delta(c_1 \vee c_2)) \cup (\|\psi\|_{nf}\delta(c_1 \vee c_2))\\
&\quad= \bigcup\{\|\phi\|_{nf}\delta c_1' \cap \|\psi\|_{nf}\delta c_2' \mid\\
&\qquad \models c_1 \supset c_1' \vee c_2' \text{ and } \models c_2 \supset c_1' \vee c_2'\}\\
&\qquad \cup(\|\phi\|_{nf}\delta(c_1 \vee c_2)) \cup (\|\psi\|_{nf}\delta(c_1 \vee c_2))\\
&\quad= \bigcup\{\|\phi\|_{nf}\delta(c_{1,1} \vee c_{2,1}) \cap \|\psi\|_{nf}\delta(c_{1,2} \vee c_{2,2}) \mid\\
&\qquad \models c_1 \supset c_{1,1} \vee c_{1,2} \text{ and } \models c_2 \supset c_{2,1} \vee c_{2,2}\}\\
&\qquad \cup(\|\phi\|_{nf}\delta(c_1 \vee c_2)) \cup (\|\psi\|_{nf}\delta(c_1 \vee c_2))\\
&\quad= \bigcup\{\|\phi\|_{nf}\delta c_{1,1} \cap \|\phi\|_{nf}\delta c_{2,1} \cap \|\psi\|_{nf}\delta c_{1,2}\\
&\qquad \cap\|\psi\|_{nf}\delta c_{2,2} \mid\models c_1 \supset c_{1,1} \vee c_{1,2} \text{ and } \models c_2 \supset c_{2,1} \vee c_{2,2}\}\\
&\qquad \cup((\|\phi\|_{nf}\delta c_1) \cap (\|\phi\|_{nf}\delta c_2)) \cup ((\|\psi\|_{nf}\delta c_1) \cap (\|\psi\|_{nf}\delta c_2))\\
&\qquad \text{(by the induction hypothesis)}\\
&\quad= (\bigcup\{\|\phi\|_{nf}\delta c_{1,1} \cap \|\psi\|_{nf}\delta c_{1,2} \mid\models c_1 \supset c_{1,1} \vee c_{1,2}\} \cup (\|\phi\|_{nf}\delta c_1) \cup (\|\psi\|_{nf}\delta c_1)) \cap\\
&\qquad (\bigcup\{\|\phi\|_{nf}\delta c_{2,1} \cap \|\psi\|_{nf}\delta c_{2,2} \mid\models c_2 \supset c_{2,1} \vee c_{2,2}\} \cup (\|\phi\|_{nf}\delta c_2) \cup (\|\psi\|_{nf}\delta c_2))\\
&\qquad \text{(By calculation)}\\
&\quad= \|\phi \vee \psi\|_{nf}\delta c_1 \cap \|\phi \vee \psi\|_{nf}\delta c_2
\end{aligned}
$$

$$
\begin{aligned}
&\|[\alpha]\phi\|_{nf}\delta(c_1 \vee c_2)\\
&\quad= \{A \in \mathrm{NF} \mid \forall\varepsilon, \beta, B.\ \text{if } \varepsilon \models c_1 \vee c_2,\ A \succ_\varepsilon \beta.B,\ \text{and } \varepsilon \models \alpha = \beta\\
&\qquad \text{then } B \in \|\phi\|_s\delta c_\varepsilon\}\\
&\quad= \{A \in \mathrm{NF} \mid \forall\varepsilon, \beta, B.\ \text{if } (\varepsilon \models c_1 \text{ or } \varepsilon \models c_2),\ A \succ_\varepsilon \beta.B,\ \text{and } \varepsilon \models \alpha = \beta\\
&\qquad \text{then } B \in \|\phi\|_s\delta c_\varepsilon\}\\
&\quad= (\|[\alpha]\phi\|_{nf}\delta c_1) \cap (\|[\alpha]\phi\|_{nf}\delta c_2)
\end{aligned}
$$

The remaining cases follow in equally straightforward manners from the induction hypothesis, and are left to the reader. $\Box$(Lemma 6.1)

7 Soundness

Crucial to the proofs of soundness, completeness, and decidability is the use of ordinal approximations $\nu^\alpha X.\phi$ and $\mu^\alpha X.\phi$. These are defined as follows:

$$\|\nu^0 X.\phi\|_s\delta x_1 \cdots x_n c \;=\; \mathcal{A}$$

$$
\begin{aligned}
\|\nu^{\alpha+1}X.\phi\|_s\delta x_1\cdots x_n c &= \|\phi\|_s\delta[X\mapsto\|\nu^\alpha X.\phi\|_s\delta]x_1\cdots x_n c\\
\|\nu^{\lambda}X.\phi\|_s\delta x_1\cdots x_n c &= \bigcap_{\alpha<\lambda}\|\nu^\alpha X.\phi\|_s\delta x_1\cdots x_n c\\
\|\mu^{0}X.\phi\|_s\delta x_1\cdots x_n c &= \emptyset\\
\|\mu^{\alpha+1}X.\phi\|_s\delta x_1\cdots x_n c &= \|\phi\|_s\delta[X\mapsto\|\mu^\alpha X.\phi\|_s\delta]x_1\cdots x_n c\\
\|\mu^{\lambda}X.\phi\|_s\delta x_1\cdots x_n c &= \bigcup_{\alpha<\lambda}\|\mu^\alpha X.\phi\|_s\delta x_1\cdots x_n c
\end{aligned}
$$

It follows by standard techniques that $\sqcap_\alpha\|\nu^\alpha X.\phi\|_s\delta$ is the greatest fixed point of $\lambda f.\|\phi\|_s\delta[X\mapsto f]$, and that $\sqcup_\alpha\|\mu^\alpha X.\phi\|_s\delta$ is the least. In fact only reference to countable ordinals are needed. Note, however, that we need also to verify that $\|\nu^\alpha X.\phi\|_s\delta$ and $\|\mu^\alpha X.\phi\|_s\delta$ are well-behaved for all ordinals α, whenever δ is well-behaved too. That this is so can be seen from the proof of Lemma 6.1.

We can then proceed to prove soundness. The proof given here follows the lines of the corresponding proof in [18].

Theorem 7.1 (Soundness) *If $c\vdash_\Delta A:\phi$ is derivable then it is true.*

Proof First observation to note is that if all antecedents of a local rule are true then so is the conclusion. This follows immediately from the symbolic semantics, Lemma 6.1. Suppose then that a proof of $c\vdash_\Delta A:\phi$ is given, and that $c\vdash_\Delta A:\phi$ is false, i.e. (by Lemma 6.1) $A\notin\|\phi_\Delta\|_s c$. For every sequent occurring in the proof, if it is false then so is an antecedent of that sequent. If a sequent has no antecedents then it is true. Thus we can find a constant U_1 such that

1. it is possible to trace a path upwards through the proof using only false sequents from the sequent $c\vdash_\Delta A:\phi$ to a sequent of the form $c_1\vdash_{\Delta_1} A_1 : U_1 x_{1,1}\cdots x_{1,m_1}$,
2. $\Delta_1(U_1)$ is a ν-formula, and
3. If U is another ν-constant introduced strictly before U_1 (i.e. occurring before U_1 in Δ_1) then (1) and (2) fails to hold of U.

