Sequent Calculi for Process Verification: Hennessy-Milner Logic for an Arbitrary GSOS

Citation for published version:
Simpson, A 2004, 'Sequent Calculi for Process Verification: Hennessy-Milner Logic for an Arbitrary GSOS', *Journal of Logic and Algebraic Programming*, vol. 60-61, pp. 287-322.
https://doi.org/10.1016/j.jlap.2004.03.004

Digital Object Identifier (DOI):
10.1016/j.jlap.2004.03.004

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Journal of Logic and Algebraic Programming

Sequent Calculi for Process Verification: Hennessy-Milner Logic for an Arbitrary GSOS

Alex Simpson [1]

LFCS, School of Informatics, University of Edinburgh, Scotland

Abstract

We argue that, by supporting a mixture of "compositional" and "structural" styles of proof, sequent-based proof systems provide a useful framework for the formal verification of processes. As a worked example, we present a sequent calculus for establishing that processes from a process algebra satisfy assertions in Hennessy-Milner logic. The main novelty lies in the use of the operational semantics to derive introduction rules, on the left and right of sequents, for the operators of the process calculus. This gives a generic proof system applicable to any process algebra with an operational semantics specified in the GSOS format. Using a general algebraic notion of GSOS model, we prove a completeness theorem for the cut-free fragment of the proof system, thereby establishing the admissibility of the cut rule. Under mild (and necessary) conditions on the process algebra, an ω-completeness result, relative to the "intended" model of closed process terms, follows.

Key words: Formal verification, Compositionality, Process algebra, Structural operational semantics, Sequent calculus

1 Introduction

This paper is concerned with the construction of proof systems for the formal verification of programs, specifically of concurrent processes. The main thesis is that Gentzen's sequent calculus [14] provides an ideal foundation upon which

[1] Research supported by an EPSRC Postdoctoral Research Fellowship (1994–1995), an EPSRC Advanced Research Fellowship (2001–) and a visiting professorship at RIMS, Kyoto University (2002–2003).

to base such systems. This is illustrated by a substantial worked example: a sequent-based proof system for establishing that processes with operational semantics specified in the GSOS format [5] satisfy properties of Hennessy-Milner logic [16]. We end the paper with a discussion of how the approach extends to other programming paradigms and to more powerful logics.

There are several desirable properties that one might require of any proof system for program verification. First, concerning the basic logical properties of the system:

Soundness: everything provable is true. We take this as the *sine qua non* of formal verification.

Completeness: everything true is provable. For expressive programming languages and logics this will be unachievable. Nevertheless, it is vital that the proof system is sufficiently complete to establish verification goals that occur in practice. Moreover, informative restricted completeness theorems may be available as a mathematical indication of the power of the system.

Of practical importance is that the proof system should permit useful methods of reasoning. We identify three independent requirements here.

Compositional reasoning. Often, in order to verify that a compound program satisfies a property, one would like to verify that its component subprograms independently satisfy properties that are together sufficient to establish the original goal. For example, to verify that a parallel composition $p|q$ satisfies a property A, one might verify that p satisfies some property B_1 and that q satisfies some property B_2, where these two facts together imply that $p|q$ indeed satisfies A. The importance of such *compositional* reasoning is that it allows the verification task to be split up into independent goals that can be verified separately. This possibility provides a foundation for the modular development and verification of software.

Structural reasoning. It should be possible to verify a goal by breaking the goal down into subgoals obtained via a canonical decomposition of the original goal based on its syntactic structure. Such *structural* methods support a natural goal-directed approach to proof construction. They are thus important in the provision of proof support by proof assistants and theorem provers, and especially vital to the efficient implementation of proof search algorithms.

Natural reasoning. In addition, one would like the proof system to support natural and intuitive methods of reasoning. Ideally, it should be possible for a formal proof of correctness to closely adhere to the natural informal argument justifying correctness

We believe that it is important for a proof system to fulfill all three requirements. Indeed, significant though compositional methods undoubtedly are, there is no reason at all to require every proof step to be compositional. Compositionality should be used at natural points, where a program divides into modules, and where the independent verification of these modules is desirable. But, within the verification of any individual module itself, it may well be useful to maintain logical dependencies between subcomponents, and these dependencies might be less easily expressed if compositional methods were enforced throughout the verification process. Furthermore, there is no general guarantee that compositional reasoning is always applicable. For example, there is no reason for it to always be possible to reduce a goal "$p|q$ satisfies A" to two independent goals of the form "p satisfies B_1" and "q satisfies B_2". Thus, although it is essential for a proof system to support compositional reasoning, this should not be the only method of verification permitted.

Finally, another general concern is that, preferably, the proof system should be derived from the programming language and logic in a principled way. In such cases there is better chance of being able to adapt or extend the system to deal with new programming language features and different logical primitives.

We now present a very general argument that sequent calculus provides an excellent foundation for the development of proof systems with the above properties. Suppose we have a language for expressing processes (or programs), and a logic for expressing properties of them. Then the basic relation of interest is the *satisfaction relation* between processes and properties: process p satisfies property A. We outline the potential virtues of having a proof system based on sequents of the form $\Gamma \implies \Delta$ where Γ and Δ are sets of *assertions*, including a basic assertion form, $p\colon A$, expressing the above satisfaction relation. As usual, a sequent $\Gamma \implies \Delta$ should be understood as expressing the implication: if all the assertions in Γ are true then so is at least one of the assertions in Δ.

First, the sequent-based formalism is rich enough to express diverse types of verification goal.

Ordinary goals. The ordinary verification goal, of establishing that process p satisfies property A, is expressed by the sequent $\implies p\colon A$. The verification task is then to construct a proof of this sequent.

Parametrized goals. By allowing variables ranging over processes in assertions, one can express parametrized verification goals of the form

$$x_1\colon B_1, \ldots, x_n\colon B_n \implies p(x_1, \ldots, x_n)\colon A$$

This sequent states that if the parameters $x_1, \ldots, x_n$ in p are instantiated with processes $q_1, \ldots, q_n$ satisfying $B_1, \ldots, B_n$ respectively, then the resulting compound process $p(q_1, \ldots, q_n)$ satisfies A.

Further, sequent-calculus-based proof systems address the various stylistic requirements concerning reasoning methods.

Compositional reasoning. Parametrized verification goals can be used to support compositional reasoning. By combining the familiar cut and substitution rules from sequent calculus, one obtains derived rules of the form

$$\frac{\Longrightarrow q_1 : B_1 \quad \dots \quad \Longrightarrow q_n : B_n \qquad x_1 : B_1, \dots, x_n : B_n \Longrightarrow p(x_1, \dots, x_n) : A}{\Longrightarrow p(q_1, \dots, q_n) : A}$$

The above rule reduces the goal of showing that a compound process $p(q_1, \dots, q_n)$ satisfies property A to individual subgoals for its component subprocesses $q_1, \dots, q_n$, together with an additional parametrized goal required to justify the choice of $B_1, \dots, B_n$. This approach to compositionality was proposed by Stirling in [25], who presented a proof calculus based on primitive decomposition rules of this form for CCS parallel compositions $q_1|q_2$. In our approach, such rules arise automatically as a consequence of having a sequent-based proof system allowing the expression of general parametrized verification goals.

Structural reasoning. The primitive proof rules of sequent calculus are introduction rules, on the left and right of sequents, for logical connectives. Such proof rules exactly support a structural, goal-directed form of reasoning. Moreover, if a cut-elimination theorem is available then structural reasoning is sufficient for establishing any provable goal.

Natural reasoning. Each primitive proof rule of sequent calculus embodies in a direct way the meaning of the logical connective that the rule represents. This feature makes it plausible that natural informal proofs that a process satisfies a property, whose primitive steps should all be self-explanatory, might have close formal analogues. Indeed, a wide body of research, undertaken using the many proof assistants based on sequent calculi, suggests that such systems do allow direct formalizations of natural arguments, modulo the minor convolution of writing proofs in a goal-directed sequent style, rather than in a natural deduction style.

There is one major issue, however, that has not been addressed in the discussion above. We have argued that the compositional, structural and naturalness aspects of sequent-based proof follow from properties of the basic sequent calculus rule set, including "structural rules" [2] (e.g. cut) and logical rules.

[2] There is a slightly unfortunate clash between our use of "structural reasoning" and the sequent calculus notion of "structural rule". For us, structural reasoning is implemented by the non-structural rules of sequent calculus. Our terminology is chosen to be consistent with the sense of "structural" in "structural operational semantics", in which the premises of a rule are obtained by a similar syntactic decomposition of the conclusion.

However, we are here envisaging an applied sequent calculus with sequents composed of verification assertions, rather than a pure logical calculus. Such a system cannot be based on logical rules alone. One also needs rules to relate processes (or programs) to their logical properties. The question thus arises as to whether it is possible to provide such applied proof systems without breaking the fundamental structural properties of sequent calculus.

In this paper we show that this is indeed possible, at least for processes with an operational semantics specified in the GSOS format, and for propositional modal logic (Hennessy-Milner logic [16]). Our method of approach concerns adding introduction rules, on the left and right of sequents, for process operators, in addition to the standard rules for the logical connectives. These proof rules for process operators are derived in a principled way directly from the operational semantics. Thus the approach also provides a modular proof system, easily adaptable to a range of process algebras.

At the end of the paper we include an epilogue discussing work that has been done, since the research in this paper was first carried out, towards adapting our approach to richer program logics and other programming paradigms.

2 Proof rules for modalities and process operators

In this section we present an informal introduction to our approach of incorporating proof rules for modalities and process operators into the sequent calculus. A detailed technical treatment is given in Section 4.

As motivated in Section 1, the proof system is a sequent calculus with sequents of the form $\Gamma \implies \Delta$ where Γ and Δ are finite sets of *assertions*. Our main assertion form in $p\colon A$, where p is term representing a process and A is a formula of Hennessy-Milner logic [16]. For illustrative purposes, we use CCS [18] as the process language in this section. As discussed in Section 1, we allow process terms to contain free process variables.

The task we address is how to give proof rules for the logic and for the process operators. We consider each issue in turn.

For the formulas of Hennessy-Milner logic we need proof rules both for the propositional connectives and for the modalities. The rules for the former are standard. For the modalities, we give rules which reflect in as direct a way as possible their meanings. For example, in the case of the necessity modality, we have that p satisfies $[a]A$ (where a is some action) if and only if, for every process q such that p can perform a to become q (notation $p \xrightarrow{a} q$), it holds that q satisfies A. In order to translate this in terms of primitive rules it is necessary to have a further assertion form expressing that $p \xrightarrow{a} q$ for processes p and q. Then one has natural rules:

$$\frac{\Gamma \implies p \xrightarrow{a} q,\, \Delta \quad \Gamma,\, q\colon A \implies \Delta}{\Gamma,\, p\colon [a]A \implies \Delta} \qquad \frac{\Gamma,\, p \xrightarrow{a} x \implies x\colon A,\, \Delta}{\Gamma \implies p\colon [a]A,\, \Delta} \tag{1}$$

where, in the right-hand rule, x is a variable (ranging over processes) that does not appear in the concluding sequent of the rule, thus x represents an arbitrary process to which p can evolve via a.

The rules for process operators are derived from the operational semantics of the process algebra, making crucial use of the presence of $p \xrightarrow{a} q$ assertions. Indeed, the right-hand rules are copied directly from the operational semantics. For example, the rules for the CCS prefix and sum operators [18] are:

$$\frac{}{\Gamma \implies a.p \xrightarrow{a} p,\ \Delta} \qquad \frac{\Gamma \implies p \xrightarrow{a} p',\, \Delta}{\Gamma \implies p + q \xrightarrow{a} p',\, \Delta} \qquad \frac{\Gamma \implies q \xrightarrow{a} q',\, \Delta}{\Gamma \implies p + q \xrightarrow{a} q',\, \Delta}$$

The rules for introducing process operators on the left express that an action $f(p_1, \ldots, p_k) \xrightarrow{a} r$ may only happen if it is derivable via one of the operational rules for f. For example, for the prefix, zero and sum operators of CCS, this property is expressed by the following rules:

$$\frac{\Gamma[p, r] \implies \Delta[p, r]}{\Gamma[r, p],\, a.p \xrightarrow{a} r \implies \Delta[r, p]} \qquad \frac{}{\Gamma,\, a.p \xrightarrow{b} r \implies \Delta}\ a \neq b$$

$$\frac{}{\Gamma,\, 0 \xrightarrow{a} r \implies \Delta} \qquad \frac{\Gamma,\, p \xrightarrow{a} r \implies \Delta \quad \Gamma,\, q \xrightarrow{a} r \implies \Delta}{\Gamma,\, p + q \xrightarrow{a} r \implies \Delta}$$

Here we write $\Gamma[p, r]$ for $\Gamma[p/x, q/y]$, where neither x nor y occur in p and r. Then $\Gamma[r, p]$ is simply $\Gamma[r/x, p/y]$. Equivalently, one can understand $\Gamma[p, r]$ as simply a set of assertions with some (but not necessarily all) occurrences of p and q highlighted, in which case $\Gamma[r, p]$ is then obtained by replacing the distinguished occurrences of p with r, and vice versa. Incidentally, we have not mentioned a right-hand rule for zero because it does not have one.

All the rules we have discussed so far have the properties we identified in Section 1 as being desirable of structural reasoning. In the modality rules, the formula $[a]A$ in the rule conclusion is decomposed to the formula A in the premise. In the operational rules, a conclusion involving a process $f(p_1, \ldots, p_k)$ is derived from premises mentioning only its arguments $p_1, \ldots, p_k$.

One would like some further assurance that the left and right rules for process operators are well chosen. One yardstick by which this can be judged, is whether they properly complement each other in the sense of supporting local cut-elimination steps. Unfortunately, the rules formulated above do not support such proof reductions. For example, the following derivation just uses the above process rules and cut:

$$\dfrac{\dfrac{b.0 \xrightarrow{c} 0 \implies}{c.0 \xrightarrow{c} 0,\ a.b.0 \xrightarrow{a} c.0 \implies} \qquad \dfrac{}{\implies c.0 \xrightarrow{c} 0}}{a.b.0 \xrightarrow{a} c.0 \implies}\ \text{cut}$$

but there is no way of eliminating cut from the derivation. The sequents $a.b.0 \xrightarrow{a} x,\ a.c.0 \xrightarrow{a} x \implies$ and $a.b.0 + c.d.0 \xrightarrow{a} x,\ a.b.0 + c.d.0 \xrightarrow{c} x \implies$ give other examples of similar phenomena. This failure of cut-elimination does not seem to be a result of the particular formulation of the rules, but rather an unavoidable problem for the sequents considered above. As seems desirable, all the rules are sound (in a sense explained in Section 4) relative to models in which bisimilar processes are identified. So the only way to show the impossibility of $a.p \xrightarrow{a} q$ is to show that p and q are not bisimilar. This involves considering the hereditary behaviour of p and q, and a cut is required to effect such an argument.

Rather than changing the rules, we address the non-eliminability of cut by restricting the class of sequents. Serendipitously, it turns out that if one imposes certain natural structural requirements on sequents (excluding, amongst others, the sequents above) then a full cut-elimination result holds for the proof system. This means that structural reasoning is alone sufficient for establishing any goal. Of course this result in no way devalues the importance of compositional reasoning, which does require cut. The situation is analogous to that for ordinary logic, where the cut rule is essential for structuring proofs using lemmas, and where the eliminability of cut in no way undermines the usefulness of lemmas in proofs.

In Section 4, we present our approach in detail. There, we formulate a sequent calculus for any process algebra whose operational semantics is specified in the GSOS format [5]. The choice of GSOS format is motivated by its wide expressivity. As is argued in [5], the GSOS format apparently forms the largest class of operational rule that enjoys certain basic sanity properties. In particular, GSOS rules generate transition systems with finite image, and strong bisimulation is a congruence relative to any GSOS operator. In this paper, a further benefit of the use of GSOS systems is that their generic rule format allows us to explicitly exhibit the uniformity in our method of deriving sequent calculus proof rules from operational semantics.

The main result of Section 4 is a completeness theorem for the sequent calculus (Theorem 2), relative to a natural algebraic notion of GSOS model introduced in Section 3. The completeness theorem is for a cut-free proof system. Thus the admissibility of the cut rule is obtained as a corollary of completeness

(Corollary 1). Proving completeness is the central technical task of the paper, and Section 5 is devoted to this.

For the purpose of process verification, one is interested in having completeness relative to the "intended" model, given by the process calculus itself, rather than relative to a general class of models. In Section 6, we show that, for certain sequents, the proof system is indeed complete for deriving truth in the intended model (Theorem 3). Furthermore, we give necessary and sufficient conditions for an ω-completeness result to hold (Theorem 4). The latter result implies, as a special case, that the sequent calculus is complete for deriving parametrized verification goals in the sense of Section 1.

3 Technical preliminaries

This section provides the technical background for the rest of the paper. We review the GSOS rule format for specifying the operational semantics of processes [5], we introduce a general algebraic notion of *GSOS model*, we recall the stratified definition of strong bisimilarity, we review Hennessy-Milner logic and its relationship to bisimilarity [16], and we establish basic properties of the class of finite processes.

First some notational and terminological preliminaries. Given a binary relation R, we write R^+ for its transitive closure. We say that R is *well-founded* if there is no infinite sequence (x_i) with $x_{i+1} R x_i$ for all i. Given a set X we write $\mathcal{P}(X)$ for the powerset of X, and $\mathcal{P}_{\text{fin}}(X)$ for the finite powerset. We say that a property holds for *almost all* natural numbers, to mean that it holds for all but finitely many numbers.

We use $x, y, z, \ldots$ to range over a countably infinite set, Vars, of *process variables*. We use $f, g, \ldots$ to range over a countable set of *operator symbols* each of which has an associated arity ≥ 0. We use $p, q, r, \ldots$ to range over *process terms* built from the operators and variables in the standard way, respecting arities. We write $r(\vec{x})$ to mean that all the variables of r are contained in the vector of distinct variables $\vec{x}$; in which case, given a vector of process terms, $\vec{p}$, of the same length as $\vec{x}$, we write $r(\vec{p})$ for the process term obtained by the evident substitution. We write $\text{Vars}(p)$ for the set of variables appearing in p. We say that p is *closed* if $\text{Vars}(p) = \emptyset$. For $V \subseteq \text{Vars}$, we write Terms_V for the set of all terms p with $\text{Vars}(p) \subseteq V$.

A *substitution function*, σ, is a total function from variables to process terms. We also write σ for the unique homomorphism on process terms determined by a substitution function. Thus $\sigma(p)$ means the term obtained by substituting $\sigma(x)$ for each variable x in p.

The operational semantics is to be specified by a GSOS system [5]. We use $a, b, c, \ldots$ to range over a *finite* set, Act, of *actions*. A *GSOS rule* has the form:

$$\frac{\{x_i \xrightarrow{a_{ij}} y_{ij}\}_{1\leq j\leq m_i}^{1\leq i\leq k} \quad \{x_i \overset{b_{ij}}{\not\rightarrow}\}_{1\leq j\leq n_i}^{1\leq i\leq k}}{f(x_1,\ldots,x_k) \xrightarrow{c} r(\vec{x},\vec{y})} \tag{2}$$

where: all the variables are distinct; $\vec{x}$ and $\vec{y}$ are the vectors of all x_i and y_{ij} variables respectively; $m_i, n_i \geq 0$; and k is the arity of f. We say that f is the *operator* of the rule and c is its *action*. A *GSOS system*, $\mathcal{R}$, is given by a set of GSOS rules containing, for each operator-action pair f, c, only a finite number of rules with operator f and action c. Henceforth, we assume given a fixed GSOS system, $\mathcal{R}$.

Normally a GSOS system is used to determine a labelled transition system between closed processes giving their operational behaviour. We shall be interested in this transition system as one intended model amongst a wider class of models, see Definition 3.1 below. First, some preliminary definitions. A *(labelled) transition system* is a structure of the form $T = (|T|, \{\xrightarrow{a}_T\}_{a\in\mathrm{Act}})$ where: $|T|$ is a set (of *states*); and $\xrightarrow{a}_T$ is a binary relation on $|T|$ for each action a. We use $s, t, \ldots$ to range over states of transition systems, and we often write $s \in T$ rather than $s \in |T|$. We write $s \overset{a}{\not\rightarrow}_T$ if there does not exist t such that $s \xrightarrow{a}_T t$. For $s \in T$ define $\mathrm{succ}_a(s) = \{t \mid s \xrightarrow{a}_T t\}$. We say that T is *image finite* if, for each state $s \in T$, the set $\bigcup_{a\in\mathrm{Act}} \mathrm{succ}_a(s)$ is finite.

A *premodel* is a structure $T = (|T|, \{\xrightarrow{a}_T\}, \{f_T\})$ where: $(|T|, \{\xrightarrow{a}_T\})$ is a transition system; and f_T is a k-ary function on $|T|$ for each operator f of arity k. Given premodels S, T, we say a binary relation $R \subseteq |S| \times |T|$ is a *generalized congruence* if, for all $s_1, \ldots s_k \in |S|$ and $t_1, \ldots, t_k \in |T|$, and k-ary f, it holds that $R(f_S(s_1,\ldots,s_k), f_T(t_1,\ldots,t_k))$ whenever $R(s_i, t_i)$ for all $i \in \{1,\ldots,k\}$.

Given a premodel T, an *environment* is a function from process variables to $|T|$. Given an environment γ, we write $\gamma[x \mapsto t]$ for the environment obtained from γ by updating the value at x to t. An environment γ induces a function mapping each process term p to a state $\gamma(p) \in |T|$, defined inductively by

$$\gamma(f(p_1,\ldots,p_k)) = f_T(\gamma(p_1),\ldots,\gamma(p_k)),$$

for each operator f. Clearly $\gamma(p)$ depends only on the values taken by γ on $\mathrm{Vars}(p)$. Often, rather than dealing with environments directly, we shall more conveniently write $p(t_1,\ldots,t_k)$ for the value $\gamma(p)$ of a term $p(x_1,\ldots,x_k)$ in any environment γ with $\gamma(x_1) = t_1$ and $\ldots$ and $\gamma(x_k) = t_k$.

Definition 3.1 (GSOS model) We say that a premodel T is a *model* if it holds that $f(s_1,\ldots,s_k) \xrightarrow{c}_T s'$ if and only if there exists a rule in $\mathcal{R}$, of the form (2) above, and there exist states $\{t_{ij}\}_{1\leq i\leq k}^{1\leq j\leq n_i}$ such that:

(1) for all i, j with $1 \leq i \leq k$ and $1 \leq j \leq m_i$, it holds that $s_i \xrightarrow{a_{ij}}_T t_{ij}$;
(2) for all i, j with $1 \leq i \leq k$ and $1 \leq j \leq n_i$, it holds that $s_i \overset{b_{ij}}{\not\rightarrow}_T$; and
(3) $s' = r(\vec{s}, \vec{t})$.

The above definition essentially implements the *soundness* and *witnessing* properties of [5, §4.3] within a general algebraic framework.

Of particular interest are term models. The existence and uniqueness of these is given by the result below.

Proposition 3.2 *For any $V \subseteq \mathrm{Vars}$, and $B \subseteq V \times \mathrm{Act} \times \mathrm{Terms}_V$, there is a unique model T satisfying: $|T| = \mathrm{Terms}_V$; for each operator f of arity k, it holds that $f_T(p_1, \ldots, p_k) = f(p_1, \ldots, p_k)$; and $x \xrightarrow{a}_T p$ if and only if $(x, a, p) \in B$.*

PROOF. One shows, by an easy structural induction on a term p, that the transitions out of p are uniquely determined by the initial data. □

The $V = \emptyset$ case of the proposition is just Lemma 4.3.9 of [5]. We refer to this model as the *closed term model*, and we write $T_{\mathcal{R}}$ for it. In cases in which one thinks of $\mathcal{R}$ as specifying a complete self-contained language, it is natural to think of the closed term model as the "intended" operational model of $\mathcal{R}$. Our general notion of model also includes all quotients of the closed term models of *disjoint extensions* of $\mathcal{R}$, in the sense of [1, Def. 2.11], by congruence relations contained in bisimilarity. Of course, there are many non-term models too.

In Theorem 5.1.1 of [5], it is observed that the closed term model is image finite. This property does not, of course, hold for arbitrary models. So we shall avoid making assumptions that depend upon image finiteness.

One of the basic results about GSOS systems is that bisimulation is a conguence on closed terms, [5, Theorem 5.1.2]. We shall need a stratified generalization of this result to arbitrary GSOS models, Proposition 3.4 below. Accordingly, we recall the relation $\sim$ of *(strong) bisimilarity* between states of transition systems, and its ordinal-indexed approximations $\sim_\alpha$, see e.g. [18, §10.4]. For transition systems S and T and ordinals α, the relation $\sim_\alpha^{ST}$ between $|S|$ and $|T|$ is defined by:

$$\begin{aligned} s \sim_{\alpha+1}^{ST} t \quad \text{iff} \quad & s \xrightarrow{a}_S s' \text{ implies } \exists t' \text{ such that } t \xrightarrow{a}_T t' \text{ and } s' \sim_\alpha^{ST} t', \text{ and} \\ & t \xrightarrow{a}_T t' \text{ implies } \exists s' \text{ such that } s \xrightarrow{a}_S s' \text{ and } s' \sim_\alpha^{ST} t', \end{aligned}$$

$$s \sim_\lambda^{ST} t \quad \text{iff} \quad \text{for all } \alpha < \lambda,\ s \sim_\alpha^{ST} t, \qquad \text{when } \lambda \text{ is a limit ordinal.}$$

Note that, vacuously, $s \sim_0^{ST} t$ always holds. Also $s \sim_\alpha^{ST} t$ implies $s \sim_{\alpha'}^{ST} t$, for all ordinals $\alpha' \leq \alpha$. Bisimilarity is defined by:

$$s \sim^{ST} t \quad \text{iff} \quad \text{for all ordinals } \alpha,\ s \sim_\alpha^{ST} t.$$

Note that the bisimilarity relation $\sim^{ST}$ always coincides with $\sim_\alpha^{ST}$ for some sufficiently large α; for example, take α to be the smallest cardinal strictly greater than the cardinality of $\mathcal{P}(|S| \times |T|)$.[3] This allows results about $\sim^{ST}$ to be inferred easily from properties of the $\sim_\alpha^{ST}$ relations. For example, one obtains the fixed-point property of $\sim^{ST}$.

$$\begin{aligned} s \sim^{ST} t \quad \text{iff} \quad & s \xrightarrow{a}_S s' \text{ implies } \exists t' \text{ such that } t \xrightarrow{a}_T t' \text{ and } s' \sim^{ST} t', \text{ and} \\ & t \xrightarrow{a}_T t' \text{ implies } \exists s' \text{ such that } s \xrightarrow{a}_S s' \text{ and } s' \sim^{ST} t'. \end{aligned} \tag{3}$$

Similarly, one immediately derives the analogues for $\sim^{ST}$ of Propositions 3.3 and 3.4 below.

Proposition 3.3

(1) For all $s \in S$, it holds that $s \sim_\alpha^{SS} s$.
(2) If $s \sim_\alpha^{ST} t$ then $t \sim_\alpha^{TS} s$.
(3) If $s \sim_\alpha^{SS'} s'$ and $s' \sim_\alpha^{S'S''} s''$ then $s \sim_\alpha^{SS''} s''$.

The proof is both standard and routine, so omitted.

Proposition 3.4 *If S and T are models then $\sim_\alpha^{ST}$ is a generalized congruence.*

In the proof, and henceforth, we shall drop superscripts on the $\sim_\alpha^{ST}$ (and $\sim^{ST}$) relations, as they can always be inferred from the context.

PROOF. By transfinite induction on α. The case for a limit ordinal is trivial, as generalized congruences are closed under intersection. To show the result for successor ordinals, assume that $\sim_\alpha$ is a generalized congruence and suppose $s_i \sim_{\alpha+1} t_i$, for all i with $1 \leq i \leq k$. We must show that $f(s_1, \ldots, s_k) \sim_{\alpha+1} f(t_1, \ldots, t_k)$.

Suppose $f(s_1, \ldots, s_k) \xrightarrow{c}_S s'$. We must find t' such that $f(t_1, \ldots, t_k) \xrightarrow{c}_T t'$ and $s' \sim_\alpha t'$. By the definition of model, there exists a rule in $\mathcal{R}$ of the form (2) above and there exist $\{s''_{ij}\}_{a \leq i \leq k}^{1 \leq j \leq n_i}$ such that:

(1) $s_i \xrightarrow{a_{ij}}_S s''_{ij}$, for all i, j with $1 \leq i \leq k$ and $1 \leq j \leq m_i$;

(2) $s_i \overset{b_{ij}}{\not\rightarrow}_S$, for all i, j' where $1 \leq i \leq k$ and $1 \leq j' \leq n_i$; and

[3] If S and T are image finite then α can be taken to be ω, see [16, Theorem 2.1].

(3) $s' = r(\vec{s}, \vec{s''})$.

For each i, j as above, we have $s_i \sim_{\alpha+1} t_i$, so, by 1, there exists t_{ij} such that $t_i \xrightarrow{a_{ij}}_S t''_{ij}$ and $s''_{ij} \sim_\alpha t''_{ij}$. Also, for each i, j', we have $t_i \overset{b_{ij}}{\not\rightarrow}_T$, again because $s_i \sim_{\alpha+1} t_i$. Thus, by the definition of model, $t \xrightarrow{c}_T r(\vec{t}, \vec{t''})$. Moreover, by the induction hypothesis, $r(\vec{s}, \vec{s''}) \sim_\alpha r(\vec{t}, \vec{t''})$, i.e. $s \sim_\alpha r(\vec{t}, \vec{t''})$. Thus $r(\vec{t}, \vec{t''})$ is the state t' we are looking for.

A similar argument establishes that $f_T(t_1, \ldots, t_k) \xrightarrow{c}_T t'$ implies that there exists s' with $f(s_1, \ldots, s_k) \xrightarrow{c}_S s'$ and $s' \sim_\alpha t'$. $\square$

Theorem 5.1.2 of [5] follows, as it is just the special case of the Proposition 3.4 in which $\sim_\alpha$ is the relation $\sim$ between the closed term model $\mathcal{T}_\mathcal{R}$ and itself. Also, it follows that, for any closed process p and models S, T, we have $p_S \sim p_T$, where p_S (resp. p_T) is the interpretation of p in S (resp. T).

Next we review Hennessy-Milner logic [16]. We use $A, B, C, \ldots$ to range over formulas, which are given by the grammar:

$$A \ ::= \ \top \ \mid \ \neg A \ \mid \ A \wedge B \ \mid \ \langle a \rangle A.$$

The other connectives and the $[a]$ modality can be defined by:

$$\bot \ = \ \neg\top \qquad A \vee B \ = \ \neg(\neg A \ \wedge \ \neg B) \qquad [a]A \ = \ \neg\langle a \rangle \neg A.$$

Given a labelled transition system, $T = (|T|, \{\xrightarrow{a}_T\})$, the satisfaction relation, $\Vdash_T$, relating states of T and formulas, is defined as usual: [4]

$$\begin{array}{rll} t \Vdash_T \top & \text{always holds} & \\ t \Vdash_T \neg A & \text{iff} & t \nVdash_T A \\ t \Vdash_T A \wedge B & \text{iff} & t \Vdash_T A \text{ and } t \Vdash_T B \\ t \Vdash_T \langle a \rangle A & \text{iff} & \text{if there exists } t' \text{ such that } t \xrightarrow{a}_T t' \text{ and } t' \Vdash_T A. \end{array}$$

The *modal depth*, $\mathrm{md}(A)$, of a formula A is defined by:

$$\begin{array}{rclcrcl} \mathrm{md}(\top) & = & 0 & \qquad & \mathrm{md}(A \wedge B) & = & \max(\mathrm{md}(A), \mathrm{md}(B)) \\ \mathrm{md}(\neg A) & = & \mathrm{md}(A) & & \mathrm{md}(\langle a \rangle A) & = & 1 + \mathrm{md}(A). \end{array}$$

Proposition 3.5 *If S, T are transition systems then, for any $s \in S$ and $t \in T$, the following are equivalent:*

(1) $s \sim_m t$.

[4] We use the symbol $\Vdash$ rather than $\models$, because, for us, T is the model, rather than t. Moreover, the symbol $\models$ is already "overloaded" by other uses in Section 4.

(2) For all A with $\mathrm{md}(A) \leq m$*, it holds that* $s \Vdash_S A$ *if and only if* $t \Vdash_T A$.

Theorem 2.2 of [16] states a similar result, but the setting is slightly different. In [16], T is required to be image finite, but Act is allowed to be infinite. The proof of the (1) $\Longrightarrow$ (2) direction in [16] does not use the assumption of image finiteness, and thus establishes this implication of Proposition 3.5. However, the proof in [16] of the (2) $\Longrightarrow$ (1) implication does make essential use of image finiteness. Indeed, when Act is infinite, such an assumption is necessary. Nevertheless, for finite Act, the (2) $\Longrightarrow$ (1) implication holds without any restrictions on T, s and t. We include a proof of this at the end of the section.

The final goal of this section is to establish various basic results about finite processes, which will be required in Section 6. Define:

$$F_0 \;=\; \{\emptyset\} \qquad\qquad F_{i+1} \;=\; \mathcal{P}(\mathrm{Act} \times F_i) \qquad\qquad F \;=\; \bigcup_{i \geq 0} F_i.$$

Note that $i \leq j$ implies $F_i \subseteq F_j$. Also, because Act is finite, each $u \in F$ is a finite set. Indeed, F is the smallest set such that $F = \mathcal{P}_{\mathrm{fin}}(\mathrm{Act} \times F)$. We consider F as a transition system, with the transition relation:

$$u \xrightarrow{a}_F u' \quad \text{iff} \quad (a, u') \in u.$$

We call the states in F the *finite processes*. If $u \in F_i$ then u is a finite process of *depth* at most i.

Proposition 3.6 *For every transition system T and $t \in T$:*

(1) There exists a unique $u^t_m \in F_m$ such that $t \sim_m u^t_m$.
(2) For every $u \in F_m$, if $t \sim_{m+1} u$ then $t \sim u$.

PROOF. We first prove statement 1 by induction on m. When $m = 0$, we have $\emptyset$ is the unique element of F_0 and trivially $t \sim_0 \emptyset$. When $m > 1$, for each $a \in \mathrm{Act}$ and $t' \in \mathrm{succ}_a(t)$ we have, by the induction hypothesis, a unique $u^{t'}_{m-1} \in F_{m-1}$ such that $t' \sim_{m-1} u^t_{m-1}$. Define

$$u^t_m \;=\; \{(a, u^{t'}_{m-1}) \mid a \in \mathrm{Act} \text{ and } t' \in \mathrm{succ}_a(t)\}.$$

It is easily verified that $t \sim_m u^t_m$. For uniqueness, consider any $u \in F_m$ such that $t \sim_m u$. Whenever $t \xrightarrow{a}_T t'$, there exists u' such that $u \xrightarrow{a}_F u'$ and $t' \sim_{m-1} u'$. But then $u' = u^{t'}_{m-1}$, by the uniqueness of $u^{t'}_{m-1}$. Hence $u \xrightarrow{a}_F u^{t'}_{m-1}$, i.e. $(a, u^{t'}_{m-1}) \in u$. This shows that $u^t_m \subseteq u$. For the converse inclusion, suppose that $(a, u') \in u$, i.e. $u \xrightarrow{a}_F u'$. Then there exists t' such that $t \xrightarrow{a}_T t'$ and $t' \sim_{m-1} u'$. Again, by the uniqueness of $u^{t'}_{m-1}$, we have $u' = u^{t'}_{m-1}$. Thus indeed $(a, u') \in u^t_m$, by the definition of u^t_m. So $u = u^t_m$, as required.

Statement 2 is also proved by induction on m. Observe that, for $\alpha > 0$, we have $\emptyset \sim t$ if and only if $t \overset{a}{\nrightarrow}_T$ for all actions $a \in \mathrm{Act}$ if and only if $\emptyset \sim_1 t$. The $m = 0$ case is now immediate as $\emptyset$ is the unique element of F_0. For $m > 0$, take any $u \in F_m$, and suppose that $t \sim_{m+1} u$. We show that $t \sim u$. Suppose first that $t \xrightarrow{a}_T t'$. Then there exists u' such that $u \xrightarrow{a}_F u'$ and $t' \sim_m u'$. But $u' \in F_{m-1}$. So, by the induction hypothesis, $t' \sim u'$. Also, if $u \xrightarrow{a}_F u'$ then there exists $t \xrightarrow{a}_T t'$ such that $t' \sim_m u'$. Then $u' \in F_{m-1}$. So, again by the induction hypothesis, $t' \sim u'$. Thus, by (3), indeed $t \sim u$. $\square$

An important property of finite processes is that each has a characteristic formula up to $\sim_m$ and $\sim$, in the sense of the proposition below. Statement (2) of the proposition is based on [15, Theorem 1].