For if no such U_1 exists then it will be possible to trace an infinite path upwards from $c\vdash_\Delta A:\phi$, but this is impossible. Note that we can additionally require the traced path to be as short as possible. Thus $c_1\vdash_{\Delta_1} A_1 : U_1 x_{1,1}\cdots x_{1,m_1}$ is prevented from being an occurrence of a hypothesis.

Having now reached the sequent $c_1\vdash_{\Delta_1} A_1 : U_1 x_{1,1}\cdots x_{1,m_1}$ the proof proceeds iteratively, in the limit tracing an infinite path through the given (finite) proof. The first iteration step proceeds as follows:

Consider the subproof rooted in $c_1\vdash_{\Delta_1} A_1 : U_1 x_{1,1}\cdots x_{1,m_1}$. Using ordinal approximations we can find a minimal α such that if $\Delta_1(U_1)=\nu X.\phi_1$ then

$$A_1\notin\|\nu^\alpha X.\phi_{1_{\Delta_1}} x_{1,1}\cdots x_{1,m_1}\|_s c_1.$$

We index occurrences of U_1 in the subproof. Thus occurrences of U_1 indexed by α' are interpreted as $\nu^{\alpha'}X.\phi_1$ rather than simply $\nu X.\phi_1$ in determining truthhood of sequents. At the root sequent U_1 is indexed by α and subsequently, every time U_1 is unfolded, the index is minimised (while preserving truthhood/falsehood of sequents), and thus strictly decreased. Using this procedure all occurrences of U_1 are indexed. For the only rule that could prevent this from being true is FIX eliminating U_1. But then the choice of U_1 would have violated the convention that constants are defined at most once.

In the indexed subproof the root sequent $c_1 \vdash_{\Delta_1} A_1 : U_1^{\alpha} x_{1,1} \cdots x_{1,m_1}$ is false. We now show that we can find some new constant U_2 such that

1. it is possible to trace a path in the indexed subproof using only false sequents upwards from $c_1 \vdash_{\Delta_1} A_1 : U_1^{\alpha} x_{1,1} \cdots x_{1,m_1}$ to a sequent of the form $c_2 \vdash_{\Delta_2} A_2 : U_2 x_{2,1} \cdots x_{2,m_2}$,
2. $\Delta_2(U_2)$ is a ν-formula, and
3. If U is another ν-constant introduced strictly before U_2 then (1)–(2) fails to hold of U.
4. U_2 is introduced strictly after U_1.

Starting from the root sequent the path is built step by step. Having reached a false sequent $c' \vdash_{\Delta'} A' : \phi'$ it is either an occurrence of a hypothesis, or else it has some antecedent which is false too. If the latter case applies and a suitable U_2 has not yet been found, the construction merely proceeds. Suppose the first case applies with ϕ' of the form, say, $U' x_1' \cdots x_{m'}'$. It cannot be that U' was introduced before U_1 since otherwise U' would have been chosen instead of U_1. Neither can it be the case that $U' = U_1$. For suppose otherwise. Let then α' index U_1 at this hypothesis occurrence. Since the path from $c \vdash_{\Delta} A : \phi$ to $c_1 \vdash_{\Delta_1} A_1 : U_1 x_{1,1} \cdots x_{1,m_1}$ was chosen as short as possible, the construction must previously have encountered a sequent of the form $c'' \vdash_{\Delta'} A' : U_1^{\alpha''} x_1' \cdots x_{m'}'$ which was not an occurrence of a hypothesis, and such that $\models c' \supset c''$, by rule DIS. Since $c' \vdash_{\Delta'} A' : \phi'$ is false, $A' \notin \|U_1^{\alpha'} x_1' \cdots x_{m'}'\|_s c'$. Then $A' \notin \|U_1^{\alpha'} x_1' \cdots x_{m'}'\|_s c''$ either. But α'' is strictly greater than α', and α'' was chosen minimal such that $A' \notin \|U_1^{\alpha''} x_1' \cdots x_{m'}'\|_s c''$, a contradiction. The only possibility is thus that U' be introduced strictly after U_1. But then we're done, since we have identified one possible candidate for U_2, and among all candidates we can then choose one for which (3) above is true.

Note that, again, by choosing the path as small as possible we can ensure that $c_2 \vdash_{\Delta_2} A_2 : U_2 x_{2,1} \cdots x_{2,m_2}$ is not an occurrence of a hypothesis. For if it were we would find some application of DIS discharging this hypothesis, and concluding the sequent $c_2' \vdash_{\Delta_2} A_2 : U_2 x_{2,1} \cdots x_{2,m_2}$ for some c_2'. The application of this sequent must be above the current root sequent $c_1 \vdash_{\Delta_1} A_1 : U_1^{\alpha} x_{1,1} \cdots x_{1,m_1}$, since if it were below the convention preventing redefinition of constants would be violated. But then the path construction would have terminated when reaching the sequent $c_2' \vdash_{\Delta_2} A_2 : U_2 x_{2,1} \cdots x_{2,m_2}$, and we're done.

The construction can now proceed iteratively from the false sequent $c_2 \vdash_{\Delta_2} A_2 : U_2 x_{2,1} \cdots x_{2,m_2}$, and the proof is concluded. $\square$

8 The Decision Procedure

In this section we describe the decision procedure central to the completeness and decidability parts of Theorem 5.2. Let an initial sequent $c_0 \vdash_{\Delta_0} A_0 : \phi_0$ be given such that A_0 is of finite control. The decision procedure provides a strategy for building a proof of $c_0 \vdash_{\Delta_0} A_0 : \phi_0$, provided such a proof exists. The procedure builds proofs in a refinement- or goal-directed manner as is usual in tableaux-based approaches. The key issue is to allow attention to be restricted to finite subsets of state spaces which are in general infinite.

First the issue of choice of free and bound names is addressed. Define

$$\begin{aligned}
\sharp_{fns}(A) &\triangleq \max\{|\mathrm{fn}(B)| \mid B \text{ a subterm of } A\} \\
\sharp_{fns}(\phi) &\triangleq \max\{|\mathrm{fn}(\psi)| \mid \psi \text{ a subterm of } \phi\} \\
\sharp_{par}(A) &\triangleq \text{number of occurrences of } \mid \text{ in } A
\end{aligned}$$

We then fix a set $N_0 = \{y_1, \ldots, y_k\}$ of names from which all free and bound occurrences af names will be chosen, where

$$k = \sharp_{fns}(A_0) \cdot (\sharp_{par}(A_0) + 1) + \sharp_{fns}(\phi_0) + 1.$$

The factor $\sharp_{par}(A_0) + 1$ is needed to avoid name clashes during scope extrusion. An alternative to using N_0 for both bound and free names is to use N_0 for free names only, and then use de Bruijn's indexes for bound variables. Whereas little seems to be gained from the latter approach from the point of view of worst case complexity or clarity of presentation, the use of de Bruijn's indexes may prove valuable in speeding up actual implementations.