Proposition 3.7

(1) For all $u \in F_m$, there exists a formula $\chi_m(u)$ with $\mathrm{md}(\chi_m(u)) \leq m$ such that, for all transition systems T and $t \in T$, it holds that $t \Vdash_T \chi_m(u)$ if and only $t \sim_m u$.

(2) For all $u \in F$, there exists a formula $\chi(u)$ such that, for all transition systems T and $t \in T$, it holds that $t \Vdash_T \chi(u)$ if and only $t \sim u$.

PROOF. Statement 1 is proved by induction on m. When $m = 0$, the only $u \in F_0$ is $u = \emptyset$. Define $\chi_0(\emptyset) = \top$. Then $t \Vdash_T \chi_m(\emptyset)$ always holds, as does $t \sim_0 \emptyset$. Suppose $m > 0$, and consider any $u \in F_m$. Define

$$\chi_m(u) \;=\; (\bigwedge\nolimits_{(a,u')\in u} \langle a\rangle\, \chi_{m-1}(u')) \;\wedge\; (\bigwedge\nolimits_{a\in \mathrm{Act}} [a]\, (\bigvee\nolimits_{u'\in\{u' \mid (a,u')\in u\}} \chi_{m-1}(u'))).$$

(N.b. an empty conjunction is $\top$ and an empty disjunction is $\bot$.) We have $\mathrm{md}(\chi_m(u)) \leq m$ because each $\mathrm{md}(\chi_{m-1}(u')) \leq m-1$, by induction hypothesis.

To prove that $t \Vdash_T \chi_m(u)$ implies $t \sim_m u$, assume that $t \Vdash_T \chi_m(u)$. Suppose $t \xrightarrow{a}_T t'$. Then, because $t \Vdash_T \chi_m(u)$, it holds that $t' \Vdash_T \bigvee_{u'\in\{u' \mid (a,u')\in u\}} \chi_{m-1}(u')$. So, for some u' with $(a, u') \in u$, we have $t' \Vdash_T \chi_{m-1}(u')$. Thus, by the induction hypothesis, $t' \sim_{m-1} u'$. Also $u \xrightarrow{a}_F u'$ because $(a, u') \in u$. Thus $t \xrightarrow{a}_T t'$ implies there indeed exists u' with $u \xrightarrow{a}_F u'$ and $t' \sim_{m-1} u'$. For the other implication in the definition of $\sim_m$, suppose $u \xrightarrow{a}_F u'$, i.e. $(a, u') \in u$. Then $t \Vdash_T \langle a\rangle\, \chi_{m-1}(u')$. Thus there exists t' such that $t \xrightarrow{a}_T t'$ and $t' \Vdash_T \chi_{m-1}(u')$. By the induction hypothesis $t' \sim_{m-1} u'$ as required. Thus indeed $t \Vdash_T \chi_m(u)$ implies $t \sim_m u$.

For the converse implication, assume that $t \sim_m u$. We show that $t \Vdash_T \chi_m(u)$. To establish that $t \Vdash_T \bigwedge_{(a,u')\in u} \langle a\rangle\, \chi_{m-1}(u')$, consider any $(a, u') \in u$. Then $u \xrightarrow{a}_F u'$. So, as $t \sim_m u$, we have $t \xrightarrow{a}_T t'$ for some t' such that $t' \sim_{m-1} u'$. By the induction hypothesis, $t' \Vdash_T \chi_{m-1}(u)$. So $t \Vdash_T \langle a\rangle\, \chi_{m-1}(u')$, as required. To establish that $t \Vdash_T \bigwedge_{a\in \mathrm{Act}} [a]\, (\bigvee_{u'\in\{u' \mid (a,u')\in u\}} \chi_{m-1}(u'))$, consider any $a \in \mathrm{Act}$

and suppose that $t \xrightarrow{a}_T t'$. Because $t \sim_m u$, we have $u \xrightarrow{a}_F u'$ for some u' with $t' \sim_{m-1} u'$. By the induction hypothesis, $t' \Vdash_T \chi_{m-1}(u')$. Also, $(a, u') \in u$, so $t' \Vdash_T \bigvee_{u' \in \{u' \mid (a,u') \in u\}} \chi_{m-1}(u')$, as required. This proves statement 1.

Statement 2 now follows easily. Consider any $u \in F$. Then there is a least m such that $u \in F_m$. Define $\chi(u) = \chi_{m+1}(u)$. We then have $t \Vdash_t \chi(u)$ iff $t \Vdash_t \chi_{m+1}(u)$ iff (by 1 above) $t \sim_{m+1} u$ iff (by Proposition 3.6(2)) $t \sim u$. □

Having obtained the above results, we can now establish the (2) $\implies$ (1) implication of Proposition 3.5, as promised above.

PROOF of Proposition 3.5 (2) $\implies$ (1). Suppose that $s \Vdash_S A$ if and only if $t \Vdash_T A$, for all A with $\mathrm{md}(A) \leq m$. By Proposition 3.6(1), there exists a unique $u^s_m \in F_m$ such that $s \sim_m u^s_m$. By Proposition 3.7(1), $s \Vdash_S \chi_m(u^s_m)$. As $\mathrm{md}(\chi_m(u^s_m)) \leq m$, we have, by assumption, that $t \Vdash_T \chi_m(u^s_m)$. Thus, by Proposition 3.7(1), $t \sim_m u^s_m$. We have that $s \sim_m u^s_m$ and $t \sim_m u^s_m$. It follows, by Proposition 3.3, that $s \sim_m t$, as required. □

4 The sequent calculus

In this section we present our applied sequent calculus. As motivated in Section 2, it has different assertion forms: *logical assertions* $p\colon A$, and *action assertions* $p \xrightarrow{a} q$. In addition, as the operational semantics allows negative premises, we include *inaction assertions* of the form $p \overset{a}{\nrightarrow}$. We use $J, K, \ldots$ to range over assertions and $\Gamma, \Delta, \ldots$ to range over (possibly infinite) sets of assertions.

Assertions have interpretations in arbitrary premodels. A relation $T \models_\gamma J$ between premodels T, environments γ and assertions J is defined by:

$$\begin{aligned}
T \models_\gamma p \xrightarrow{a} q \quad &\text{iff} \quad \gamma(p) \xrightarrow{a}_T \gamma(q),\\
T \models_\gamma p \overset{a}{\nrightarrow} \quad &\text{iff} \quad \gamma(p) \overset{a}{\nrightarrow}_T,\\
T \models_\gamma p\colon A \quad &\text{iff} \quad \gamma(p) \Vdash_T A.
\end{aligned}$$

Observe that $T \models_\gamma p \overset{a}{\nrightarrow}$ if and only if $T \models_\gamma [a]\bot$. Thus the inclusion of inaction assertions is, expressivity wise, unnecessary. Nevertheless, we find it convenient to include them, as it allows a clean separation between (in)action assertions, used in formalizing the operational semantics, and logical assertions.

We write $\Gamma \models_T \Delta$ to mean that, for all environments γ, if, for all $J \in \Gamma$, it holds that $T \models_\gamma J$ then there exists $K \in \Delta$ such that $T \models_\gamma K$. We write $\Gamma \models \Delta$ to mean that $\Gamma \models_T \Delta$ for all models T.

The sequent calculus uses *sequents* of the form $\Gamma \implies \Delta$, where Γ and Δ are finite, which are to be read as expressing that $\Gamma \models \Delta$. As we saw in Section 2, there are problems in obtaining a cut-free system for arbitrary sequents. We avoid these problems by defining a proof system operating on a restricted class of sequents. This restricted class is obtained by imposing conditions on the left-hand set of assertions.

Definition 4.1 (Assumable set) A (possibly infinite) set of assertions, Γ, is said to be *assumable* if it satisfies the following three conditions.

(A1) If $p \xrightarrow{a} q \in \Gamma$ then q is a process variable.
(A2) If $p \xrightarrow{a} x \in \Gamma$ and $q \xrightarrow{b} x \in \Gamma$ then $p = q$ (syntactic identity) and $a = b$.
(A3) The relation, $\lhd_\Gamma$, on process variables, defined by $x \lhd_\Gamma y$ if there exists $p \xrightarrow{a} y \in \Gamma$ such that p contains x, is well-founded.

Note that any subset of an assumable set is itself assumable.

Definition 4.2 (Admissible sequent) We call a sequent $\Gamma \implies \Delta$ *admissible* if Γ is assumable.

Our sequent calculus will work with admissible sequents.

Conditions (A1)–(A3) above are the simplest we could find with which we could obtain completeness and cut-elimination theorems. The three counterexamples from Section 2 are ruled out by conditions (A1) and (A2). Condition (A3) prevents, for example, assertions $p \xrightarrow{a} x$ from occurring in Γ when $x \in \mathrm{Vars}(p)$. For p containing arbitrary GSOS operators (involving negative premises) it may be impossible to satisfy such assertions for reasons to do with the nonexistence of solutions to arbitrary unguarded recursion equations. As in Section 2, one can find examples of such assertions for which the sequent $p \xrightarrow{a} x \implies$ apparently requires cut to be derivable.

The conditions on assumability are also intuitively motivated in the following way. They amount to being able to construct Γ from the empty set by (transfinitely) adding assertions one at a time, subject to the restriction that, whenever an action assertion $p \xrightarrow{a} x$ is added, x neither occurs in p nor in any of the assertions already included. Thus, at the time of adding $p \xrightarrow{a} x$, the variable x is unconstrained and represents an arbitrary process to which p can evolve. There is an analogy here with the declaration of variables in dependent type theories. Indeed, if one reads $p \xrightarrow{a} q$ as an assertion that q has "type" $p \xrightarrow{a}$, then the conditions on assumability are just an infinitary generalization of the usual dependency requirements on contexts.

We now give the proof rules for the sequent calculus. In the rules we adopt standard notational conventions, using comma for set union, omitting the empty set and omitting delimiters from singleton sets. Each rule is to be read as applying only when the premises and conclusion are all admissible sequents.

We present two proof systems: a *basic* proof system, and a *full* proof system. The rules for the basic system are contained in Figs. 1–3. Additional rules for the full system are listed in Fig. 4

The rules for logical assertions are presented in Fig. 1. These rules essentially form a sequent calculus for a multi-modality version of the minimal modal logic K, albeit with the extra baggage of process terms and (in)action assertions.

The rules for inaction assertions are presented in Fig. 2. They are a straightforward implementation of the definition of $\overset{a}{\nrightarrow}$ in terms of $\xrightarrow{a}$.

The rules for action assertions are presented in Fig. 3. The rationale behind them is that the method of deriving an action assertion $p \xrightarrow{c} q$ depends on the structure of p. If p is a variable y, then the only applicable rule is the ($\xrightarrow{c}$Ax) rule. This is just an instance of the familiar identity axiom of sequent calculus. It is the only instance of the identity axiom included in the basic proof system.

When p is of the form $f(p_1, \dots, p_k)$, for some operator f, then the rules for deriving $p \xrightarrow{c} q$ are determined by the GSOS system $\mathcal{R}$. Suppose that $\mathcal{R}$ contains exactly l_{fc} rules with operator f and action c, so for each h with $1 \le h \le l_{fc}$ we have a distinct rule:

$$\frac{\{x_i \xrightarrow{a_{hij}} y_{ij}\}^{1\le i\le k}_{1\le j\le m_{hi}} \qquad \{x_i \overset{b_{hij}}{\nrightarrow}\}^{1\le i\le k}_{1\le j\le n_{hi}}}{f(x_1, \dots, x_k) \xrightarrow{c} r_h(\vec{x}, \vec{y})} \tag{4}$$

Then we include l_{fc} rules in the sequent calculus for introducing action assertions of the form $f(p_1, \dots, p_k) \xrightarrow{c} r$ on the right of sequents, namely the rules $(f\xrightarrow{c}\mathrm{R})_1$, ..., $(f\xrightarrow{c}\mathrm{R})_{l_{fc}}$; and we include one rule introducing such assertions on the left, namely the rule ($f\xrightarrow{c}$L). Note that in any application of ($f\xrightarrow{c}$L), when $l_{fc} > 0$, it must be the case that $f(p_1, \dots, p_k) \xrightarrow{c} x \notin \Gamma$, as otherwise the premises would not be admissible. We give examples of the rules generated by some specific process operators below.

Observe that the basic system contains none of the usual "structural rules" of sequent calculus. The exchange and contraction rules are redundant because sequents are built from finite sets. The full system is obtained from the basic system by extending it with the rules of Fig. 4, all of which may be reasonably described as "structural rules". These rules include the standard weakening rules (WkL) and (WkR), and the standard axiom rule (Ax). Note that, in these rules, admissibility considerations require that any action assertion $p \xrightarrow{a} q$, appearing as J on the left-hand side of a sequent, must be of the form $p \xrightarrow{a} x$. The substitution and cut rules, (Sub) and (Cut), are as expected. However, for action assertions, we also include a natural combination of the two, ($\xrightarrow{c}$Cut),

$$\dfrac{}{\Gamma \Longrightarrow p\colon \top,\, \Delta}\;(\top\mathrm{R})$$

$$\dfrac{\Gamma \Longrightarrow p\colon A,\, \Delta}{\Gamma,\, p\colon \neg A \Longrightarrow \Delta}\;(\neg\mathrm{L}) \qquad \dfrac{\Gamma,\, p\colon A \Longrightarrow \Delta}{\Gamma \Longrightarrow p\colon \neg A,\, \Delta}\;(\neg\mathrm{R})$$

$$\dfrac{\Gamma, p\colon A,\, p\colon B \Longrightarrow \Delta}{\Gamma,\, p\colon A \wedge B \Longrightarrow \Delta}\;(\wedge\mathrm{L}) \qquad \dfrac{\Gamma \Longrightarrow p\colon A,\, \Delta \quad \Gamma \Longrightarrow p\colon B,\, \Delta}{\Gamma \Longrightarrow p\colon A \wedge B,\, \Delta}\;(\wedge\mathrm{R})$$

$$\dfrac{\Gamma,\, p \xrightarrow{a} x,\, x\colon A \Longrightarrow \Delta}{\Gamma,\, p\colon \langle a\rangle A \Longrightarrow \Delta}\;(\langle a\rangle\mathrm{L})^{*} \qquad \dfrac{\Gamma \Longrightarrow p \xrightarrow{a} q,\, \Delta \quad \Gamma \Longrightarrow q\colon A,\, \Delta}{\Gamma \Longrightarrow p\colon \langle a\rangle A,\, \Delta}\;(\langle a\rangle\mathrm{R})$$

*Restriction on ($\langle a\rangle$L): the variable x must not occur in the rule conclusion.

Fig. 1. Rules for logical assertions

$$\dfrac{\Gamma \Longrightarrow p \xrightarrow{a} q,\, \Delta}{\Gamma,\, p \overset{a}{\nrightarrow} \Longrightarrow \Delta}\;(\overset{a}{\nrightarrow}\mathrm{L}) \qquad \dfrac{\Gamma,\, p \xrightarrow{a} x \Longrightarrow \Delta}{\Gamma \Longrightarrow p \overset{a}{\nrightarrow},\, \Delta}\;(\overset{a}{\nrightarrow}\mathrm{R})^{*}$$

*Restriction on ($\overset{a}{\nrightarrow}$R): the variable x must not occur in the rule conclusion.

Fig. 2. Rules for inaction assertions

$$\dfrac{}{\Gamma,\, y \xrightarrow{c} x \Longrightarrow y \xrightarrow{c} x,\, \Delta}\;(\xrightarrow{c}\mathrm{Ax})$$

$$\dfrac{\{\Gamma \Longrightarrow p_i \xrightarrow{a_{hij}} q_{hij},\, \Delta\}^{1\le i\le k}_{1\le j\le m_{hi}} \qquad \{\Gamma \Longrightarrow p_i \overset{b_{hij}}{\nrightarrow},\, \Delta\}^{1\le i\le k}_{1\le j\le n_{hi}}}{\Gamma \Longrightarrow f(p_1,\ldots,p_k) \xrightarrow{c} r_h(\vec{p},\vec{q}),\, \Delta}\;(f\xrightarrow{c}\mathrm{R})_h$$

$$\dfrac{\left\{\, \Gamma[r_h(\vec{p},\vec{y})/x],\, \{p_i \xrightarrow{a_{hij}} y_{ij}\}^{1\le i\le k}_{1\le j\le m_{hi}},\, \{p_i \overset{b_{hij}}{\nrightarrow}\}^{1\le i\le k}_{1\le j\le n_{hi}} \Longrightarrow \Delta[r_h(\vec{p},\vec{y})/x] \,\right\}_{1\le h\le l_{fc}}}{\Gamma,\, f(p_1,\ldots,p_k) \xrightarrow{c} x \Longrightarrow \Delta}\;(f\xrightarrow{c}\mathrm{L})^{*}$$

*Restriction on ($f\xrightarrow{c}$L): all of the variables y_{ij} are distinct and do not occur in the rule conclusion.

Fig. 3. Rules for action assertions

$$\frac{}{\Gamma,\, J \implies J,\, \Delta}\,(\mathrm{Ax}) \qquad \frac{\Gamma \implies \Delta}{\Gamma,\, J \implies \Delta}\,(\mathrm{WkL}) \qquad \frac{\Gamma \implies \Delta}{\Gamma \implies J,\, \Delta}\,(\mathrm{WkR})$$

$$\frac{\Gamma \implies \Delta}{\Gamma[p/x] \implies \Delta[p/x]}\,(\mathrm{Sub}) \qquad \frac{\Gamma,\, J \implies \Delta \quad \Gamma \implies J,\, \Delta}{\Gamma \implies \Delta}\,(\mathrm{Cut})$$

$$\frac{\Gamma,\, p \xrightarrow{c} x \implies \Delta \quad \Gamma[q/x] \implies p \xrightarrow{c} q,\, \Delta[q/x]}{\Gamma[q/x] \implies \Delta[q/x]}\,(\xrightarrow{c}\mathrm{Cut})$$

Fig. 4. Additional rules for the full proof system

which allows one to cut out an arbitrary action assertion from the right-hand side of a sequent in a way that is consistent admissibility requirements.

The ethos behind the separation of the basic and full proof systems is as follows. Each rule in the basic system is associated to a single logical connective or operational primitive. Moreover, the premises of the rule are obtained by decomposing a formula or process term in a principled way depending on the associated primitive. Furthermore, each rule embodies, as directly as possible, a basic logical property of its associated notion. Thus the rules of the basic system directly implement structural and natural reasoning in the sense of Section 1. On the other hand, the rules of the extended system implement general properties of logical consequence, useful for modularizing proofs, and essential for formalizing compositional verification methods.

Before stating our main results we give some illustrative examples of the induced rules for particular process operators. For the prefix, zero and sum operators, the right-hand rules are the same as those given earlier in Section 2. The left-hand rules differ in that they are specifically tailored to admissible sequents. The new versions are (modulo trivial variable renamings):

$$\frac{\Gamma[p/x] \implies \Delta[p/x]}{\Gamma,\, a.p \xrightarrow{a} x \implies \Delta} \qquad \frac{}{\Gamma,\, a.p \xrightarrow{b} x \implies \Delta}\, a \neq b$$

$$\frac{}{\Gamma,\, 0 \xrightarrow{a} x \implies \Delta} \qquad \frac{\Gamma,\, p \xrightarrow{a} x \implies \Delta \quad \Gamma,\, q \xrightarrow{a} x \implies \Delta}{\Gamma,\, p + q \xrightarrow{a} x \implies \Delta}$$

All the rules are special cases of their earlier counterparts. We remark that the renaming and restriction operators of CCS can also be dealt with easily.

Parallel operators are more interesting. First, we consider the interleaving (non-communicating) parallel, $p \| q$, whose operational rules are:

$$\frac{x \xrightarrow{a} x'}{x \| y \xrightarrow{a} x' \| y} \qquad \frac{y \xrightarrow{a} y'}{x \| y \xrightarrow{a} x \| y'}$$

The derived sequent rules for $\|$ are therefore:

$$\frac{\Gamma \implies p \xrightarrow{a} p',\, \Delta}{\Gamma \implies p \| q \xrightarrow{a} p' \| q,\, \Delta} \qquad \frac{\Gamma \implies q \xrightarrow{a} q',\, \Delta}{\Gamma \implies p \| q \xrightarrow{a} p \| q',\, \Delta}$$

$$\frac{\Gamma[x \| q \,/\, z],\, p \xrightarrow{a} x \implies \Delta[x \| q \,/\, z] \quad \Gamma[p \| y \,/\, z],\, q \xrightarrow{a} y \implies \Delta[p \| y \,/\, z]}{\Gamma,\, p \| q \xrightarrow{a} z \implies \Delta}$$

where in the last rule, the variables x and y must not appear in the concluding sequent. This rule nicely illustrates the difference between structural and compositional proof methods. The rule is manifestly structural, as the premises are obtained via syntactic decompositions of the conclusion. In particular, the assertion $p \| q \xrightarrow{a} z$ in the conclusion is broken down into assertions $p \xrightarrow{a} x$ and $q \xrightarrow{a} y$ in the premises. However, the rule is not compositional, because each premise contains both the process terms p and q.

The communicating parallel $p|q$ of CCS is slightly more complicated, due to the fact that a silent τ action can be triggered by a synchronization on any non-τ action. Thus the left-hand rule for $p|q \xrightarrow{\tau} z$ assertions is

$$\frac{\Gamma_1,\, p \xrightarrow{\tau} x \implies \Delta_1 \quad \Gamma_2,\, q \xrightarrow{\tau} y \implies \Delta_2 \quad \{\Gamma_3,\, p \xrightarrow{a} x,\, q \xrightarrow{\overline{a}} y \implies \Delta_3\}_{a \in \mathrm{Act} \setminus \{\tau\}}}{\Gamma,\, p|q \xrightarrow{\tau} z \implies \Delta}$$

where: Γ_1, Δ_1 are obtained from Γ, Δ using the substitution $[x|q \,/\, z]$; the sets Γ_2, Δ_2 are obtained using $[p|y \,/\, z]$; the sets Γ_3, Δ_3 are obtained using $[x|y \,/\, z]$; and x, y do not appear in the rule conclusion. Similar complexities in the handling of communicating parallel were encountered by Stirling [25]. In our setting, a natural way of reducing the complexity would be to include an algebra of action variables and terms (e.g. $\overline{(\cdot)}$ should be a unary operator on actions), together with a new assertion form stating equality between action terms. However, to properly incorporate these features, similar such modifications must also be made at the level of the GSOS rule specifications. In this paper, we content ourselves with dealing with ordinary GSOS specifications, based on a finite set of actions.

None of the example operators above exploits the GSOS feature of allowing negative premises (inaction assertions) in operational rules. For an example involving inaction assertions, the reader is referred to the original conference version of this paper [23].

Subderivation (A):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Longrightarrow x''\|y : \top} \qquad \dfrac{\dfrac{}{x' \xrightarrow{b} x'' \Longrightarrow x' \xrightarrow{b} x''}}{x' \xrightarrow{b} x'' \Longrightarrow x'\|y \xrightarrow{b} x''\|y}}{x' \xrightarrow{b} x'',\, x'' : \top \Longrightarrow x'\|y' : \langle b\rangle\top}}{x' : \langle b\rangle\top \Longrightarrow x'\|y : \langle b\rangle\top} \qquad \dfrac{}{x \xrightarrow{a} x' \Longrightarrow x \xrightarrow{a} x'}}{x : [a]\langle b\rangle\top,\, x \xrightarrow{a} x' \Longrightarrow x'\|y : \langle b\rangle\top}$$

Subderivation (B):

$$\dfrac{\dfrac{\dfrac{}{\Longrightarrow x'\|y' : \top} \qquad \dfrac{\dfrac{}{x \xrightarrow{b} x' \Longrightarrow x \xrightarrow{b} x'}}{x \xrightarrow{b} x' \Longrightarrow x\|y' \xrightarrow{b} x'\|y'}}{x \xrightarrow{b} x',\, x' : \top \Longrightarrow x\|y' : \langle b\rangle\top}}{x : \langle b\rangle\top, y \xrightarrow{a} y' \Longrightarrow x\|y' : \langle b\rangle\top}$$

Main derivation:

$$\dfrac{\dfrac{\begin{matrix}\vdots\ \text{(A)} \\ x : [a]\langle b\rangle\top,\, x \xrightarrow{a} x' \Longrightarrow x'\|y : \langle b\rangle\top\end{matrix} \qquad \begin{matrix}\vdots\ \text{(B)} \\ x : \langle b\rangle\top, y \xrightarrow{a} y' \Longrightarrow x\|y' : \langle b\rangle\top\end{matrix}}{x : \langle b\rangle\top,\, x : [a]\langle b\rangle\top,\, x\|y \xrightarrow{a} z \Longrightarrow z : \langle b\rangle\top}}{x : \langle b\rangle\top,\, x : [a]\langle b\rangle\top \Longrightarrow x\|y : [a]\langle b\rangle\top}$$

Fig. 5. Example derivation of $x : \langle b\rangle\top,\, x : [a]\langle b\rangle\top \Longrightarrow x\|y : [a]\langle b\rangle\top$.

To show the basic proof system at work, we give in Fig. 5 an example derivation of the sequent $x : \langle b\rangle\top,\, x : [a]\langle b\rangle\top \Longrightarrow x\|y : [a]\langle b\rangle\top$. For readability, we write $[a]$ as a primitive modality, using the proof rules (1) from Section 2. When $[a]$ is defined in terms of $\langle a\rangle$, these proof rules are easily derived via a combination of the rules for $\langle a\rangle$ and negation. We also avoid including extraneous assertions in the sequents in Fig. 5. The full derivation involves evident weakenings of the written sequents.

We end this section with the main results. For assumable (possibly infinite) Γ and arbitrary Δ we write $\Gamma \vdash_{\mathrm{b}} \Delta$ to mean that there exist finite subsets $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ such that the sequent $\Gamma' \Longrightarrow \Delta'$ (which is admissible) is derivable in the basic proof system. Similarly, we write $\Gamma \vdash_{\mathrm{f}} \Delta$ to mean that for some finite subsets $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ the sequent $\Gamma' \Longrightarrow \Delta'$ is derivable in the full proof system. Trivially $\Gamma \vdash_{\mathrm{b}} \Delta$ implies $\Gamma \vdash_{\mathrm{f}} \Delta$.

Theorem 1 (Soundness of the full proof system) *If* $\Gamma \vdash_{\mathrm{f}} \Delta$ *then* $\Gamma \models \Delta$.

PROOF. We prove, by induction on derivations, that if $\Gamma \implies \Delta$ is derivable in the full proof system then $\Gamma \models \Delta$. We consider three illustrative cases involving action assertions.

Case 1. Suppose we have derived $\Gamma \implies f(p_1,\ldots,p_k) \xrightarrow{c} r_h(\vec{p},\vec{q}),\ \Delta$ as a result of an application of the $(f\xrightarrow{c}\mathrm{R})_h$ rule. Then, by the induction hypothesis, we have $\Gamma \models p_i \overset{a_{hij}}{\rightarrow} q_{hij},\ \Delta$, for all i,j with $1 \leq i \leq k$ and $1 \leq j \leq m_{hi}$; and $\Gamma \models p_i \overset{b_{hij}}{\nrightarrow},\ \Delta$, for all i,j with $1 \leq i \leq k$ and $1 \leq j \leq n_{hi}$. We must show that $\Gamma \models f(p_1,\ldots,p_k) \xrightarrow{c} r_h(\vec{p},\vec{q}),\ \Delta$. Consider then any model T and environment γ such that, for all $J \in \Gamma$, it holds that $T \models_\gamma J$ and, for all $K \in \Delta$, it holds that $T \not\models_\gamma K$. We must show that $T \models_\gamma f(p_1,\ldots,p_k) \xrightarrow{c} r_h(\vec{p},\vec{q})$. By the induction hypothesis, $T \models_\gamma p_i \overset{a_{hij}}{\rightarrow} q_{hij}$ and $T \models_\gamma p_i \overset{b_{hij}}{\nrightarrow}$, for all relevant i,j; i.e. $\gamma(p_i) \overset{a_{hij}}{\rightarrow}_T \gamma(q_{ij})$ and $\gamma(p_i) \overset{b_{hij}}{\nrightarrow}_T$. Thus, by Definition 3.1, we have $\gamma(f(p_1,\ldots,p_k)) \xrightarrow{c}_T \gamma(r_h(\vec{p},\vec{q}))$, i.e. $T \models_\gamma f(p_1,\ldots,p_k) \xrightarrow{c} r_h(\vec{p},\vec{q})$, as required.

Case 2. Suppose we have derived $\Gamma,\ f(p_1,\ldots,p_k) \xrightarrow{c} x \implies \Delta$, as a result of an application of the $(f\xrightarrow{c}\mathrm{L})$ rule. Then, by the induction hypothesis, we have

$$\Gamma[r_h(\vec{p},\vec{y})/x],\ \{p_i \overset{a_{hij}}{\rightarrow} y_{ij}\}^{1\leq i\leq k}_{1\leq j\leq m_{hi}},\ \{p_i \overset{b_{hij}}{\nrightarrow}\}^{1\leq i\leq k}_{1\leq j\leq n_{hi}} \models \Delta[r_h(\vec{p},\vec{y})/x]\,,$$

for each h with $1 \leq h \leq l_{fc}$. We must show that $\Gamma,\ f(p_1,\ldots,p_k) \xrightarrow{c} x \models \Delta$. Consider any model T and environment γ such that $T \models_\gamma f(p_1,\ldots,p_k) \xrightarrow{c} x$ and, for all $J \in \Gamma$, it holds that $T \models_\gamma J$. We must show that $T \models_\gamma K$ for some $K \in \Delta$. As $T \models_\gamma f(p_1,\ldots,p_k) \xrightarrow{c} x$, we have $\gamma(f(p_1,\ldots,p_k)) \xrightarrow{c}_T \gamma(x)$. So, by Definition 3.1, there exists h with $1 \leq h \leq l_{fc}$, and there exist $\{t_{ij}\}^{1\leq j\leq n_i}_{1\leq i\leq k}$ such that: for all i,j with $1 \leq i \leq k$ and $1 \leq j \leq m_i$, it holds that $\gamma(p_i) \overset{a_{hij}}{\rightarrow}_T t_{ij}$; for all i,j with $1 \leq i \leq k$ and $1 \leq j \leq n_i$, it holds that $\gamma(p_i) \overset{b_{hij}}{\nrightarrow}_T$; and $\gamma(x) = r(\gamma(\vec{p}),\vec{t})$. Define $\gamma' = \gamma[\vec{y} \mapsto \vec{t}]$. Then $\gamma(x) = \gamma'(r_h(\vec{p},\vec{y}))$, because none of the $\vec{y}$ variables occurs in $\vec{p}$. Also, for every $J \in \Gamma \cup \Delta$, none of the $\vec{y}$ variables occurs in J, so $T \models_{\gamma'} J[r_h(\vec{p},\vec{y})/x]$ if and only if $T \models_\gamma J$. Therefore: for all $J \in \Gamma$, it holds that $T \models_{\gamma'} J[r_h(\vec{p},\vec{y})/x]$; for all relevant i,j, it holds that $T \models_{\gamma'} p_i \overset{a_{hij}}{\rightarrow} y_{ij}$; and, for all relevant i,j, it holds that $T \models_{\gamma'} p_i \overset{b_{hij}}{\nrightarrow}$. Thus, by the induction hypothesis, there exists $K \in \Delta$ such that $T \models_{\gamma'} K[r_h(\vec{p},\vec{y})/x]$. So indeed $T \models_\gamma K$.

Case 3. Suppose we have derived $\Gamma[q/x] \implies \Delta[q/x]$ as a result of an application of the $(\xrightarrow{c}\mathrm{Cut})$ rule. Then, by the induction hypothesis, $\Gamma,\ p \xrightarrow{c} x \models \Delta$ and $\Gamma[q/x] \models p \xrightarrow{c} q,\ \Delta[q/x]$. We must show that $\Gamma[q/x] \models \Delta[q/x]$. Accordingly, consider any model T and environment γ. Suppose, for contradiction that: for all $J \in \Gamma$, it holds that $T \models_\gamma J[q/x]$; and, for all $K \in \Delta$, it holds that $T \not\models_\gamma K[q/x]$. Then, because $\Gamma[q/x] \models p \xrightarrow{c} q,\ \Delta[q/x]$, we have $T \models_\gamma p \xrightarrow{c} q$. Define $\gamma' = \gamma[x \mapsto \gamma(q)]$. Then, for all $J \in \Gamma$, it holds that $T \models_{\gamma'} J$; and, for all $K \in \Delta$, it holds that $T \not\models_{\gamma'} K$. So, as $\Gamma,\ p \xrightarrow{c} x \models \Delta$, we have $T \not\models_{\gamma'} p \xrightarrow{c} x$.

But $x \notin \mathrm{Vars}(p)$, because $\Gamma, p \xrightarrow{a} x$ is assumable, so $\gamma'(p) = \gamma(p)$. Therefore $T \not\models_\gamma p \xrightarrow{c} q$. This gives the desired contradiction.

Remaining cases. These are similar, and are left to the reader. $\square$

Theorem 2 (Completeness of the basic proof system) *If* Γ *is assumable and* $\Gamma \models \Delta$ *then* $\Gamma \vdash_{\mathrm{b}} \Delta$.

The proof of this theorem is given in Section 5 below.

We state two immediate corollaries of Theorems 1 and 2.

Corollary 1 (Equivalence of the basic and full systems) $\Gamma \vdash_{\mathrm{b}} \Delta$ *if and only if* $\Gamma \vdash_{\mathrm{f}} \Delta$.

Thus all the proof rules in Fig. 4 are admissible in the basic system, including, in particular, the (Cut) and ($\xrightarrow{c}$Cut) rules. It would be interesting to obtain a syntactic proof of this fact. We have not carried out such a proof, but we have checked that the "local" proof reductions all go through. Interestingly, these local reductions confirm the naturalness of including the ($\xrightarrow{c}$Cut) rule as primitive in conjunction with (Cut).

Corollary 2 (Compactness) *If* $\Gamma \models \Delta$ *then there exist finite subsets* $\Gamma' \subseteq \Gamma$ *and* $\Delta' \subseteq \Delta$ *such that* $\Gamma' \models \Delta'$.

5 Proof of completeness

This entire section is dedicated to the proof of Theorem 2. As usual we prove the contrapositive. Suppose that $\Gamma_0 \not\vdash_{\mathrm{b}} \Delta_0$ where Γ_0 is assumable, and where, for convenience in the proof, we assume, without loss of generality, that there are infinitely many variables not contained in $\mathrm{Vars}(\Gamma_0 \cup \Delta_0)$. We shall construct a model T_c together with an environment γ_c showing that $\Gamma_0 \not\models \Delta_0$.

To help construct the model, we define a sequence of 4-tuples $(\Gamma_i, \Delta_i, \sigma_i, D_i)$, for $i \geq 0$. In doing so, we ensure that Γ_i is assumable and that $\Gamma_i \not\vdash_{\mathrm{b}} \Delta_i$. The sequence is constructed by eventually breaking down all compound assertions in Γ_i (resp. Δ_i) into components that witness their truth (resp. falsity) in the constructed model. This much will be familiar to anyone who has previously seen a direct proof of completeness for a cut-free sequent calculus or tableau system. In our case, there are added complications to do with maintaining the assumability of Γ_i. To deal with this, we simultaneously build substitution functions σ_i (see Section 3) recording the sequence of term substitutions made when decomposing action assertions in the process of generating Γ_i and Δ_i from Γ_0 and Δ_0. We also record the domains $D_i \subseteq \mathrm{Vars}$ on which these substitution functions are nontrivial. The main technical difficulty is to show that the sequence of substitution functions converges to a limiting substitution function, required to define the environment γ_c, see Lemma 5.7.