Rather than general name conditions the proof building procedure uses finite representations of name partitions. Partitions of a finite set N of names have obvious representations as name conditions. Such conditions c have the property that whenever $x, y \in N$ then either $\models c \supset x = y$ or $\models c \supset x \neq y$. Call a condition with this property *N-prime* (or just prime if N is understood from context). Note that if c is N-prime then $\exists x.c$ is $N - \{x\}$-prime, and if $y \notin N \cup \mathrm{fn}(c)$ then $c[x = y] \triangleq c \wedge x = y$ and $(\nu y)c \triangleq c \wedge \bigwedge\{x \neq y \mid x \in N\}$ are $N \cup \{y\}$-prime. By means of the rules OR-COND and EX-COND, $c_0 \vdash_{\Delta_0} A_0 : \phi_0$ can be replaced by a finite set of sequents of the form $c_0' \vdash_{\Delta_0} A_0 : \phi_0$ where $\mathrm{fn}(c_0') = \mathrm{fn}(A_0) \cup \mathrm{fn}(\phi_0)$, and where c_0' is $\mathrm{fn}(c_0')$-prime. We can therefore assume the initial sequent itself to have this property, and let the procedure maintain it invariant.

At each step the procedure either terminates or else it chooses to refine the current goal, say $c \vdash_\Delta A : \phi$, by an instance of one of the proof rules. We assume of c that it is prime, and that all names occurring freely or bound in A or ϕ are in N_0. The choice of proof rule is guided by the structure of ϕ. Using EQUIV and EX-COND A can be assumed to be in normal form, and $\mathrm{fn}(c)$ can if needed be replaced by its restriction to $\mathrm{fn}(A) \cup \mathrm{fn}(\phi)$. Guided by the outermost connective of ϕ, the procedure now proceeds as described by the pseudo-ML function `check2` of fig. 6. The definition of `check2` uses a few anxillary functions and abbreviations:

- `normalform`(A, c) returns a normal form A' such that $A \equiv_\varepsilon A'$ for some ε such that $\varepsilon \models c$. Since $\mathrm{fn}(c) = \mathrm{fn}(A) \cup \mathrm{fn}(\phi)$ and c is $\mathrm{fn}(c)$-prime, the choice of ε is irrelevant, and `normalform`(A, c) is thus well-defined by Proposition 3.1. It is assumed of `normalform`(A, c) that whenever $(\nu x)B$ is a subterm of `normalform`(A, c) then x has a free occurrence in B.

- $\vec{x}$ abbreviates vectors $x_1 \cdots x_n$, and if $\vec{x} = x_1 \cdots x_n$ then $(y, \vec{x}) = y x_1 \cdots x_n$, `hd`$(\vec{x}) = x_1$ and `tl`$(\vec{x}) = x_2 \cdots x_n$.

- `restrict`$(\{x_1, \ldots, x_n\}, c) = \exists x_1, \ldots, x_n.c$.

- `newcon`(Δ) determines a new constant not in $\mathrm{dom}(\Delta)$.

- `newname`$(A, \phi, \vec{x})$ determines a name in N_0 not in $\mathrm{fn}(A) \cup \mathrm{fn}(\phi) \cup \{\vec{x}\}$ if one exists.

In most cases `check2` is self-explanatory. Here we comment only on the case of ϕ of the form $U\vec{x}$. For the purpose of handling constants a table indicating what constant sequents have previously been refined is maintained. Suppose first that no sequent of the form $c' \vdash_{\Delta'} A : \phi$ for some Δ' and c' such that $\models c \leftrightarrow c'$ has been visited. Then we record that $c \vdash_\Delta A : \phi$ has now been visited, and proceed by checking $c \vdash_\Delta A : \phi_1[X := U]\vec{x}$ when $\Delta(U) = \sigma X.\phi_1$. Logically, the recording of $c \vdash_\Delta A : \phi$ amounts to nought when $\sigma = \mu$. However, when $\sigma = \nu$ it corresponds to refinement by DIS. If on the other hand a sequent of the form $c' \vdash_{\Delta'} A : \phi$ as above has already been visited then, if $\sigma = \mu$, the procedure terminates unsuccessfully (as the chosen strategy for refining $c' \vdash_{\Delta'} A : \phi$ did not succeed in eliminating the recursion), and if $\sigma = \nu$ it terminates successfully (since the current goal can then be discharged).

In the next section we prove that the model checking procedure of fig. 6 is well-defined, and then in section 10 we show that it is correct.

9 Termination and Well-definedness

An invocation of `check`$(c \vdash_\Delta A : \phi)$ can, if it yields a well-defined result, be viewed as determinining not only a truth-value, but also a set of proof structures. The aim of the present section is to show that on all inputs `check` is indeed well-defined, and determines a a set of proof structures all members of which are generated by the local and global rules of section 5. To show this the following must be established:

`fun check`$(c \vdash_\Delta A : \phi)$ `=` initialize visited table;
  for all c' such that $\mathrm{fn}(c') = \mathrm{fn}(A) \cup \mathrm{fn}(\phi)$, c' is $\mathrm{fn}(c')$-prime, and $\models c' \supset c$:
  `check1`$(c' \vdash_\Delta A : \phi \ ())$