In defining the sequence, we write U_i for the set $\bigcup_{j\leq i} \mathrm{Vars}(\Gamma_j \cup \Delta_j)$, and V_i for $U_i \backslash D_i$. We also write ϵ for an empty vector (of process terms).

We already have Γ_0 and Δ_0. Define σ_0 to be the identify function, and $D_0 = \emptyset$.

To define the rest of the sequence we use a "scheduling" set. Let $\{\tau_0, \tau_1, \dots\}$ be an enumeration of all 3-tuples $(J, \vec{q}, m)$, where J is an assertion, $\vec{q}$ is a (possibly empty) vector of process terms and $m \geq 0$, so that each such 3-tuple appears infinitely often in the enumeration. The purpose of these 3-tuples is to record, in a convenient (though somewhat redundant) format, each possible way of decomposing a goal into new subgoals using basic proof rules. The first component represents the assertion decomposed by a rule. The second component is the sequence of term substitutions used in the decomposition. The third component, if 0, signifies a left rule and, if ≥ 1, signifies a right rule, with the higher numbers being used to enumerate the rules $(f\xrightarrow{c}\mathrm{R})_1, \dots,$ $(f\xrightarrow{c}\mathrm{R})_{l_{fc}}$. The precise way in which all this information is used will be clarified by the construction below.

Define $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$, $\sigma_{i+1} = \sigma_i$ and $D_{i+1} = D_i$, *unless* one of the following conditions holds:

- $\tau_i = (p\colon \neg A, \epsilon, 0)$ and $p\colon \neg A \in \Gamma_i$, in which case $\Delta_{i+1} = \Delta_i \cup \{p\colon A\}$;
- $\tau_i = (p\colon A \wedge B, \epsilon, 0)$ and $p\colon A \wedge B \in \Gamma_i$, in which case $\Gamma_{i+1} = \Gamma_i \cup \{p\colon A,\ p\colon B\}$;
- $\tau_i = (p\colon \langle a\rangle A, \epsilon, 0)$ and $p\colon \langle a\rangle A \in \Gamma_i$, in which case $\Gamma_{i+1} = \Gamma_i \cup \{p \xrightarrow{a} x,\ x\colon A\}$ where x is a chosen variable not contained in U_i;
- $\tau_i = (p\overset{a}{\not\rightarrow}, q, 0)$, and $p\overset{a}{\not\rightarrow} \in \Gamma_i$ and also $\mathrm{Vars}(q) \subseteq V_i$, in which case $\Delta_{i+1} = \Delta_i \cup \{p \xrightarrow{a} q\}$;
- $\tau_i = (f(p_1, \dots, p_k) \xrightarrow{c} x,\ \epsilon,\ 0)$ and $f(p_1, \dots, p_k) \xrightarrow{c} x \in \Gamma_i$, in which case:

$$\begin{aligned}\Gamma_{i+1} &= (\Gamma_i \backslash \{f(p_1, \dots, p_k) \xrightarrow{c} x\})[r_h(\vec{p}, \vec{y})/x] \cup \\ &\qquad \{p_{i'} \xrightarrow{a_{hi'j}} y_{i'j}\}^{1\leq i'\leq k}_{1\leq j\leq m_{hi'}} \cup \{p_{i'} \overset{b_{hi'j}}{\not\rightarrow}\}^{1\leq i'\leq k}_{1\leq j\leq n_{hi'}} \\ \Delta_{i+1} &= \Delta_i[r_h(\vec{p}, \vec{y})/x] \\ \sigma_{i+1} &= z \mapsto (\sigma_i(z))[r_h(\vec{p}, \vec{y})/x] \\ D_{i+1} &= D_i \cup \{x\}.\end{aligned}$$

  where $\vec{y}$ is a chosen vector of distinct variables not contained in U_i and h is chosen with $1 \leq h \leq l_{fc}$ so that $\Gamma_{i+1} \nvdash_{\mathrm{b}} \Delta_{i+1}$;
- $\tau_i = (p\colon \neg A, \epsilon, 1)$ and $p\colon \neg A \in \Delta_i$, in which case $\Gamma_{i+1} = \Gamma_i \cup \{p\colon A\}$;
- $\tau_i = (p\colon A \wedge B, \epsilon, 1)$ and $p\colon A \wedge B \in \Delta_i$, in which case $\Delta_{i+1} = \Delta_i \cup \{p\colon A\}$ if $\Gamma_i \nvdash_{\mathrm{b}} p\colon A,\ \Delta_i$, and $\Delta_{i+1} = \Delta_i \cup \{p\colon B\}$ otherwise;
- $\tau_i = (p\colon \langle a\rangle A, q, 1)$ and $p\colon \langle a\rangle A \in \Delta_i$ and also $\mathrm{Vars}(q) \subseteq V_i$, in which case if

$\Gamma_i \not\vdash_{\mathrm{b}} q\colon A,\ \Delta_i$ then $\Delta_{i+1} = \Delta_i \cup \{q\colon A\}$, otherwise $\Delta_{i+1} = \Delta_i \cup \{p \xrightarrow{a} q\}$.

- $\tau_i = (p \overset{a}{\not\rightarrow}, \epsilon, 1)$ and $p \overset{a}{\not\rightarrow} \in \Delta_i$, in which case $\Gamma_{i+1} = \Gamma_i \cup \{p \xrightarrow{a} x\}$ where x is a chosen variable not contained in U_i;
- $\tau_i = (f(p_1, \ldots, p_k) \xrightarrow{c} r_h(\vec{p}, \vec{q}),\ \vec{q},\ h)$, where $1 \leq h \leq l_{fc}$, and $\mathrm{Vars}(\vec{q}) \subseteq V_i$, and also $f(p_1, \ldots, p_k) \xrightarrow{c} r_h(\vec{p}, \vec{q}) \in \Delta_i$, in which case: if there exist i', j with $1 \leq i' \leq k$ and $1 \leq j \leq m_{hi'}$ such that $\Gamma_i \not\vdash_{\mathrm{b}} p_{i'} \xrightarrow{a_{hi'j}} q_{i'j},\ \Delta_i$ then $\Delta_{i+1} = \Delta_i \cup \{p_{i'} \xrightarrow{a_{hi'j}} q_{i'j}\}$ for a chosen such i', j; otherwise $\Delta_{i+1} = \Delta_i \cup \{p_{i'} \overset{b_{hi'j}}{\not\rightarrow}\}$ for a chosen i', j with $1 \leq i' \leq k$ and $1 \leq j \leq n_{hi'}$ such that $\Gamma_i \not\vdash_{\mathrm{b}} p_{i'} \overset{b_{hi'j}}{\not\rightarrow},\ \Delta_i$;

where, in the action assertion cases, it is assumed that the rules in $\mathcal{R}$ with operator f and action c have the form in (4) of Section 4, and that the vectors $\vec{y}$ and $\vec{q}$ have the appropriate length. (Similar assumptions are made below without further comment.)

Curiously, in order to establish that the definition above is good, we first need a simple proof-theoretic lemma.

Lemma 5.1 *The weakening rules* (WkL) *and* (WkR) *are admissible in the basic proof system.*

PROOF. A straightforward induction on the structure of derivations.

Lemma 5.2 *The sequence* $(\Gamma_i, \Delta_i, \sigma_i, D_i)$ *is well defined, each* Γ_i *is assumable and* $\Gamma_i \not\vdash_{\mathrm{b}} \Delta_i$*.*

PROOF. We prove by induction on i that: $(\Gamma_i, \Delta_i, \sigma_i, D_i)$ is well defined; Γ_i is assumable; $\Gamma_i \not\vdash_{\mathrm{b}} \Delta_i$, and there are infinitely many variables not contained in U_i. The base case $i = 0$ is trivial. For $i > 0$, the result is also trivial if none of the τ_{i-1} clauses applies. If one of the τ_{i-1} clauses does apply, then the proof splits into a case analysis, one for each clause. In each case the argument is similar. We illustrate the general pattern, by considering the two most interesting cases.

Case 1: $\tau_i = (f(p_1, \ldots, p_k) \xrightarrow{c} x,\ \epsilon,\ 0)$. We give the argument for this case in detail. For each h with $1 \leq h \leq l_{fc}$, define:

$$\begin{aligned}
\Gamma'_h &= (\Gamma_{i-1} \backslash \{f(p_1, \ldots, p_k) \xrightarrow{c} x\})[r_h(\vec{p}, \vec{y})/x] \cup \\
&\qquad\qquad \{p_{i'} \xrightarrow{a_{hij'}} y_{i'j}\}^{1 \leq i' \leq k}_{1 \leq j \leq m_{hi}} \cup \{p_{i'} \overset{b_{hi'j}}{\not\rightarrow}\}^{1 \leq i' \leq k}_{1 \leq j \leq n_{hi'}} \\
\Delta'_h &= \Delta_{i-1}[r_h(\vec{p}, \vec{y})/x]
\end{aligned}$$

selecting the $y_{i'j}$ from the infinitely many variables not contained in U_{i-1}. We show that each Γ'_h is assumable. Conditions (A1) and (A2) are immediate.

For the well-foundedness of the $\lhd_{\Gamma'_h}$ relation, suppose, for contradiction, that $\ldots \lhd_{\Gamma'_h} z_2 \lhd_{\Gamma'_h} z_1 \lhd_{\Gamma'_h} z_0$ is a descending sequence of variables. For any $l \geq 1$, it holds that $z_l \in \mathrm{Vars}(\Gamma_{i-1})\backslash\{x\}$ (note that z_0 may be one of the $y_{i'j}$ variables). For variables $z, z' \in \mathrm{Vars}(\Gamma_{i-1})\backslash\{x\}$, it holds that $z' \lhd_{\Gamma'_h} z$ iff: either $z' \lhd_{\Gamma_{i-1}} z$; or $z' \lhd_{\Gamma_{i-1}} x \lhd_{\Gamma_{i-1}} z$. Thus, it holds that $\ldots \lhd^+_{\Gamma_{i-1}} z_3 \lhd^+_{\Gamma_{i-1}} z_2 \lhd^+_{\Gamma_{i-1}} z_1$, contradicting the well-foundedness of $\lhd_{\Gamma_{i-1}}$. So Γ'_h is indeed assumable.

Next, suppose, for contradiction, that $\Gamma'_h \vdash_{\mathrm{b}} \Delta'_h$ for each h. Then there exist finite subsets $\Gamma''_h \subseteq \Gamma'_h$ and $\Delta''_h \subseteq \Delta'_h$ such that, each sequent $\Gamma''_h \implies \Delta''_h$ is derivable. Define

$$\begin{aligned}
\Gamma^{\dagger}_{i-1} &= \{J \in \Gamma_{i-1} \mid \text{for some } h,\ J[r_h(\vec{p},\vec{y})/x] \in \Gamma''_h\} \\
\Delta^{\dagger}_{i-1} &= \{J \in \Delta_{i-1} \mid \text{for some } h,\ J[r_h(\vec{p},\vec{y})/x] \in \Delta''_h\} \\
\Gamma'''_h &= \Gamma^{\dagger}_{i-1}[r_h(\vec{p},\vec{y})/x] \cup \{p_{i'} \overset{a_{hi'j}}{\longrightarrow} y_{i'j}\}^{1\leq i'\leq k}_{1\leq j\leq m_{hi}} \cup \{p_{i'} \overset{b_{hi'j}}{\not\rightarrow}\}^{1\leq i'\leq k}_{1\leq j\leq n_{hi'}} \\
\Delta'''_h &= \Delta^{\dagger}_{i-1}[r_h(\vec{p},\vec{y})/x]
\end{aligned}$$

Then $\Gamma^{\dagger}_{i-1}$ and $\Delta^{\dagger}_{i-1}$ are both finite sets (because, for any assertion K, there are only finitely many assertions J that satisfy $J[r_h(\vec{p},\vec{y})/x] = K$). Thus Γ'''_h and Δ'''_h are also finite. Also $\Gamma^{\dagger}_{i-1},\ f(p_1,\ldots,p_k) \xrightarrow{c} x$ is assumable because it is a subset of Γ_{i-1}, and Γ'''_h is assumable because it is a subset of Γ'_h. Moreover, $\Gamma'''_h \supseteq \Gamma''_h$ and $\Delta'''_h \supseteq \Delta''_h$. So, each sequent $\Gamma'''_h \implies \Delta'''_h$ is derivable, by the admissibility of weakening. However, together these sequents form the premises for an application of the ($f\xrightarrow{c}$L) rule, with concluding sequent $\Gamma^{\dagger}_{i-1},\ f(p_1,\ldots,p_k) \xrightarrow{c} x \implies \Delta^{\dagger}_{i-1}$. Because $\Gamma^{\dagger}_{i-1},\ f(p_1,\ldots,p_k) \xrightarrow{c} x \subseteq \Gamma_{i-1}$ and $\Delta^{\dagger}_{i-1} \subseteq \Delta_{i-1}$, it follows that $\Gamma_{i-1} \vdash_{\mathrm{b}} \Delta_{i-1}$, which contradicts the induction hypothesis.

We have shown that there exists h such that $\Gamma'_h \nvdash_{\mathrm{b}} \Delta'_h$. Thus $(\Gamma_i, \Delta_i, \sigma_i, D_i)$ is well-defined with, for a chosen such h,

$$\begin{aligned}
\Gamma_i &= \Gamma'_h & \sigma_i &= z \mapsto (\sigma_{i-1}(z))[r_h(\vec{p},\vec{y})/x] \\
\Delta_i &= \Delta'_h & D_i &= D_{i-1} \cup \{x\}.
\end{aligned}$$

We have already established that Γ_i is assumable, and we have ensured that $\Gamma_i \nvdash_{\mathrm{b}} \Delta_i$. Also $U_i = U_{i-1} \cup \{y_{i'j}\}^{1\leq i\leq k}_{1\leq j\leq m_{hi}}$, so clearly there are infinitely many variables not in U_i. This completes case 1.

Case 2: $\tau_i = (f(p_1,\ldots,p_k) \xrightarrow{c} r_h(\vec{p},\vec{q}),\ \vec{q},\ h)$, where $1 \leq h \leq l_{fc}$. We have $f(p_1,\ldots,p_k) \xrightarrow{c} r_h(\vec{p},\vec{q}) \in \Delta_{i-1}$ where $\mathrm{Vars}(\vec{q}) \subseteq V_{i-1}$. For each i', j with $1 \leq i' \leq k$ and $1 \leq j \leq m_{hi'}$ define:

$$\Gamma'_{i'j} = \Gamma_{i-1} \qquad\qquad \Delta'_{i'j} = \Delta_{i-1},\ p_{i'} \overset{a_{hi'j}}{\longrightarrow} q_{i'j}.$$

Also, for each i', j with $1 \leq i' \leq k$ and $1 \leq j \leq n_{hi'}$ define:

$$\Gamma''_{i'j} \;=\; \Gamma_{i-1} \qquad\qquad \Delta''_{i'j} \;=\; \Delta_{i-1},\, p_{i'} \overset{b_{hi'j}}{\not\rightarrow}$$

By the induction hypothesis, each $\Gamma'_{i'j}$ and $\Delta''_{i'j}$ is assumable.

Suppose, for contradiction, that $\Gamma'_{i'j} \vdash_{\mathrm{b}} \Delta'_{i'j}$ and $\Gamma''_{i'j} \vdash_{\mathrm{b}} \Delta''_{i'j}$, in every case. Then, using the admissibility of weakening, as in the treatment of case 1 above, we have, by an application of the $(f\overset{c}{\rightarrow}\mathrm{R})_h$ rule, that $\Gamma_{i-1} \vdash_{\mathrm{b}} \Delta_{i-1}$, contradicting the induction hypothesis.

Thus: either $\Gamma'_{i'j} \not\vdash_{\mathrm{b}} \Delta'_{i'j}$ for some i', j with $1 \leq i' \leq k$ and $1 \leq j \leq m_{hi'}$; or $\Gamma''_{i'j} \not\vdash_{\mathrm{b}} \Delta''_{i'j}$ for some i', j with $1 \leq i' \leq k$ and $1 \leq j \leq n_{hi'}$. In the first case, $(\Gamma_i, \Delta_i, \sigma_i, D_i)$ is well-defined with $\Gamma_i = \Gamma'_{i'j}$ and $\Delta_i = \Delta'_{i'j}$ for a chosen $i'j$ such that $\Gamma'_{i'j} \not\vdash_{\mathrm{b}} \Delta'_{i'j}$. If the first case does not apply, then $(\Gamma_i, \Delta_i, \sigma_i, D_i)$ is well-defined with $\Gamma_i = \Gamma''_{i'j}$ and $\Delta_i = \Delta''_{i'j}$ for a chosen $i'j$ such that $\Gamma''_{i'j} \not\vdash_{\mathrm{b}} \Delta''_{i'j}$. In either case it holds that Γ_i is assumable and $\Gamma_i \not\vdash_{\mathrm{b}} \Delta_i$. Also, because $\mathrm{Vars}(\vec{q}) \subseteq V_{i-1}$, we have $U_i = U_{i-1}$. So, by the induction hypothesis, there are indeed infinitely many variables not contained in U_i. This completes the argument for case 2.

Remaining cases. These are similar, and are left to the reader. □

Lemma 5.3

(1) $D_i \subseteq U_i$ *and* $D_i \subseteq D_{i+1}$.
(2) $z \notin D_i$ *implies* $\sigma_i(z) = z$.
(3) $z \in U_i$ *implies* $\mathrm{Vars}(\sigma_i(z)) \subseteq V_i$.
(4) $\mathrm{Vars}(\Gamma_i \cup \Delta_i) \subseteq V_i$.
(5) If $j \geq i$ *then* $\sigma_j = \sigma_j \circ \sigma_i$.
(6) If $j \geq i$ *and* $J \in \Gamma_i$ *then either* $\sigma_j(J) \in \Gamma_j$ *or* J *is* $p\overset{c}{\rightarrow}x$ *with* $x \in D_j$.
(7) If $j \geq i$ *and* $J \in \Delta_i$ *then* $\sigma_j(J) \in \Delta_j$.

PROOF. Statements (1)–(4) are proved by induction on i. We omit the easy arguments. Statements (5)–(7) are proved by induction on $j - i$.