`fun check1`$(c \vdash_\Delta A : \phi \ \vec{x})$ `=`
  `let` A' `= normalform`(A, c)
  `in check2(restrict(`$\mathrm{fn}(A') \cup \mathrm{fn}(\phi)$`,`$c$`)` $\vdash_\Delta A' : \phi \ \vec{x}$`) end`
`and check2`$(c \vdash_\Delta A : \phi \ \vec{x})$ `=`
  `case` ϕ `of`
    $y = z$ `=>` $\models c \supset y = z$ `|`
    $y \neq z$ `=>` $\models c \supset y \neq z$ `|`
    $\phi_1 \wedge \phi_2$ `=> check2`$(c \vdash_\Delta A : \phi_1 \ \vec{x})$ `andalso check2`$(c \vdash_\Delta A : \phi_2 \ \vec{x})$ `|`
    $\phi_1 \vee \phi_2$ `=> check2`$(c \vdash_\Delta A : \phi_1 \ \vec{x})$ `orelse check2`$(c \vdash_\Delta A : \phi_2 \ \vec{x})$ `|`
    $<\alpha>\phi_1$ `=>` for some A_1, β such that $\models c \supset \alpha = \beta$ and $A \succ_c \beta.A_1$:
      `check1`$(c \vdash_\Delta A_1 : \phi_1 \ \vec{x})$ `|`
    $[\alpha]\phi_1$ `=>` for all A_1, β such that $\models c \supset \alpha = \beta$ and $A \succ_c \beta.A_1$:
      `check1`$(c \vdash_\Delta A_1 : \phi_1 \ \vec{x})$ `|`
    $\sigma X.\phi_1$ `=> let` U `= newcon`(Δ) `in check2`$(c \vdash_{\Delta \cdot (U \mapsto \nu X.\phi_1)} A : U \ \vec{x})$ `end |`
    U `=> if` visited $c \vdash_\Delta A : \phi \ \vec{x}$
      `then (case` $\Delta(U)$ `of` $\nu X.\phi_1$ `=> true |` $\mu X.\phi_1$ `=> false)`
      `else` mark $c \vdash_\Delta A : \phi \ \vec{x}$ visited;
        `(case` $\Delta(U)$ `of` $\sigma X.\phi_1$ `=> check2`$(c \vdash_\Delta A : \phi_1[X := U] \ \vec{x})$`) |`
    $\lambda y.\phi_1$ `=> check2`$(c \vdash_\Delta A : \phi_1\{$`hd`$(\vec{x})/y\}$ `tl`$(\vec{x}))$ `|`
    $\phi_1 y$ `=> check2`$(c \vdash_\Delta A : \phi_1 \ (y, \vec{x}))$ `|`
    $\Sigma\phi_1$ `=> (case` A `of`
      $[y]A_1$ `=> check1`$(c \vdash_\Delta A_1 : \phi_1 \ (y, \vec{x}))$ `|`
      $(\nu y)[y]A_1$ `=> let` z `= newname(`A`,`ϕ_1`,`$\vec{x}$`)`
        `in check1`$((\nu z)c \vdash_\Delta A_1\{z/y\} : \phi_1 \ (z, \vec{x}))$ `end | _ => false) |`
    $\forall\phi_1$ `=> (case` A `of`
      $(\lambda y)A_1$ `=> let` z `= newname(`A`,`ϕ_1`,`$\vec{x}$`)`
        `in check1`$((\nu z)c \vdash_\Delta A_1\{z/y\} : \phi_1 \ (z, \vec{x}))$ `andalso`
          $\forall z' \in \mathrm{fn}(A) \cup \mathrm{fn}(\phi_1) \cup \{\vec{x}\}$:
          `check1`$(c[z = z'] \vdash_\Delta A_1\{z/y\} : \phi_1 \ (z, \vec{x}))$ `end | _ => false) |`
    $\exists\phi_1$ `=> (case` A `of`
      $(\lambda y)A_1$ `=> let` z `= newname(`A`,`ϕ_1`,`$\vec{x}$`)`
        `in check1`$((\nu z)c \vdash_\Delta A_1\{z/y\} : \phi_1 \ (z, \vec{x}))$ `orelse`
          $\exists z' \in \mathrm{fn}(A) \cup \mathrm{fn}(\phi_1) \cup \{\vec{x}\}$:
          `check1`$(c[z = z'] \vdash_\Delta A_1\{z/y\} : \phi_1 \ (z, \vec{x}))$ `end | _ => false)`

Figure 6: Pseudo-ML functions `check`, `check1`, `check2`.

1. That, using the algorithm of fig. 6, only a finite number of agents are reachable.

2. Using (1), that the algorithm terminates on all inputs.

3. That the algorithm determines a well-defined truth-value on all inputs.

4. That each refinement step determined by the algorithm corresponds to a well-defined proof structure.

Together these results show that if a sequent is true then there is a proof for it. It does not follow, however, that the proof has no undischarged hypotheses occurring in it. The proof of this (completeness) is delayed till section 10.

9.1 Agents

We define the relation $A \rightarrow B$ intended to capture the ways agents A in single steps give rise to other agents B using the algorithm of fig. 6. Parametrising the definition is the set N_0 determined from an initial sequent as in section 8. The relation $\rightarrow$ is given as the least relation which respects alpha-conversion with bound names in N_0 (i.e. such that $A \rightarrow B$ whenever A and B are alpha-congruent and B results from A by replacing bound names in N_0 by bound names in N_0), and for which the following properties hold:

1. $A + B \rightarrow A$, $A + B \rightarrow B$

2. $\alpha.A \rightarrow A$

3. $bAB \rightarrow A$, $bAB \rightarrow B$

4. $(\lambda x)A \rightarrow A$

5. $Ax \rightarrow A$

6. $\mathrm{fix}D.A \rightarrow A[\mathrm{fix}D.A/D]$

7. $[x]A \rightarrow A$

8. If $A \rightarrow B$ and $x \in \mathrm{fn}(B)$ then $(\nu x)A \rightarrow (\nu x)B$

9. $(\nu x)A \rightarrow A$

10. $(\nu x)(\lambda y)A \rightarrow (\lambda y)(\nu x)A$

11. If $x \neq y$ then $(\nu x)[y]A \rightarrow [y](\nu x)A$

12. $(\nu x)(\nu y)[y]A \rightarrow (\nu y)[y](\nu x)A$

13. If $A \rightarrow A'$ then $A \mid B \rightarrow A' \mid B$ and $B \mid A \rightarrow B \mid A'$

14. $((\lambda x)A) \mid B \to (\lambda x)(A \mid B)$, $A \mid ((\lambda x)B) \to (\lambda x)(A \mid B)$

15. $([x]A) \mid B \to [x](A \mid B)$, $A \mid ([x]B) \to [x](A \mid B)$

16. $((\nu x)A) \mid B \to (\nu x)(A \mid B)$, $A \mid ((\nu x)B) \to (\nu x)(A \mid B)$

We first show that the relation $A \to B$ correctly reflects the intention:

Proposition 9.1 *If* `check2(`$c' \vdash_{\Delta'} A' : \phi\ \vec{y}$`)` *is invoked from* `check2(`$c \vdash_\Delta A : \phi\ \vec{x}$`)` *then* $A \to^* A'$.

Proof Suppose that c is N-prime and that $\mathrm{fn}(A) \subseteq N$. We need to show the following:

1. A normal form A' can be computed such that $A \equiv_c A'$.

2. If A is a process and $A \succ_c^- \alpha.B$ then $A \to^*$ `normalform`(B, c).

3. $(\lambda x)A \to A\{y/x\}$, $(\nu x)[x]A \to A\{y/x\}$, and $[x]A \to A$ whenever $y \in N_0$.

Of these, (1) can be seen to hold from the proof of Proposition 3.1, (2) by inspecting the rules defining $\succ_\varepsilon^-$, and (3) holds by definition. □

We then proceed to prove finiteness:

Lemma 9.2 (Finiteness) *For all A, $\{B \mid A \to^* B\}$ is finite.*

Proof By König's Lemma, since the assumption of guarded recursion ensures that $\{B \mid A \to B\}$ is always finite, it suffices to show that any infinite derivation

$$d = A_0 \to \cdots \to A_n \to \cdots$$

with $A_0 = A$ visits a finite number of distinct agents only, i.e. $\mathcal{R}(d) = \{A_i \mid i \in \omega\}$ is finite. To show this we define the *size*, $|A|$, of A in the following manner:

$$|\mathbf{0}| = |D| = 1$$

$$|A + B| = |bAB| = |A| + |B| + 1$$
$$|\alpha.A| = |(\lambda x)A| = |Ax| = |[x]A| = |\mathrm{fix} D.A| = |A| + 1$$
$$|(\nu x)A| = 2 \cdot |A| + 1$$
$$|A \mid B| = |A| \cdot |B|$$

Lemma 9.3 *All axioms among (1)–(16) except (6) decrease size, and all rules among (1)–(16) preserve size decrease* □(Lemma 9.3)

Thus, for d to be infinite the unfolding axiom (6) must be appealed to infinitely often (or else alpha conversions are used almost always—this situation is left to the reader to dispose of). We assume here that we can find some i_0 for which A_{i_0} has the form $C_{i_0}(B_{i_0,1}, \ldots, B_{i_0,m})$ where C_{i_0} is an m-ary context built using only operators of the form $[x]$, (λx), (νx), or $|$, and for which each B_j has no occurrences of parallel composition. The situation where this assumption might fail is when one of the $B_{i_0,j}$ fails to have the desired form but is never again "touched" by d. This situation can be handled by entirely analogous techniques as the case we consider here. Now for all $i \geq i_0$, A_i will have a similar form $C_i(B_{i,1}, \ldots, B_{i,m})$, and for each $j : 1 \leq j \leq m$, either $B_{i,j} = B_{i+1,j}$, or else $B_{i,j} \rightarrow B_{i+1,j}$. In addition we can assume that for infinitely many i, does $B_{i,j} \rightarrow B_{i+1,j}$, since otherwise it suffices to pick a larger i_0. Thus the proof has been reduced to showing

(i) only a finite number of distinct C_i are reachable

(ii) any derivation d that does not involve parallel composition visits a finite number of distinct agents only.

To prove (i) we introduce a new little transition system on contexts, and prove it finite. Formally, *contexts* are terms C generated by the abstract syntax

$$C ::= [\cdot] \mid (\nu x)C \mid (\lambda x)C \mid [x]C \mid C \mid C$$

Here $[\cdot]$ is the empty context. Say of a context C that *x is visible through C* if either there is some occurrence of $[\cdot]$ in C not within the scope of a binding occurrence of x, or else x occurs unbound in C. Rule (5) below shows where this notion is needed. The transition relation $\rightarrow$ is now determined in the following way where Ω ranges over operators among (νx), (λx), and $[x]$ with $x \in N_0$:

1. If C_1 and C_2 are alpha congruent then $C_1 \rightarrow C_2$
2. $[\cdot] \rightarrow \Omega[\cdot]$
3. $(\Omega C_1) \mid C_2 \rightarrow \Omega(C_1 \mid C_2)$, $C_1 \mid (\Omega C_2) \rightarrow \Omega(C_1 \mid C_2)$
4. $[x]C \rightarrow C$, $(\nu x)C \rightarrow C$, $(\lambda x)C \rightarrow C\{y/x\}$ whenever $y \in N_0$
5. $(\nu x)\Omega C \rightarrow \Omega(\nu x)C$
6. if $C_1 \rightarrow C_1'$ and x is visible through C_1' then $(\nu x)C_1 \rightarrow (\nu x)C_1'$
7. if $C_1 \rightarrow C_1'$ then $C_1 \mid C_2 \rightarrow C_1' \mid C_2$ and $C_2 \mid C_1 \rightarrow C_2 \mid C_1'$

It is easy to verify that for $i \geq i_0$, if A_i has the form $C_i(B_{i,1}, \ldots, B_{i,m})$ and A_{i+1} similarly the form $C_{i+1}(B_{i+1,1}, \ldots, B_{i+1,m})$ and for each $j : 1 \leq j \leq m$, either $B_{i,j} = B_{i+1,j}$ or $B_{i,j} \rightarrow B_{i+1,j}$, then either $C_i = C_{i+1}$ or $C_i \rightarrow C_{i+1}$. To prove (i) it therefore suffices to establish the following Lemma:

Lemma 9.4 *For all C, $\{C' \mid C \to^* C'\}$ is finite.*

Proof If $C \to^* C'$ say that C' is reachable (from C). For delimiting the reachable contexts we need the notion of legitimate prefix. First, a (context) *prefix* is a string $\Omega_1 \cdots \Omega_n$ where each Ω_i is either (νx), (λx), or $[x]$, $x \in N_0$. Write $p \cdot C$ for the context obtained by prefixing C with the prefix p. A prefix $\Omega_1 \cdots \Omega_n$ is *legitimate* if

(i) at most one Ω_i has the form either (λx) or $[x]$ for some $x \in N_0$, and

(ii) the total number of occurrences of operators of the form (νx) or (λx) for some $x \in N_0$ is at most $|N_0|$.

We can now prove Lemma 9.4 by induction in the size of C:

$C = [\cdot]$: If suffices to show that any context reachable from $[\cdot]$ has the form $p \cdot [\cdot]$ where p is a legitimate prefix. To show this assume that p is legitimate and that $p \cdot [\cdot] \to C'$. Then C' has the form $p' \cdot [\cdot]$. Clearly condition (i) above is satisfied. To see that also (ii) is satisfied suppose for a contradiction that it is not, so that p' has $|N_0| + 1$ occurrences of a binding operator. Then p' must have the form $p_1(\nu x)p_2\Omega p_3$ for some x where Ω binds x. But this cannot happen since the justification of $p \cdot [\cdot] \to p' \cdot [\cdot]$ must have appealed to rule (6) for justifying $(\nu x)p'' \cdot [\cdot] \to (\nu x)p_2\Omega p_3 \cdot [\cdot]$ for some p''. But x is not visible through $p_2\Omega p_3$—a contradiction.

$C = (\nu x)C'$: We show that any context reachable from C has the form $p \cdot C_1$ where p is a legitimate prefix and C_1 is reachable from C'. So assume that $p \cdot C_1 \to C_2$. The only case that needs considering is when p has the form $p'(\nu x)$, C_1 the form $\Omega C_1'$, and C_2 the form $p\Omega(\nu x) \cdot C_1'$. We then need to show that $p\Omega(\nu x)$ is legitimate, but this follows exactly as in the previous case.

$C = (\lambda x)C'$: The only contexts reachable from C are those reachable from $C'\{y/x\}$ for some $y \in N_0$.

$C = [x]C'$: As the previous case.

$C = C_1 \mid C_2$: We show that any context reachable from C has the form $p \cdot (C_1' \mid C_2')$ where p is legitimate, C_1' is reachable from C_1, and C_2' reachable from C_2. The only cases that need considering are applications of rule (3), but these follow as in the case for restriction above. □ (Lemma 9.4)

We then proceed to the proof of (ii).

Lemma 9.5 *Suppose that A has no occurrences of $\mid$. For all derivations $d = A_0 \to \cdots \to A_n \to \cdots$ with $A_0 = A$, $\mathcal{R}(d)$ is finite.*

Proof The proof uses the notion of legitimate prefix, introduced in the proof of Lemma 9.4, and proceeds by induction in the size of A. The cases for $\mathbf{0}$, $+$, prefixing, conditional, abstraction, application, and concretion follow directly from the induction hypothesis. This leaves two cases to be considered. For restriction the proof is a correlate of the corresponding case in the proof of Lemma 9.4. So assume that $A = \text{fix}D.A'$. We to show that any agent reachable from A has the form $p \cdot (A''[A/D])$ where p is a legitimate prefix and A'' is reachable from A', thus completing the proof by the induction hypothesis. To each transition $A_i \rightarrow A_{i+1}$ is associated a unique justification, a proof using the axioms and rules among (1)–(16) together with alpha-conversion. Say that *step i refers to A*, if the justification of the transition $A_i \rightarrow A_{i+1}$ involves an appeal to (6) with D instantiated to itself, and A to A'. Suppose now that A_i has the form $p \cdot (A''[A/D])$ such that A'' is reachable from A'. Handling the case where step i is an instance of one of the axioms (10)–(12) as in the proof of Lemma 9.4 only one potentially problematic case remains, namely where step i refers to A. This, however, can only be the case when A'' has the form $p' \cdot D$ for p' a prefix, and in this situation it must, as we have seen, be the case that the prefix pp' is legitimate. Thus A_{i+1} has been brought into the desired form. $\Box$ (Lemma 9.5) $\Box$ (Lemma 9.2)