For (5), suppose $j = i$. If $z \notin U_i$ then $\sigma_i(z) = z$, by (1) and (2), so $\sigma_i(\sigma_i(z)) = \sigma_i(z)$. If $z \in U_i$ then, by (3), $\mathrm{Vars}(\sigma_i(z)) \subseteq V_i = U_i \backslash D_i$, so $\sigma_i(\sigma_i(z)) = \sigma_i(z)$, by (2), because σ_i is a term homomorphism. Thus indeed $\sigma_i \circ \sigma_i = \sigma_i$.

If $j - i > 0$ then we have $\sigma_{j-1} \circ \sigma_i = \sigma_{j-1}$, by the induction hypothesis. In the case that $\sigma_j = \sigma_{j-1}$, the result is immediate. Otherwise $\sigma_j(z) = \sigma_{j-1}(z)[r_h(\vec{p}, \vec{y})/x]$ for some term $r_h(\vec{p}, \vec{y})$ and variable x. So:

$$\sigma_j(\sigma_i(z)) \;=\; \sigma_{j-1}(\sigma_i(z))[r_h(\vec{p}, \vec{y})/x] \;=\; \sigma_{j-1}(z)[r_h(\vec{p}, \vec{y})/x] \;=\; \sigma_j(z).$$

Thus indeed $\sigma_j \circ \sigma_i = \sigma_j$.

For (6), suppose $J \in \Gamma_i$. By (4), $\mathrm{Vars}(J) \subseteq V_i$. Thus, by (2), $\sigma_i(J) = J$. Thus the base case, $j = i$, is trivial.

If $j - i > 0$ then, by the induction hypothesis, either $\sigma_{j-1}(J) \in \Gamma_{j-1}$ or J is $p \xrightarrow{a} z$ with $z \in D_{j-1}$. In the latter case $z \in D_j$ and we are done. In the former, we have $\sigma_{j-1}(J) \in \Gamma_{j-1}$, and we may assume that J is not of the form $p \xrightarrow{a} z$ with $z \in D_{j-1}$. There are now many subcases. If $\sigma_j = \sigma_{j-1}$ then, however Γ_j is constructed, we have $\Gamma_{j-1} \subseteq \Gamma_j$. So indeed $\sigma_j(J) \in \Gamma_j$. Otherwise we have $\sigma_j = z \mapsto (\sigma_{j-1}(z))[r_h(\vec{p}, \vec{y})/x]$ and $D_j = D_{j-1} \cup \{x\}$. There are now three possibilities. If J is of the form $p \xrightarrow{a} x$, then we have $x \in D_j$ as required. The second possibility is that J is $p \xrightarrow{a} z$ for some $z \neq x$. By the earlier assumption, $z \notin D_{j-1}$ hence $\sigma_{j-1}(z) = z \neq x$. Thus $\sigma_{j-1}(p) \xrightarrow{a} z \in \Gamma_{j-1}$ and so $\sigma_{j-1}(p)[r_h(\vec{p}, \vec{y})/x] \xrightarrow{a} z \in \Gamma_j$, i.e. indeed $\sigma_j(J) \in \Gamma_j$. The third possibility is that J is not an action assertion. In this case, we have immediately that $\sigma_{j-1}(J)[r_h(\vec{p}, \vec{y})/x] \in \Gamma_j$, i.e. $\sigma_j(J) \in \Gamma_j$. This proves statement (6).

The proof of statement (7) is similar, but easier. $\square$

We now define the required model T_c. We write D_ω for $\bigcup_i D_i$ and U_ω for $\bigcup_i U_i$. Define $V_\omega = U_\omega \backslash D_\omega$. The model T_c is determined as the unique model such that: $|T_c| = \{p \mid \mathrm{Vars}(p) \subseteq V_\omega\}$; the operators are interpreted using the term algebra structure; and $x \xrightarrow{a}_{T_C} q$ holds if and only if, for almost all j, it holds that $x \xrightarrow{a} q \in \Gamma_j$ (in which case q must be a process variable). The existence and uniqueness of T_c is guaranteed by Proposition 3.2.

To define the required environment γ_c takes some work. The crucial fact here is that the sequence (σ_i) of substitution functions converges to a limiting substitution function, see Lemma 5.7 below. In order to prove this, it is useful to assign ordinals $< \omega^\omega$ to process terms. Recall that any such ordinal has a unique normal form

$$\omega^l.n_l + \cdots + \omega.n_1 + n_0, \tag{5}$$

where $l, n_l, \ldots, n_0$ are natural numbers with $n_l > 0$ or $l = 0$. The order relation on such ordinals is given by $\omega^l.n_l + \cdots + \omega.n_1 + n_0 > \omega^{l'}.n'_{l'} + \cdots + \omega.n'_1 + n'_0$ iff: either $l > l'$; or $l = l'$ and $n_m > n'_m$ where $m \leq l$ is the greatest number such that $n_m \neq n'_m$. We recall some basic operations of ordinal arithmetic. For an ordinal α with normal form (5) above, the ordinals $\alpha + 1$, $\alpha + \omega$ and $\omega.\alpha$, using the standard non-commutative ordinal operations, have normal forms:

$$\begin{aligned}
\alpha + 1 &= \omega^l.n_l + \cdots + \omega.n_1 + (n_0 + 1) \\
\alpha + \omega &= \omega^l.n_l + \cdots + \omega.(n_1 + 1) + 0 \\
\omega.\alpha &= \omega^{l+1}.n_l + \cdots + \omega^2.n_1 + \omega.n_0 + 0.
\end{aligned}$$

To each term p we assign an ordinal $|p|_i < \omega^\omega$, depending on the structure of Γ_i. The assignment is defined by:

$$\begin{aligned} |x|_i &= \begin{cases} 0 & \text{if there is no assertion } p \xrightarrow{a} x \text{ in } \Gamma_i \\ \omega.|p|_i & \text{if } p \xrightarrow{a} x \in \Gamma_i \end{cases} \\ |f(p_1,\dots,p_k)|_i &= \max(|p_1|_i,\dots,|p_k|_i)+1. \end{aligned}$$

This is easily shown to be well-defined using the well-foundedness of $\lhd_{\Gamma_i}$. In particular, when defining $|x|_i$ in the case that $p \xrightarrow{a} x \in \Gamma_i$, we are given, by the induction hypothesis, that $|y|_i$ is defined for all variables y occurring in p, hence $|p|_i$ is indeed defined.

Lemma 5.4 *For any term $p(x_1,\dots,x_k)$:*

(1) $|q_l|_i \le |p(q_1,\dots,q_k)|_i$ for each $l \in \{1,\dots,k\}$.
(2) $|p(q_1,\dots,q_k)|_i < \max(|q_1|_i,\dots,|q_k|_i)+\omega$.
(3) If $|q_l|_i \le |q'_l|_j$, for all $l \in \{1,\dots,k\}$, then $|p(q_1,\dots,q_k)|_i \le |p(q'_1,\dots,q'_k)|_j$.

PROOF. By easy inductions on the structure of p. □

The next lemma is the reason for the particular choice of ordinal assigment.

Lemma 5.5 *If $x \in D_{i+1}\backslash D_i$, with $\sigma_{i+1}(x) = r_h(\vec{p},\vec{y})$, then the following hold.*

(1) $|r_h(\vec{p},\vec{y})|_{i+1} < |x|_i$.
(2) If $z \in V_i$ then $|z[r_h(\vec{p},\vec{y})/x]|_{i+1} \le |z|_i$.

PROOF. If $x \in D_{i+1}\backslash D_i$, with $\sigma_{i+1}(x) = r_h(\vec{p},\vec{y})$, then there is some assertion $f(p_1,\dots,p_k)\xrightarrow{c}x$ in Γ_i. So

$$\begin{aligned} \Gamma_{i+1} = {} & (\Gamma_i\backslash\{f(p_1,\dots,p_k)\xrightarrow{c}x\})[r_h(\vec{p},\vec{y})/x] \cup \\ & \{p_{i'} \xrightarrow{a_{hi'j}} y_{i'j}\}^{1\le i'\le k}_{1\le j\le m_{hi}} \cup \{p_{i'} \overset{b_{hi'j}}{\not\rightarrow}\}^{1\le i'\le k}_{1\le j\le n_{h'}}. \end{aligned}$$

Define $V = \{z \mid z \lhd^+_{\Gamma_i} x\}$. By the well-foundedness of $\lhd_{\Gamma_i}$, we have $x \notin V$. We show that, for any process term p with $\mathrm{Vars}(p) \subseteq V$, it holds that $|p|_{i+1} = |p|_i$. To see this, observe that if $z \in V$ and $p \xrightarrow{a} z \in \Gamma_i$ then $\mathrm{Vars}(p) \subseteq V$, so $p \xrightarrow{a} z \in \Gamma_{i+1}$. Conversely, suppose that $z \in V$ and $p \xrightarrow{a} z \in \Gamma_{i+1}$. Then z is not any of the $y_{i'j}$, because these are not in U_i, so not in V. Thus p must be $p'[r_h(\vec{p},\vec{y})/x]$ where $p' \xrightarrow{a} z \in \Gamma_i$. But then $\mathrm{Vars}(p') \subseteq V$, so $p' = p'[r_h(\vec{p},\vec{y})/x] = p$. Thus $p \xrightarrow{a} z \in \Gamma_i$. We have shown that if $y \in V$ then $p \xrightarrow{a} y \in \Gamma_i$ if and only if $p \xrightarrow{a} y \in \Gamma_{i+1}$. It is now straightforward from the definitions of $|\cdot|_{i+1}$ and $|\cdot|_i$ that $\mathrm{Vars}(p) \subseteq V$ implies $|p|_{i+1} = |p|_i$.

To prove statement (1), suppose that $\max(|p_1|_i, \ldots, |p_k|_i)$ has the normal form $\omega^l.n_l + \cdots + \omega.n_1 + n_0$. Then

$$|x|_i \;=\; \omega.(\max(|p_1|_i, \ldots, |p_k|_i) + 1) \;=\; \omega^{l+1}.n_l + \cdots + \omega^2.n_1 + \omega.(n_0 + 1) + 0.$$

By the fact shown above, we have $|p_{i'}|_i = |p_{i'}|_{i+1}$ for each $i' \in \{1, \ldots, k\}$. Thus, for each $y_{i'j}$, we have

$$|y_{i'j}|_{i+1} \;=\; \omega.|p_{i'}|_{i+1} = \;\omega.|p_{i'}|_i \;\leq\; \omega^{l+1}.n_l + \cdots + \omega^2.n_1 + \omega.n_0 + 0.$$

Clearly, we also have

$$|p_{i'}|_{i+1} \;=\; |p_{i'}|_i \;\leq\; \omega^{l+1}.n_l + \cdots + \omega^2.n_1 + \omega.n_0 + 0.$$

So, by Lemma 5.4(2),

$$\begin{aligned} |r_h(\vec{p}, \vec{y})|_{i+1} \;&<\; \max(\{|p_1|_{i+1}, \ldots, |p_k|_{i+1}\} \cup \{|y_{i'j'}|_{i+1}\}^{1 \leq i' \leq k}_{1 \leq j \leq m_{hi}}) + \omega \\ &\leq\; (\omega^{l+1}.n_l + \cdots + \omega^2.n_1 + \omega.n_0 + 0) + \omega \\ &=\; \omega^{l+1}.n_l + \cdots + \omega^2.n_1 + \omega.(n_0 + 1) + 0 \\ &=\; |x|_i. \end{aligned}$$

Thus indeed $|r_h(\vec{p}, \vec{y})|_{i+1} < |x|_i$.

Statement (2) is proved by induction over the well-founded relation $\lhd_{\Gamma_i}$. There are three possibilities for $|z[r_h(\vec{p}, \vec{y})/x]|_{i+1}$.

In the first case, $z = x$, in which case $|z[r_h(\vec{p}, \vec{y})/x]|_{i+1} = |r_h(\vec{p}, \vec{y})|_{i+1} < |x|_i = |z|_i$, by statement (1).

In the second case, $z \neq x$ and z does not appear in any assertion $q \xrightarrow{a} z$ in Γ_{i+1}. Then z does not appear in any assertion $q' \xrightarrow{a} z$ in Γ_i, for otherwise $q'[r_h(\vec{p}, \vec{y})/x] \xrightarrow{a} z$ would be in Γ_{i+1}. Thus we have

$$|z[r_h(\vec{p}, \vec{y})/x]|_{i+1} \;=\; |z|_{i+1} \;=\; 0 \;=\; |z|_i.$$

In the third case, $z \neq x$ and z does appear in some assertion $q \xrightarrow{a} z$ in Γ_{i+1}. As $z \in V_i$, it is not equal to any of the $y_{i'j}$, so there must be an assertion $q' \xrightarrow{a} z \in \Gamma_i$ with $q = q'[r_h(\vec{p}, \vec{y})/x]$. For each $z' \in \mathrm{Vars}(q')$, we have $z' \lhd_{\Gamma_i} z$, so, by the induction hypothesis, $|z'[r_h(\vec{p}, \vec{y})/x]|_{i+1} \leq |z'|_i$. Then, by Lemma 5.4(3), we have $|q'[r_h(\vec{p}, \vec{y})/x]|_{i+1} \leq |q'|_i$, i.e. $|q|_{i+1} \leq |q'|_i$. But then

$$|z[r_h(\vec{p}, \vec{y})/x]|_{i+1} \;=\; \omega.|q|_{i+1} \;\leq\; \omega.|q'|_i \;=\; |z|_i.$$

Thus, in all three cases, $|z[r_h(\vec{p}, \vec{y})/x]|_{i+1} \leq |z|_i$. $\Box$

Lemma 5.6 *If $x \in V_i$ and $i \leq j$ then $|\sigma_j(x)|_j \leq |x|_i$.*

PROOF. By induction on $j - i$. When $j = i$, by Lemma 5.3(2), $\sigma_j(x) = x$. If $j - i > 0$ then, by the induction hypothesis, $|\sigma_{j-1}(x)|_{j-1} \leq |x|_i$.

If $\sigma_j = \sigma_{j-1}$, then it is easily checked that each possible case for the definition of Γ_j has the property that $|z|_j = |z|_{j-1}$ for all $z \in V_{j-1}$. Thus, by Lemma 5.4(3), for all p with $\mathrm{Vars}(p) \subseteq V_{j-1}$, we have $|p|_j = |p|_{j-1}$. So $|\sigma_j(x)|_j = |\sigma_{j-1}(x)|_j = |\sigma_{j-1}(x)|_{j-1} \leq |x|_i$.

Alternatively, there exists $z \in D_j \backslash D_{j-1}$, with $\sigma_j(x) = \sigma_{j-1}(x)[r_h(\vec{p}, \vec{y})/z]$ By Lemma 5.5(2), we have, for each $z' \in \mathrm{Vars}(\sigma_{j-1}(x))$ that $|z'[r_h(\vec{p}, \vec{y})/z]|_j \leq |z'|_{j-1}$. So, by Lemma 5.4(3), $|\sigma_{j-1}(x)[r_h(\vec{p}, \vec{y})/z]|_j \leq |\sigma_{j-1}(x)|_{j-1}$. Therefore $|\sigma_j(x)|_j = |\sigma_{j-1}(x)[r_h(\vec{p}, \vec{y})/z]|_j \leq |\sigma_{j-1}(x)|_{j-1} \leq |x|_i$.

Thus indeed $|\sigma_j(x)|_j \leq |x|_i$. □

Lemma 5.7 *For every variable x, there exists j such that $\sigma_{j'}(x) = \sigma_j(x)$ for all $j' \geq j$.*

PROOF. For $x \notin U_\omega$, the statement is trivial. Otherwise we prove: for all ordinals $\alpha < \omega^\omega$, and all $x \in U_\omega$, if there exists i with $x \in \mathrm{Vars}(\Gamma_i \cup \Delta_i)$ and $|x|_i = \alpha$ then there exists j_x such that $\sigma_{j'}(x) = \sigma_{j_x}(x)$ for all $j' \geq j_x$. The proof is by transfinite induction on α.

Suppose that $x \in \mathrm{Vars}(\Gamma_i \cup \Delta_i)$ and $|x|_i = \alpha$. There are two cases.

If $x \notin D_j$ for all j, then the result is trivial, as $\sigma_j(x) = x$ for all j.

Otherwise, $x \in D_{j+1} \backslash D_j$ for some j. Moreover, $j \geq i$ because, by Lemma 5.3(4), for $j' > j$, we have $x \notin \mathrm{Vars}(\Gamma'_j \cup \Delta_{j'})$. By, Lemma 5.6, we have $|x|_j \leq |x|_i = \alpha$. Also, $\sigma_{j+1}(x) = r_h(\vec{p}, \vec{y})$. By, Lemmas 5.4(1) and 5.5(1), for all $z \in \mathrm{Vars}(r_h(\vec{p}, \vec{y}))$, we have $|z|_{j+1} \leq |r_h(\vec{p}, \vec{y})|_{j+1} < |x|_j \leq \alpha$. So, by the induction hypothesis, for each such variable z, there exists j_z such that $\sigma_{j'}(z) = \sigma_{j_z}(z)$ for all $j' \geq j_z$. There are only finitely many variables $z_1, \ldots, z_m$ in $\mathrm{Vars}(r_h(\vec{p}, \vec{y}))$. Define $j_x = \max(j_{z_1}, \ldots, j_{z_m}, j+1)$. Take any $j' \geq j_x$. We must show that $\sigma_{j'}(x) = \sigma_{j_x}(x)$.

For all variables $z \in \mathrm{Vars}(r_h(\vec{p}, \vec{y}))$, it holds that $\sigma_{j'}(z) = \sigma_{j_x}(z)$. So, as $\sigma_{j'}$ and σ_{j_x} are homomorphisms, $\sigma_{j'}(r_h(\vec{p}, \vec{y})) = \sigma_{j_x}(r_h(\vec{p}, \vec{y}))$. Moreover, because $j' \geq j + 1$, we have $\sigma_{j'} = \sigma_{j'} \circ \sigma_{j+1}$, by Lemma 5.3(5). Thus indeed:

$$\sigma_{j'}(x) = \sigma_{j'}(\sigma_{j+1}(x)) = \sigma_{j'}(r_h(\vec{p}, \vec{y})) = \sigma_{j_x}(r_h(\vec{p}, \vec{y})) = \sigma_{j_x}(\sigma_{j+1}(x)) = \sigma_{j_x}(x).$$

□

Lemma 5.7 allows us to define the limiting substitution function, σ_ω, by:

$$\sigma_\omega(x) \;=\; \text{the unique } p \text{ such that } \sigma_i(x) = p \text{ for almost all } i.$$

Lemma 5.8

(1) $z \notin D_\omega$ *implies* $\sigma_\omega(z) = z$.
(2) $z \in U_\omega$ *implies* $\mathrm{Vars}(\sigma_\omega(z)) \subseteq V_\omega$.
(3) $\sigma_\omega = \sigma_\omega \circ \sigma_i$.
(4) If $J \in \Gamma_i$ *then either* $\sigma_\omega(J) \in \Gamma_j$ *for almost all* j*, or* J *is* $p \overset{a}{\rightarrow} x$ *with* $x \in D_\omega$.
(5) If $J \in \Delta_i$ *then* $\sigma_\omega(J) \in \Delta_j$ *for almost all* j

PROOF. All statements follow easily from the definition of σ_ω, using the analogous statements for the various σ_i in Lemma 5.3. □

We can now finally define the required environment. Define $\gamma_c(x)$ to be $\sigma_\omega(x)$ if $x \in U_\omega$ and an arbitrary element of $|T_c|$ otherwise. This is indeed an environment for T_c, by Lemma 5.8(2) above.

Lemma 5.9

(1) $J \in \Gamma_i$ *implies* $T_c \models_{\gamma_c} J$.
(2) $J \in \Delta_i$ *implies* $T_c \not\models_{\gamma_c} J$.

PROOF. First the lemma is proved for assertions $p \overset{c}{\rightarrow} q$ and $p \overset{c}{\not\rightarrow}$, by induction on the structure of $\sigma_\omega(p)$.

Case 1: $p \overset{c}{\rightarrow} x \in \Gamma_i$. If $x \notin D_\omega$ then, by Lemma 5.8(1&4), it holds that $\sigma_\omega(p) \overset{c}{\rightarrow} x \in \Gamma_j$ for almost all $j \geq i$. But then $\sigma_\omega(p)$ must be a variable y, as otherwise for some τ_j of the form $(\sigma_\omega(p) \overset{c}{\rightarrow} x,\, \epsilon,\, 0)$ we would have $x \in D_{j+1}$, a contradiction. So $y \overset{c}{\rightarrow}_{T_c} x$, by the definition of T_c. Thus indeed $T_c \models_{\gamma_c} p \overset{c}{\rightarrow} x$.

Otherwise, if $x \in D_\omega$ then for some $j \geq i$, we have $x \in D_{j+1} \backslash D_j$. Thus, by Lemma 5.3(6), $\sigma_j(p) \overset{c}{\rightarrow} x \in \Gamma_j$ and $\tau_j = (\sigma_j(p) \overset{c}{\rightarrow} x,\, \epsilon,\, 0)$ where $\sigma_j(p)$ has the form $f(p_1, \ldots, p_k)$. So, for some h, we have

$$\{p_{i'} \overset{a_{hi'j}}{\longrightarrow} y_{i'j}\}^{1 \leq i' \leq k}_{1 \leq j \leq m_{hi'}} \cup \{p_{i'} \overset{b_{hi'j}}{\not\rightarrow}\}^{1 \leq i' \leq k}_{1 \leq j \leq n_{hi'}} \;\subseteq\; \Gamma_{j+1}.$$

By Lemma 5.8(3), $\sigma_\omega(p) = \sigma_\omega(\sigma_j(p)) = f(\sigma_\omega(p_1), \ldots, \sigma_\omega(p_k))$. So, by the induction hypothesis, $T_c \models_{\gamma_c} p_{i'} \overset{a_{hi'j}}{\longrightarrow} y_{i'j}$ and $T_c \models_{\gamma_c} p_{i'} \overset{b_{hi'j}}{\not\rightarrow}$, for all appropriate i', j. Then, as T_c is a model of $\mathcal{R}$, we have $T_c \models_{\gamma_c} f(p_1, \ldots, p_k) \overset{c}{\rightarrow} r_h(\vec{p}, \vec{y})$.

By Lemma 5.8(3), $\gamma_c(x) = \sigma_\omega(x) = \sigma_\omega(\sigma_{j+1}(x)) = \sigma_\omega(r_h(\vec{p}, \vec{y}))$. Thus indeed $T_c \models_{\gamma_c} p \xrightarrow{c} x$.

Case 2: $p \xrightarrow{c} r \in \Delta_i$. By Lemma 5.8(5), $\sigma_\omega(p) \xrightarrow{c} \sigma_\omega(r) \in \Delta_j$ for almost all j. Suppose, for contradiction, that $T_c \models_{\gamma_c} p \xrightarrow{c} r$, i.e. that $\sigma_\omega(p) \xrightarrow{c}_{T_c} \sigma_\omega(r)$.

If $\sigma_\omega(p)$ is a variable x then $x \xrightarrow{c}_{T_c} \sigma_\omega(r)$, so, by the definition of T_c, we have that $\sigma_\omega(r)$ is a variable y and $x \xrightarrow{c} y \in \Gamma_j$ for almost all j. But, for almost all j, we also have $x \xrightarrow{c} y \in \Delta_j$. Thus, by the ($\xrightarrow{c}$Ax) rule, $\Gamma_j \vdash_{\mathrm{b}} \Delta_j$ for almost all j, a contradiction.

If $\sigma_\omega(p) = f(p_1, \ldots, p_k)$ then $f(p_1, \ldots, p_k) \xrightarrow{c}_{T_c} \sigma_\omega(r)$. So there exist h and $\vec{q}$, with $\mathrm{Vars}(\vec{q}) \subseteq V_\omega$, such that $\sigma_\omega(r) = r_h(\vec{p}, \vec{q})$ and:

(1) for all i', j' with $1 \leq i' \leq k$ and $1 \leq j' \leq m_{hi'}$, $p_{i'} \xrightarrow{a_{hi'j'}}_{T_c} q_{i'j'}$; and
(2) for all i', j' with $1 \leq i' \leq k$ and $1 \leq j' \leq n_{hi'}$, $p_{i'} \overset{b_{hi'j'}}{\not\rightarrow}_{T_c}$.

Now $\sigma_\omega(p) \xrightarrow{c} \sigma_\omega(r) \in \Delta_j$ and $\mathrm{Vars}(\vec{q}) \subseteq V_j$, for almost all j. So for some $\tau_j = (\sigma_\omega(p) \xrightarrow{c} \sigma_\omega(r),\ \vec{q},\ h)$ it holds that either $p_{i'} \xrightarrow{a_{hi'j'}} q_{i'j'} \in \Delta_{j+1}$ or $p_{i'} \overset{b_{hi'j'}}{\not\rightarrow} \in \Delta_{j+1}$ for some suitable i', j'. Then, by the induction hypothesis, either $T_c \not\models_{\gamma_c} p_{i'} \xrightarrow{a_{hi'j'}} q_{i'j'}$ or $T_c \not\models_{\gamma_c} p_{i'} \overset{b_{hi'j'}}{\not\rightarrow}$. Thus either way contradicts 1 or 2 above.

We have shown that $T_c \models_{\gamma_c} p \xrightarrow{c} r$, whatever the structure of $\sigma_\omega(p)$. This completes case 2.

Remaining cases. There are two more cases: one for assertions $p \overset{c}{\not\rightarrow} \in \Gamma_i$ and one for assertions $p \overset{c}{\not\rightarrow} \in \Delta_i$. These are proved using similar, though easier, arguments to the above.

Having now established the lemma for action and inaction assertions, it remains to deal with logic assertions $p\colon A$. For such assertions, the proof is by induction on the structure of A.

Case 1: $p\colon \langle a \rangle A \in \Gamma_i$. For almost all j we have $\sigma_\omega(p)\colon \langle a \rangle A \in \Gamma_j$. Thus for some $\tau_j = (\sigma_\omega(p)\colon \langle a \rangle A,\ \epsilon,\ 0)$ we have $\{\sigma_\omega(p) \xrightarrow{a} x,\ x\colon A\} \subseteq \Gamma_{j+1}$. Then $T_c \models_{\gamma_c} \sigma_\omega(p) \xrightarrow{a} x$ and $T_c \models_{\gamma_c} x\colon A$, the latter by the induction hypothesis. So $T_c \models_{\gamma_c} p\colon \langle a \rangle A$.

Case 2: $p\colon \langle a \rangle A \in \Delta_i$. For almost all j we have $\sigma_\omega(p)\colon \langle a \rangle A \in \Delta_j$. Suppose, for contradiction, that $T_c \models_{\gamma_c} p\colon \langle a \rangle A$. Thus there exists q with $\mathrm{Vars}(q) \subseteq V_\omega$, such that $\sigma_\omega(p) \xrightarrow{a}_{T_c} q$ and $T_c \models_{\gamma_c} q\colon A$. For almost all j, $\mathrm{Vars}(q) \subseteq V_j$. Thus for some τ_j of the form $(\sigma_\omega(p)\colon \langle a \rangle A,\ q,\ 1)$ we have that either $q\colon A \in \Delta_{j+1}$ or $\sigma_\omega(p) \xrightarrow{a} q \in \Delta_{j+1}$. In the first case, by the induction hypothesis, $T_c \not\models_{\gamma_c} q\colon A$. In the second case $T_c \not\models_{\gamma_c} \sigma_\omega(p) \xrightarrow{a} q$, i.e. $\sigma_\omega(p) \overset{a}{\not\rightarrow}_{T_c} q$. Either alternative contradicts the selection of q. Thus indeed $T_c \not\models_{\gamma_c} p\colon \langle a \rangle A$.

Remaining cases. The cases for the propositional connectives are straightforward, and thus omitted. □

It follows immediately from the lemma above that $\Gamma_0 \not\models \Delta_0$. This completes the proof of Theorem 2.

6 The intended model and ω-completeness

The completeness theorem is relative to entailment over the class of all models of $\mathcal{R}$. Usually one is interested in truth in the intended model $\mathcal{T}_{\mathcal{R}}$, defined after Proposition 3.2. In this section we give conditions under which useful forms of completeness do indeed hold relative to $\mathcal{T}_{\mathcal{R}}$.

The first such completeness theorem is motivated by the fact, remarked upon after Proposition 3.4, that, in any model, the state interpreting a closed process p is bisimilar to the state p in $\mathcal{T}_{\mathcal{R}}$. As we shall see, the proof system is complete for deriving the truth in $\mathcal{T}_{\mathcal{R}}$ of sequents containing only closed process terms. Actually, a stronger result holds. It is enough that every process variable in a sequent is forced to represent a state interpreting a closed process. A simple syntactic condition guarantees that this is the case. We say that a pair of sets of assertions, (Γ, Δ), is *closed generated* if Γ is assumable and also every variable $x \in \mathrm{Vars}(\Gamma \cup \Delta)$ appears in an assertion of the form $p \xrightarrow{a} x \in \Gamma$. This condition ensures that each minimal variable x under $\lhd_\Gamma$ appears in an assertion of the form $p \xrightarrow{a} x \in \Gamma$ where p is closed, and thus, by the well-foundedness of $\lhd_\Gamma$, each variable is a descendent of a closed process term.

Theorem 3 *If* (Γ, Δ) *is closed generated and* $\Gamma \models_{\mathcal{T}_{\mathcal{R}}} \Delta$ *then* $\Gamma \vdash \Delta$.

Here, in view of Corollary 1, we write $\Gamma \vdash \Delta$ to mean $\Gamma \vdash_{\mathrm{b}} \Delta$ or $\Gamma \vdash_{\mathrm{f}} \Delta$.

PROOF. Suppose that (Γ_0, Δ_0) is closed-generated and $\Gamma_0 \not\vdash \Delta_0$. Let T_c be the model constructed in the proof of Theorem 2. By Lemma 5.9, we have $\Gamma_0 \not\models_{T_c} \Delta_0$. We use the construction of T_c from Section 5 to show that $T_c = \mathcal{T}_{\mathcal{R}}$.

First, observe that, by a straightforward induction on i, each (Γ_i, Δ_i) is closed-generated. One then proves easily, by well-founded induction on $\lhd_{\Gamma_i}$ that, for each $x \in \mathrm{Vars}(\Gamma_i \cup \Delta_i)$, it holds that $|x|_i > 0$, where $|x|_i$ is the ordinal assigned in Section 5.

Next, we show that $x \in U_\omega$ implies that $\sigma_\omega(x)$ is closed. Suppose for contradiction that there exists some $x \in U_\omega$ such that $\mathrm{Vars}(\sigma_\omega(x)) \neq \emptyset$. Using the ordinal assigment, select x and i with $|x|_i$ minimal such that $x \in \mathrm{Vars}(\Gamma_i \cup \Delta)$

and $\mathrm{Vars}(\sigma_\omega(x)) \neq \emptyset$. As (Γ_i, Δ_i) is closed-generated, there is some assertion $q \xrightarrow{c} x \in \Gamma_i$. By Lemma 5.8(4): either (i) $x \in D_\omega$; or (ii) $\sigma_\omega(q) \xrightarrow{c} x \in \Gamma_j$ for almost all j.

In case (i), there exists $j \geq i$ such that $x \in D_{j+1} \backslash D_j$, where $\sigma_j(q) \xrightarrow{c} x \in \Gamma_i$ and $\sigma_{j+1} = z \mapsto (\sigma_j(z))[r_h(\vec{p}, \vec{y})/x]$. By Lemmas 5.4(1), 5.5(1), and 5.6, for each $z \in \mathrm{Vars}(r_h(\vec{p}, \vec{y}))$, we have

$$|z|_{j+1} \;\leq\; |r_h(\vec{p}, \vec{y})|_{j+1} \;<\; |x|_j \;=\; |\sigma_j(x)|_j \;\leq\; |x|_i.$$

By the minimality assumption on $|x|_i$, for each such z, it holds that $\sigma_\omega(z)$ is closed. Thus $\sigma_\omega(r_h(\vec{p}, \vec{y}))$ is closed. By Lemma 5.8(3), we have

$$\sigma_\omega(x) \;=\; \sigma_\omega(\sigma_{j+1}(x)) \;=\; \sigma_\omega(r_h(\vec{p}, \vec{y})).$$

Thus $\sigma_\omega(x)$ is closed, contradicting the choice of x.

In case (ii), $\sigma_\omega(q) \xrightarrow{c} x \in \Gamma_j$ for almost all j, and $x \notin D_\omega$. Then $\sigma_\omega(q)$ must be a variable y, as otherwise for some τ_j of the form $(\sigma_\omega(p) \xrightarrow{c} x,\ \epsilon,\ 0)$ we would have $x \in D_{j+1}$, a contradiction. Therefore q is itself a variable z, with $\sigma_\omega(z) = y$. But then, by the definition of the ordinal assignment, $|x|_i = \omega.|z|_i > |z|_i$, with the latter inequality because $0 < |z|_i < \omega^\omega$. Thus, by the minimality assumption on $|x|_i$, we have $\sigma_\omega(z)$ is closed, contradicting $\sigma_\omega(z) = y$.

We have proved that, for every $x \in U_\omega$, it holds that $\mathrm{Vars}(\sigma_\omega(x))$ is closed. Therefore, by Lemma 5.8(1), $U_\omega = D_\omega$. Thus $V_\omega = \emptyset$. So $|T_c|$ is the set of closed processes. By Proposition 3.2, it follows that $T_c = T_{\mathcal{R}}$. Thus indeed $\Gamma_0 \not\models_{T_{\mathcal{R}}} \Delta_0$. $\square$

Given Theorems 1 and 2, an equivalent statement to Theorem 3 is that, when (Γ, Δ) is closed-generated, then $\Gamma \models_{T_{\mathcal{R}}} \Delta$ implies $\Gamma \models \Delta$. It is interesting to note that conditions (A1) and (A2) on the assumability of Γ are essential for this implication to hold. For example, we have that $a.0 \xrightarrow{a} 0 + 0 \models_{T_{\mathcal{R}}}$ but not that $a.0 \xrightarrow{a} 0 + 0 \models$, because 0 and $0 + 0$ have the same denotation in the model obtained by quotienting $T_{\mathcal{R}}$ by bisimilarity.

The restriction to closed generated consequences does not fully exploit the expressivity of sequents containing open terms. One would also like a completeness result for sequents in which the variables need not derive from closed processes. Indeed such sequents are used crucially to express the parametrized verification goals discussed in Section 1. What we seek is a form of *ω-completeness*, i.e. completeness relative to all environments interpreting process variables as closed processes in $T_{\mathcal{R}}$. In order to obtain such a result, it is necessary to make some very mild expressivity assumptions on the GSOS system $\mathcal{R}$. Recall the definition of the transition system F of finite processes from Section 3.

Definition 6.1 We say that $\mathcal{R}$ *represents every finite process* if, for every $u \in F$ there exists $p \in T_{\mathcal{R}}$ such that $u \sim p$.

To motivate this definition, let us consider what happens when $\mathcal{R}$ does not represent every finite process. Accordingly, suppose that there exists some finite process $u \in F$ for which there is no $p \in T_{\mathcal{R}}$ with $u \sim p$. Let $\chi(u)$ be the characteristic formula of u, given by Proposition 3.7(2). Then we have $\models_{T_{\mathcal{R}}} x\colon \neg(\chi(u))$. However, clearly $\not\models x\colon \neg(\chi(u))$ because one can find models in which states bisimilar to u do exist. Thus ω-completeness fails. Therefore, a necessary condition for ω-completeness, for even the simplest non-closed-generated sequents, is that $\mathcal{R}$ represents every finite process. Surprisingly, this turns out to be a sufficient condition for ω-completeness to hold for a very wide class of sequents.

Theorem 4 (ω-completeness) *Suppose that $\mathcal{R}$ represents every finite process. Then, for finite Γ, Δ such that Γ is assumable and Δ contains no action assertions, $\Gamma \models_{T_{\mathcal{R}}} \Delta$ implies $\Gamma \vdash \Delta$.*

The condition that $\mathcal{R}$ represents every finite process is rather mild. For example, it is satisfied by any process algebra containing prefix, zero and sum. [5]

The restrictions on Γ and Δ in Theorem 4 are necessary. The finiteness condition is required because the consequence relation $\vdash$ is compact, whereas $\models_{T_{\mathcal{R}}}$ need not be. For example, take $\mathcal{R}$ to be the GSOS system containing just the prefix, zero and sum operators. Then it holds that $\models_{T_{\mathcal{R}}} \{x\colon [a]^n\bot \mid n \geq 0\}$, but it is clear that $\not\models \{x\colon [a]^n\bot \mid n \geq 0\}$, because one can find models containing processes able to perform infinite sequences of a transitions. For an example showing why Δ is required to contain no action assertion, observe that it possible to construct a GSOS system, containing the prefix and zero operators, that represents every finite process and in which the only closed process term bisimilar to the zero process is 0 itself (so there is necessarily no sum operator). If $\mathcal{R}$ is such a system then $\{x\colon [a]\bot \mid a \in \mathrm{Act}\} \models_{T_{\mathcal{R}}} a.0 \xrightarrow{a} x$, but the corresponding sequent is not provable, because one can find models in which there are two distinct states bisimilar to 0. Note that this type of counterexample does not work for those GSOS systems in which every finite process is represented by an infinite number of distinct closed terms (such as any system with prefix, zero and sum). It seems possible that, for such systems, ω-completeness might hold for arbitrary finite Δ.