9.2 Termination

We next use the finiteness of $\{B \mid A \rightarrow^* B\}$ to show termination, following the strategy introduced by Stirling and Walker [18] for the case of the modal μ-calculus. Define the *size*, $|\phi|$, of a formula ϕ as follows:

$$|x = y| = |x \neq y| = |X| = |U| = 1$$

$$|\phi \wedge \psi| = |\phi \vee \psi| = \max(|\phi|, |\psi|) + 1$$

$$|{<}\alpha{>}\phi| = |[\alpha]\phi| = |\sigma X.\phi| = |\lambda x.\phi| = |\phi x| = |\Sigma\phi| = |\forall\phi| = |\exists\phi| = |\phi| + 1$$

and then extend this measure to sequents $c \vdash_\Delta A : \phi$ by

$$|c \vdash_\Delta A : \phi| = \begin{cases} |\Delta(U)x_1 \cdots x_n| & \text{if } \phi \text{ is of the form } Ux_1 \cdots x_n \\ |\phi| & \text{otherwise} \end{cases}$$

Theorem 9.6 *Function* `check` *terminates on all inputs.*

Proof Consider a finite or infinite structure originating in the sequent $c_0 \vdash_{\Delta_0} A_0 : \phi_0$ and generated by a run of `check`. By Proposition 9.1 all agents occurring in the structure stay within the finite set $\{A \mid A_0 \rightarrow^* A\}$. Since each refinement step is finitely branching, to show that no infinite such proof structure can exist, by Königs Lemma it suffices to show that there can be no infinite sequences

$$s = c_0 \vdash_{\Delta_0} A_0 : \phi_0, \ldots, c_n \vdash_{\Delta_n} A_n : \phi_n, \ldots$$

such that for all $i \geq 0$, $c_i \vdash_{\Delta_i} A_i : \phi_i$ derives $c_{i+1} \vdash_{\Delta_{i+1}} A_{i+1} : \phi_{i+1}$ in one step. So assume for a contradiction that such a sequence exists. Since the size of formulas decrease strictly under all refinement steps except for those unfolding constants we can find a subsequence

$$s' = c'_0 \vdash_{\Delta'_0} A'_0 : \phi'_0, \ldots, c'_n \vdash_{\Delta'_n} A'_n : \phi'_n, \ldots$$

of s which is infinite, and for which all ϕ'_i have the form $U_i x_{i,1} \cdots x_{i,m_i}$. We show that for infinitely many i is U_i the same constant U. If it is not let i_0 be maximal such that $U_{i_0} = U_0$. Then

$$|c'_{i_0+1} \vdash_{\Delta'_{i_0+1}} A'_{i_0+1} : \phi'_{i_0+1}| < |c'_{i_0} \vdash_{\Delta'_{i_0}} A'_{i_0} : \phi'_{i_0}|$$

as $|\Delta'_{i_0+1}(U_{i_0+1} x_{i_0+1,1} \cdots x_{i_0+1,m_{i_0+1}})| < |\Delta'_{i_0}(U_{i_0} x_{i_0,1} \cdots x_{i_0,m_{i_0}})|$. Repeating, let i_1 be maximal such that $U_{i_1} = U_{i_0+1}$. Similarly,

$$|c'_{i_1+1} \vdash_{\Delta'_{i_1+1}} A'_{i_1+1} : \phi'_{i_1+1}| < |c'_{i_0+1} \vdash_{\Delta'_{i_0+1}} A'_{i_0+1} : \phi'_{i_0+1}|.$$

Thus some U must occur infinitely often among the U_i. Let then

$$s'' = c''_0 \vdash_{\Delta''_0} A''_0 : \phi''_0, \ldots, c''_n \vdash_{\Delta''_n} A''_n : \phi''_n, \ldots$$

be the (infinite) subsequence of s' for which ϕ''_i has the form $U y_{i,1} \cdots y_{i,k_i}$ for all $i \geq 0$. Since all y_{i,j_i} are chosen from the finite set N_0, and since the number of distinct A''_i is finite, and since also the number of N_0-inequivalent c''_i is finite, s'' must be finite too, a contradiction. □

9.3 Normal Termination

While Theorem 9.6 shows that `sketch` terminates on all inputs it does not follow that on all inputs `sketch` produces a well-defined truth-value. For it may be that a call of `newname(`A`,`ϕ`,`$\vec{x}$`)` is ill-defined because $N_0 \subseteq \mathrm{fn}(A) \cup \mathrm{fn}(\phi) \cup \{\vec{x}\}$. We show here that this situation can not arise. The key Lemma which needs to be proved is the following:

Lemma 9.7 *For all A_n, if $A_0 \rightarrow^* A_n$ then* $|\mathrm{fn}(A_n)| \leq \sharp_{fns}(A_0) \cdot (\sharp_{par}(A_0) + 1)$

Proof Assume that

$$d = A_0 \rightarrow \cdots \rightarrow A_n \rightarrow \cdots$$

We show $|\mathrm{fn}(A_n)| \leq \sharp_{fns}(A_0) \cdot (\sharp_{par}(A_0) + 1)$ by structural induction, using the notion of legitimate prefix introduced in the proof of 9.4. For all case except λ, ν, recursion, and parallel composition, the result follows directly from the induction hypothesis, so only these four are considered:

$A_0 = (\lambda x)A'_0$. If $n > 0$ it must be the case that (up to an initial sequence of alpha-conversions) $A_0 \rightarrow A_1$ is an instance of (4), i.e. that $A_1 = A'_0$, so that $A'_0 \rightarrow^* A_n$. Then by the induction hypothesis,

$$\begin{aligned} |\mathrm{fn}(A_n)| &\leq \sharp_{fns}(A'_0) \cdot (\sharp_{par}(A'_0) + 1) \\ &= \sharp_{fns}(A_0) \cdot (\sharp_{par}(A_0) + 1) \end{aligned}$$

$A_0 = (\nu x)A'_0$. It must be the case that A_n has the form $p \cdot A'_n$ for p a legitimate prefix, that $A'_0 \rightarrow p' \cdot A'_n$ for some p' and A'_n, and that either p and p' are identical, or else p differs from p' only in that it (up to possible alpha-conversions of the bound name x) has an occurrence of (νx). In either case the result is immediate by the induction hypothesis.

$A_0 = A_{0,1} \mid A_{0,2}$. In this case A_n has the form $p \cdot (A_{n,1} \mid A_{n,2})$ for some legitimate prefix p. Then for each $i \in \{1,2\}$ we find legitimate prefixes p_i such that $A_{0,i} \rightarrow^* p_i \cdot A_{n,i}$, and p is the merge of p_1 and p_2 in a manner such that if $[x]$ occurs in p with x in a bound position then so it does in whichever p_i that contains $[x]$. By the induction hypothesis,

$$|\mathrm{fn}(p_i \cdot A_{n,i})| \leq \sharp_{fns}(A_{0,i}) \cdot (\sharp_{par}(A_{0,i}) + 1)$$

for $i = 1$ and $i = 2$. Now

$$\begin{aligned} |\mathrm{fn}(A_n)| &\leq |\mathrm{fn}(p_1 \cdot A_{n,1})| + |\mathrm{fn}(p_2 \cdot A_{n,2})| \\ &\leq \sharp_{fns}(A_{0,1}) \cdot (\sharp_{par}(A_{0,1}) + 1) + \sharp_{fns}(A_{0,2}) \cdot (\sharp_{par}(A_{0,2}) + 1) \end{aligned}$$

Let B be whichever of $A_{0,1}/A_{0,2}$ such that $\sharp_{fns}(B)$ is maximal. Then

$$\begin{aligned} &\sharp_{fns}(A_{0,1}) \cdot (\sharp_{par}(A_{0,1}) + 1) + \sharp_{fns}(A_{0,2}) \cdot (\sharp_{par}(A_{0,2}) + 1) \\ &\quad \leq \sharp_{fns}(B) \cdot (\sharp_{par}(A_{0,1}) + \sharp_{par}(A_{0,2}) + 2) \\ &\quad = \sharp_{fns}(B) \cdot (\sharp_{par}(A_0) + 1) \\ &\quad \leq \sharp_{fns}(A_0) \cdot (\sharp_{par}(A_0) + 1) \end{aligned}$$

completing the case.

$A_0 = \mathrm{fix} D.A'_0$. Since $\sharp_{par}(A_0) = 0$ by the assumption of finite control it suffices to show that $|\mathrm{fn}(A_n)| \leq \sharp_{fns}(A_0)$. We then find legitimate prefixes p and p' such that

$$\mathrm{fix} D.A'_0 \rightarrow^* p \cdot \mathrm{fix} D.A \rightarrow^* A_n,$$

A_n has the form $p' \cdot A'_n[\mathrm{fix} D.A/D]$, and $A'_0 \rightarrow^* A'_n$. Note that we can assume that p has no free occurrences of names since if it had, before $\mathrm{fix} D.A'_0$ would be subsequently unfolded, the free name (occurring in an output prefix) would be eliminated by an application of (7). By the induction hypothesis we know that

$$\begin{aligned} |\mathrm{fn}(A'_n)| &\leq \sharp_{fns}(A'_0) \\ &= \sharp_{fns}(A_0) \end{aligned}$$

The only case in which $|\mathrm{fn}(A_n)|$ could be greater than $|\mathrm{fn}(A'_n)|$ is when p' contains an occurrence of an output prefix $[y]$ such that the occurrence of y in $[y]$ is free in p'. This, however, can only happen if we can factorise the derivation $p \cdot \mathrm{fix} D.A'_0 \rightarrow^* A_n$ as follows

$$p \cdot \mathrm{fix} D.A'_0 \rightarrow^* p'' \cdot A''_n[\mathrm{fix} D.A'_0/D] \rightarrow^* A_n$$

such that p'' has no free occurrence of y (i.e. A_n results from $A''_n[\mathrm{fix} D.A'_0/D]$ by applications of 8, 9, and 11), and such that $A'_0 \rightarrow^* A''_n$. Moreover $\mathrm{fn}(A''_n) = \mathrm{fn}(A'_n) \cup \{y\}$. But then, since we know by the induction hypothesis that $|\mathrm{fn}(A''_n)| \leq \sharp_{fns}(A_0)$ the proof is complete. □

Normal termination is now an easy Corollary:

Corollary 9.8 *Function* `check` *terminates normally on all inputs.*

Proof Use Lemma 9.7. $\square$

9.4 Well-definedness and Soundness

It remains to check that the proof structure induced by an invocation of `check`$(c \vdash_\Delta A : \phi)$ is indeed a valid proof structure according to the proof rules of fig. 4.

Lemma 9.9 *On all inputs* `check` *determines a well-defined proof structure according to the local and global proof rules.*

Proof It suffices to observe, as in the proof of Proposition 9.1 that if A is a process and $A \succ_c^- \alpha.B$ then $A \to^* \mathtt{normalform}(B, c)$. $\square$

Thus:

Corollary 9.10 (Soundness of `check`) *If* `check`$(c \vdash_\Delta A : \phi)$ *returns the value* `true` *then the sequent* $c \vdash_\Delta A : \phi$ *is true.*

Proof By Lemma 9.9 `check`$(c \vdash_\Delta A : \phi)$ determines a well-defined proof structure. A simple inductive arguments shows that if the value returned is `true` then all hypotheses of the induced proof have been discharged. But then $c \vdash_\Delta A : \phi$ by 7.1. $\square$

10 Completeness and Decidability

For completeness it now only remains to check that if a sequent is true then the set of proof structures determined by an invocation of `check` on that sequent has a member with no undischarged occurrences of assumptions. The proof of this follows the approach of Streett and Emerson [19].

Theorem 10.1 (Completeness) *If the sequent* $c \vdash_\Delta A : \phi$ *is true then it is derivable.*

Proof Suppose $A \in \|\phi_\Delta\|_s c$ and assume given a set of proof structures induced by an invocation of `check`$(c \vdash_\Delta A : \phi)$. This set is well-defined and each of its members are well-defined as proof structures generated by the local and global proof rules. We show that at least one member of this set will have no undischarged occurrences of hypotheses.

Let $U_1, \ldots, U_n$ be the sequence of μ-constants of Δ in order of definition. Each n-length string $w = \alpha_1, \ldots, \alpha_n$ of ordinals determines the definition list Δ_w that coincides with Δ on all ν-constants, and for each U_i, $1 \leq i \leq n$, if $\Delta(U_i) = \mu X_i.\phi_i$, say, then $\Delta_w(U_i) = \mu^{\alpha_i} X_i.\phi_i$. The *signature* of $c \vdash_\Delta A : \phi$, $W(c \vdash_\Delta A : \phi)$, is then the lexicographically least string w such that $A \in \|\phi_{\Delta_w}\|_s c$. Using ordinal approximations it is clear that if $c \vdash_\Delta A : \phi$ is true then it has a well-defined signature.

We explain how to find a candidate proof of $c \vdash_\Delta A : \phi$ in stages. At each stage we keep track of a *current set of candidate proof structures*, $\mathcal{P}$, and a *current set of sequents* (or more precisely, sequent occurrences), $\mathcal{S}$, to be further refined. The set $\mathcal{S}$ has the property that no sequent in $\mathcal{S}$ occurs above another. Then $\mathcal{P}$ has the property that the subproofs obtained from each proof in $\mathcal{P}$ by restricting attention to sequents above the root and not above a sequent in the $\mathcal{S}$, are identical.

Initially $\mathcal{P}$ is the entire set of proof structures determined by an invocation of `check2(`$c \vdash_\Delta A : \phi$`)`, and $\mathcal{S}$ is the singleton $\{c \vdash_\Delta A : \phi\}$. We explain how to complete stage n. Pick a member of $\mathcal{S}$, say $c' \vdash_{\Delta'} A' : \phi'$. Suppose that `check2(`$c' \vdash_{\Delta'} A' : \phi'$`)` does not recurse. Then the search is finished since either ϕ' is an equation or an inequation which must be provable since it is true, or else ϕ' is a constant, and we will then have to prove that ϕ' is a ν-constant so that $c' \vdash_{\Delta'} A' : \phi'$ is a discharged occurrence of a hypothesis. In either case $c' \vdash_{\Delta'} A' : \phi'$ is removed from the $\mathcal{S}$, and we proceed to stage $n+1$. So assume instead that `check2(`$c' \vdash_{\Delta'} A' : \phi'$`)` does recurse. Choose then a maximal subset of $\mathcal{P}$ such that

1. $c' \vdash_{\Delta'} A' : \phi'$ is the conclusion of the same rule instance.
2. The antecedents of $c' \vdash_{\Delta'} A' : \phi'$ have minimal signatures.

Note that either the rule instance is determined, or else (in the cases for disjunctions or diamonds) each potential rule instance has only one antecedent. It follows from the symbolic semantics that a nonempty subset with these properties can be chosen. Having made the choice $\mathcal{P}$ is replaced by the chosen subset, and $\mathcal{S}$ is updated by replacing $c' \vdash_{\Delta'} A' : \phi'$ by its antecedents.

The result is a well-defined proof structure. Moreover, the only undischarged occurrences of sequents in this proof structure are sequents of the form $c' \vdash_{\Delta'} A' : Ux_1 \cdots x_m$ where U is a μ-constant. We need to show that no such sequents can occur. So assume that the resulting proof has an undischarged occurrence of $c' \vdash_{\Delta'} A' : Ux_1 \cdots x_m$. This means that we find an earlier visited sequent of the form $c'' \vdash_{\Delta''} A' : Ux_1 \cdots x_m$ with $\models c'' \supset c'$. We show that $W(c' \vdash_{\Delta'} A' : Ux_1 \cdots x_m) < W(c'' \vdash_{\Delta''} A' : Ux_1 \cdots x_m)$. Suppose that U is the n'th μ-constant in order of definition. Then the initial refinement step applied to $c'' \vdash_{\Delta''} A' : Ux_1 \cdots x_m$, the rule FOLD, strictly decreases signature in its n'th position. No subsequent refinement step can increase signature in positions smaller than or equal to n. Thus

$$W(c' \vdash_{\Delta'} A' : \phi') < W(c'' \vdash_{\Delta''} A' : \phi').$$

But since $\models c'' \supset c'$, $W(c'' \vdash_{\Delta'} A' : \phi') \leq W(c' \vdash_{\Delta'} A' : \phi')$. Moreover, $W(c'' \vdash_{\Delta'} A' : \phi') = W(c'' \vdash_{\Delta''} A' : \phi')$ since the two sequents differ only in constants that are no longer reachable. But this is a contradiction. $\square$

Decidability, then, is an immediate Corollary.

Corollary 10.2 *Derivability of sequents is decidable.*

PROOF By termination, soundness and the proof of the Completeness Theorem we know that a sequent is derivable if and only if the application of the model checking algorithm to that sequent results in the value "true". □

11 Conclusion and Related Work

Algorithms for value passing process calculi have been considered recently by a number of authors. For bisimulation equivalence Jonsson and Parrow [7] have considered data-independent programs, and Hennessy and Lin [5] have presented an algorithm for a certain class of "standard" symbolic transition graphs. Applied to the π-calculus both these classes are strictly weaker than the notion of finite control agent introduced in the present paper. In particular we avoid the technical conditions that prohibit reuse of variables in the algorithm of Hennessy and Lin. Note that it is likely that our model checker can be used for deciding bisimulation equivalence of finite control agents via a notion of characteristic formula (c.f. [3]). A closely related proof system for a version of Hennessy-Milner logic adapted to value-passing has been introduced by Hennessy and Liu [6]. Parts of our proof system appear originally in their work: The structural rules, and the rules for boolean connectives, $\forall$ and $\exists$. However, they fail to consider fixed points or other temporal operators, and thus their logic is far too weak to be of any practical interest. Indeed it is the handling of just recursively defined agents and properties in the presence of name passing and generation which forms the main contribution of the present paper. Other significant differences concern our choice of basic connectives and our focus on the π-calculus.

Many issues related to the work reported here needs to be further examined. More consideration is needed from both practical and theoretical perspectives of the features required from a temporal logic along the lines of the one we describe. Relations to the π-calculus encodings of data types, lambda calculus, and the higher order π-calculus should be investigated. The efficiency and usability of our proof system and decision procedure needs to be evaluated on practical examples. Mechanisms for compositional verification should be developed, perhaps along the lines of Stirling [16], or Andersen and Winskel [1]. Concerning early bisimulation equivalence a temporal logic characterising this equivalence instead of late bisimulation equivalence can be devised using the basic modalities of e.g. [12] in place of those considered here. We envisage no significant problems in obtaining similar results for such a logic.

Acknowledgements

Thanks are due to Joachim Parrow, Lars-Åke Fredlund, and Björn Victor for discussions on the π-calculus.

References

[1] H. Andersen and G. Winskel. Compositional checking of satisfaction. In *Proc. 3rd Workshop on Computer Aided Verification*, pages 49–65, July 1991.

[2] R. Cleaveland, J. Parrow, and B. Steffen. A semantics based verification tool for finite state systems. In *Proc. 9th IFIP Symp. Protocol Specification, Verification and Testing*, 1989.

[3] R. Cleaveland and B. Steffen. Computing behavioural relations, logically. In *Proc. ICALP'91*, 1991.

[4] M. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. In *Proc. 17th Coll. Trees in Algebra and Programming,* Lecture Notes in Computer Science, 581:145–164, 1992. To appear in Theoretical Computer Science.

[5] M. Hennessy and H. Lin. Symbolic bisimulations. Dept. of Computer Science, University of Sussex, Report 1/92, 1992.

[6] M. Hennessy and X. Liu. A modal logic for message passing processes. Dept. of Computer Science, University of Sussex, Report 3/93, 1993.

[7] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Information and Computation*, 1992.

[8] K. G. Larsen. Proof systems for Hennessy-Milner logic with recursion. in *Proc. 13th CAAP* Lecture Notes in Computer Science, 299, 1988.

[9] R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[10] R. Milner. The polyadic π-calculus: A tutorial. Technical Report ECS-LFCS-91-180, Laboratory for the Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1991.

[11] R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.

[12] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. Technical Report R91:03, SICS, 1991. To appear in *Theoretical Computer Science.*

[13] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40 and 41–77, 1992.

[14] F. Orava and J. Parrow. An algebraic verification of a mobile network. *Formal Aspects of Computing*, pages 497–543, 1992.

[15] D. Sangiorgi. From π-calculus to higher-order π-calculus—and back. To appear in Proc. TAPSOFT'93, 1993.

[16] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.

[17] C. Stirling. Modal and temporal logics for processes. Technical Report ECS-LFCS-92-221, LFCS, Dept. of Computer Science, University of Edinburgh, 1992.

[18] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

[19] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, **81**:249–264, 1989.