Theorem 4 is proved by establishing that, under the conditions of the theorem, $\Gamma \not\models \Delta$ implies $\Gamma \not\models_{T_{\mathcal{R}}} \Delta$. Suppose then that $\mathcal{R}$ represents every finite process and that we have T and γ such that, for all $J \in \Gamma$, $T \models_\gamma J$ and, for all $K \in \Delta$, $T \not\models_\gamma K$. We must define a $T_{\mathcal{R}}$-environment γ' such that, for all $J \in \Gamma$, $T_{\mathcal{R}} \models_{\gamma'} J$ and, for all $K \in \Delta$, $T_{\mathcal{R}} \not\models_{\gamma'} K$.

[5] These operators are assumed present in the definition of GSOS system in [5].

We shall define γ' so that, for each x, it holds that $\gamma'(x) \sim_m \gamma(x)$ for some m depending on x, using the m-th approximation to bisimilarity from Section 3. To determine m we assign a *depth*, $d(p)$, to each process term p by:

$$\begin{aligned} d(x) &= \begin{cases} d(p) + 1 \text{ if } p \xrightarrow{a} x \in \Gamma, \\ 0 \qquad\qquad \text{otherwise,} \end{cases} \\ d(f(p_1, \ldots, p_k)) &= max\{d(p_1), \ldots, d(p_k)\}. \end{aligned}$$

It follows from the well-foundedness of $\lhd_\Gamma$ that $d(p)$ is well-defined. Define

$$\begin{aligned} n \;=\; \max(&\{d(p) + 1 \mid p \xrightarrow{a} x \;\in\; \Gamma \text{ or } p \overset{a}{\nrightarrow} \;\in\; \Gamma \cup \Delta\} \cup \\ &\{d(p) + \mathrm{md}(A) \mid p\colon A \;\in\; \Gamma \cup \Delta\}), \end{aligned}$$

using the finiteness of Γ and Δ. By a trivial induction on the structure of terms, one sees that, for any term p, it holds that $n \geq d(p)$.

Lemma 6.2 *There exists a $T_\mathcal{R}$-environment γ' such that:*

(1) $\gamma'(p) \sim_{n-d(p)} \gamma(p)$, and
(2) $p \xrightarrow{a} y \in \Gamma$ implies $\gamma'(p) \xrightarrow{a}_{T_\mathcal{R}} \gamma'(y)$.

PROOF. For each $m \in \{0, \ldots, n\}$, we define $\gamma'(x)$ on variables x with $d(x) = m$, assuming it already defined on variables y with $d(y) < m$. Moreover, we ensure that: (i) property (1) holds for all p with $\mathrm{Vars}(p) \subseteq \{y \mid d(y) \leq m\}$; and (ii) property (2) holds whenever $d(y) \leq m$. Accordingly, let x be any variable with $d(x) = m$.

When $m = 0$, by Proposition 3.6(1), there exists a finite process $u \in F_n$ such that $\gamma(x) \sim_n u$. Because $\mathcal{R}$ represents every finite process, there exists $q \in T_\mathcal{R}$ such that $q \sim u$. By Proposition 3.3, it follows that $q \sim_n \gamma(x)$. Define $\gamma'(x) = q$. We thus have $\gamma'(x) \sim_{n-d(x)} \gamma(x)$. Also note that there is no assertion $p \xrightarrow{a} x \in \Gamma$ because $d(x) = 0$.

When $m > 0$, we use as induction hypothesis that (i) holds when m is replaced by $m - 1$. As $d(x) > 0$, we have $p \xrightarrow{a} x \in \Gamma$ for some p. So $\gamma(p) \xrightarrow{a}_T \gamma(x)$. But $d(p) = d(x) - 1$ so, for all $y \in \mathrm{Vars}(p)$, we have $d(y) < m$. Thus, by the induction hypothesis, $\gamma'(p) \sim_{n-(d(x)-1)} \gamma(p)$, i.e. $\gamma'(p) \sim_{(n-d(x))+1} \gamma(p)$. As $\gamma(p) \xrightarrow{a}_T \gamma(x)$, there exists $q \in |T_\mathcal{R}|$ such that $\gamma'(p) \xrightarrow{a}_{T_\mathcal{R}} q$ and $q \sim_{n-d(x)} \gamma(x)$. Define $\gamma'(x) = q$. Thus we have $\gamma'(x) \sim_{n-d(x)} \gamma(x)$ and $\gamma'(p) \xrightarrow{a}_{T_\mathcal{R}} \gamma'(x)$

We now have γ' defined on all variables x with $d(x) \leq m$. We must show that (i) and (ii) hold. By the definition of $\gamma'(x)$ above, we have ensured that $\gamma'(x) \sim_{n-d(x)} \gamma(x)$ and also that $p \xrightarrow{a} x \in \Gamma$ implies $\gamma'(p) \xrightarrow{a}_{T_\mathcal{R}} \gamma'(x)$. It remains to show that $\gamma'(p) \sim_{n-d(p)} \gamma(p)$ for any p with $\mathrm{Vars}(p) \subseteq \{x \mid d(x) \leq m\}$.

Consider any such p. Then, for each $x \in \mathrm{Vars}(p)$ we have $\gamma'(x) \sim_{n-d(x)} \gamma(x)$. But $d(x) \leq d(p)$ so also $\gamma'(x) \sim_{n-d(p)} \gamma(x)$. Thus, by Proposition 3.4, indeed $\gamma'(p) \sim_{n-d(p)} \gamma(p)$. $\square$

Lemma 6.3 *For all $J \in \Gamma$, $T_{\mathcal{R}} \models_{\gamma'} J$ and, for all $K \in \Delta$, $T_{\mathcal{R}} \not\models_{\gamma'} K$.*

PROOF. We consider the various cases in turn.

If $p \xrightarrow{a} x \in \Gamma$ then, by Lemma 6.2(2), $\gamma'(p) \xrightarrow{a}_{T_{\mathcal{R}}} \gamma'(x)$, i.e. $T_{\mathcal{R}} \models_{\gamma'} p \xrightarrow{a} x$.

If $p \overset{a}{\not\rightarrow} \in \Gamma$ then $T \models_{\gamma} p \overset{a}{\not\rightarrow}$, i.e. $\gamma(p) \overset{a}{\not\rightarrow}_T$. By Lemma 6.2(1), $\gamma'(p) \sim_{n-d(p)} \gamma(p)$. By the definition of n, we have $n - d(p) \geq 1$. Thus, $\gamma'(p) \sim_1 \gamma(p)$. It follows that $\gamma'(p) \overset{a}{\not\rightarrow}_{T_{\mathcal{R}}}$, i.e. $T_{\mathcal{R}} \models_{\gamma'} p \overset{a}{\not\rightarrow}$.

If $p\colon A \in \Gamma$ then $T \models_{\gamma} p\colon A$, i.e. $\gamma(p) \Vdash_T A$. Therefore, by Lemma 6.2(1), $\gamma'(p) \sim_{n-d(p)} \gamma(p)$. By the definition of n, we have $n - d(p) \geq \mathrm{md}(A)$. Thus, $\gamma'(p) \sim_{\mathrm{md}(A)} \gamma(p)$. So, by Proposition 3.5, $\gamma'(p) \Vdash_{T_{\mathcal{R}}} A$, i.e. $T_{\mathcal{R}} \models_{\gamma'} p\colon A$.

The cases for inaction and logic assertions in Δ are similar to the corresponding cases for Γ above. $\square$

Lemma 6.3 states that $\Gamma \not\models_{T_{\mathcal{R}}} \Delta$. This completes the proof of Theorem 4.

As far as we are aware, there is no precursor to Theorem 4 in the literature. The closest work is that of Stirling, who, in his proof system for CCS [25], used special sequents $A, B \vdash C$ for stating parametrized goals of the form $x\colon A,\ y\colon B \implies x|y\colon C$. Using such sequents, Stirling obtained completeness for ordinary verification goals $p\colon A$. However, he did not obtain the completeness of his proof system for the sequents $A, B \vdash C$. In our approach, sequents expressing parametrized goals arise in a uniform way and are available for all process operators in the language. Moreover, Theorem 4 shows that our proof system is complete for establishing all such sequents.

7 Conclusions

Previous work on compositional proof systems for process algebras (see, e.g., [30,25,2]) has often involved ingenious ideas that work for the operators under consideration. In this paper we have shown how such proof systems may be derived in a uniform way for any GSOS system. Our use of GSOS systems may be seen as analogous to that of [1], where it is shown how to derive complete equational axiomatizations of bisimilarity from arbitrary GSOS rule specifications. In this paper, we have pursued a similar programme for modal properties rather than equations.

Crucial to our approach is the use of a sufficiently expressive form of sequent, permitting the incorporation of operational semantics into the proof rules. In addition to the general completeness and cut-elimination theorems, a particularly important improvement on previous work has been the proof of an ω-completeness theorem for a wide class of sequents, including those expressing parametrized verification goals.

Regarding possible improvements to our work, there are several limitations inherent in the use of GSOS systems. One is the restriction to a finite set of actions. There are straightforward generalizations to infinite action sets which, however, involve the use of infinitary rules. It would be interesting to develop a natural class of finitary rules for dealing with infinite action sets, perhaps by using proof rules based on an algebra of action terms. A further limitation is that we have not included a recursion operator in the GSOS system. As remarked in [5], any process defined by guarded recursion can be incorporated by including a new process constant for the process and giving it explicit operational rules. However, it would be better to include direct proof rules for guarded recursion in the sequent calculus. Although the definition of such an extension of our proof system is not difficult, it seems a nontrivial task to extend our proof of completeness to cover it, although in principle it should be possible to do so.

A severe practical limitation of our work is the use of Hennessy-Milner logic, which is too weak to express interesting temporal properties of programs. Since the research in this paper was first carried out, there has been significant progress on the extension of the methods in this paper to richer logics, such as Kozen's modal μ-calculus [17]. This work is surveyed in Section 8 below.

More generally, the idea of deriving Gentzen-style rules from operational semantics is by no means restricted to GSOS-specified process algebras. Indeed, since the research in this paper was first carried out, there have been several applications to richer process languages and other programming paradigms. Once again, these are surveyed in Section 8 below.

8 Epilogue

The research in this paper was first presented at the 1995 IEEE *Logic in Computer Science* conference, under the title: "Compositionality via cut-elimination: Hennessy-Milner logic for an arbitrary GSOS" [23]. The choice of title was unfortunate, as the slogan "compositionality via cut-elimination" is misleading. At the time of the original conference submission (December 1994), I was failing to distinguish between compositional and structural reasoning, and the original title was based on a conflation of the two. By the time of the LICS conference (June 1995), I had realised my mistake, and I publicly retracted the title during my talk. But that was too late for the proceedings version! The introduction to the present paper presents what I believe is a correct account of compositional and structural reasoning and their relationship to sequent calculus. Thus, in spite of the title of [23], the cut rule is essential for compositional verification.

Instead, the significance of the admissibility of cut is twofold. First, it demonstrates that the left and right rules for process operators properly complement each other, and so justifies the formulation of these rules. Second, it shows that structural reasoning alone suffices to establish any goal. It is unclear, however, whether or not the admissibility of cut can (or even should) be maintained in extensions of the proof system to richer logics and programming languages (for example, the extensions discussed below). Indeed, even if cut is not admissible in such systems, its admissibility in the system of this paper still amounts to justification of the formulation of the rules. Of course, structural reasoning is then no longer sufficient on its own. But this is rather a fact of life than a problem. As discussed in Section 1, one anyway expects to use a combination of reasoning methods in program verification.

The other main contributions of the original paper were:

(1) The notion of GSOS model.
(2) The use of sequent calculus as a formalism for process verification.
(3) The derivation of proof rules from operational semantics.
(4) The completeness and ω-completeness results.

We end by discussing these within the context of subsequent developments.

In [23], the notion of GSOS model was used purely as a technical tool, needed to establish a general completeness result (Theorem 2 of the present paper). Nevertheless, it seemed a natural notion. This was later substantiated by Turi and Plotkin in their category-theoretic account of the operational and denotational semantics of GSOS systems [27,28]. In their work, a GSOS system gives rise to a monad on the category of transition systems with functional bisimulations as morphisms. The algebras of this monad turn out to be exactly the GSOS models in the sense of Definition 3.1.

The idea of using sequent calculus for process verification was proposed independently by Dam in [7], written at the same time as [23]. Dam was also concerned with obtaining compositional proof systems allowing natural forms of reasoning. However, in contrast to [23], Dam used the much more expressive modal μ-calculus of Kozen [17], and concentrated on soundness results (which are nontrivial in that setting) rather than completeness results (which are unachievable for the μ-calculus, see below).

There are two alternative approaches to extending the approach of this paper to the modal μ-calculus. The most obvious is to include induction and coinduction rules for least and greatest fixed points respectively. For example, natural proof rules for the greatest-fixed-point operator are given by:

$$\frac{\Gamma,\, p\colon A[\nu X.A/X] \implies \Delta}{\Gamma,\, p\colon \nu X.A \implies \Delta} \qquad \frac{\Gamma,\, x\colon B \implies x\colon A[B/X],\, \Delta \quad \Gamma \implies p\colon B,\, \Delta}{\Gamma \implies p\colon \nu X.A,\, \Delta}$$

where, in the right-hand rule, x must not appear in the rule conclusion. The right-hand rule is a coinduction rule, based on using a formula B, representing a post-fixed point for the operator $X \mapsto A$, as a coinduction hypothesis. Dual rules are applicable to least fixed points.

Unfortunately, such rules on their own appear too weak to establish any interesting properties of processes. The problem is that the induction and coinduction hypotheses required in practice, which often involve syntactic closure conditions on classes of processes, are not expressible in the modal μ-calculus. In his MSc dissertation [4], Beattie showed that by moving to a first-order μ-calculus, including processes as terms, such induction and coinduction rules can be used to prove interesting properties, including useful parametrized verification goals. However, there are two drawbacks to this approach. First, the proofs using induction and coinduction turn out to be long and awkward. Second, moving to a first-order logic with process terms amounts to a paradigm shift from an *endogenous* logic to an *exogenous* logic, in the sense of Pnueli [19]. In endogenous logics, such as Hennessy-Milner logic and the modal μ-calculus, the language of properties (logical formulas) is independent of the language of programs (process terms). One would like proof systems for such logics to maintain this desirable separation.

The second approach to including fixed-points in the logics is to adopt a tableau-based approach to derivations, influenced by local model checking [26]. Under this approach, one simply includes unfolding rules for fixed points, e.g.

$$\frac{\Gamma,\, p\colon A[\nu X.A/X] \implies \Delta}{\Gamma,\, p\colon \nu X.A \implies \Delta} \qquad \frac{\Gamma \implies p\colon A[\nu X.A/X],\, \Delta}{\Gamma \implies p\colon \nu X.A,\, \Delta}$$

The power of the method is achieved by identifying global combinatorial *discharge conditions* on derivation trees, involving repetitions of sequents, that suffice for the concluding sequent of a derivation to be valid. Such conditions do not require every leaf of the derivation tree to be an axiom.

The adaptation of such tableau-based techniques to sequent calculus including cut (required for compositionality) is nontrivial. Addressing this problem has been a main concern of Dam [7,8,11]. In [11], Dam and Gurov suggest extending the modal μ-calculus with primitives for *explicit approximants* of fixed points, implemented by adding ordinal variables to the syntax. This approach has led to an elegant characterization of sound discharge conditions in terms of Büchi automata over derivation trees [24]. In related work, Schöpp and I have shown that explicit approximants can also be expressed using a propositional extension of the modal μ-calculus with modalities for approximant modification [21,22]. These various extensions of the μ-calculus with explicit approximants all retain the desirable property of being endogenous logics.

When a proof system is formulated for an expressive temporal logic, such as the modal μ-calculus, there is no hope of achieving a general completeness theorem, even for basic sequents of the form $\Longrightarrow p\colon A$ with p a closed process. Indeed, whenever the *model checking problem*, of whether $p \Vdash A$ holds, is undecidable, completeness cannot hold for any proof system in which derivable assertions are recursively enumerable, because one would then be able to decide $p \Vdash A$ via semidecision procedures for $\vdash p\colon A$ and $\vdash p\colon \neg A$. Such undecidability, and hence incompleteness, results apply even to the simplest process algebras containing parallel and recursion, such as BPP [12].

Because unqualified completeness results are unavailable, one instead seeks restricted completeness results for cases in which such results are, at least, achievable in theory. Sequent calculus has proved a successful platform for obtaining such results. The results of the present paper deal comprehensively with the case in which the logic is restricted to Hennessy-Milner logic. For the modal μ-calculus, sequent-based proof systems based on tableau-style unfolding have yielded several restricted completeness theorems. By a reduction to Walukiewicz' completeness theorem for Kozen's axiomatization of μ-calculus validity [29], Dam and Gurov established completeness for sequents of the form $\Longrightarrow x\colon A$ [11]. Also, Dam showed that tableau-based methods yield completeness for the model checking problem for finite state processes [7,8]. Recently, Schöpp and I have extended this latter result to important classes of infinite state processes: context-free processes [20,21] and pushdown processes [22].

Of equal importance to such theoretical completeness results is the question of *practical completeness*: does a proof system suffice to establish the verification goals needed in practice? The only way to investigate this question is by means of case studies. Such case studies have been carried out within adaptations of the sequent-based proof system to richer process languages and other computational paradigms. For example, related proof systems have been developed for the π-calculus [9,6], Erlang [10,13] and JavaCard [3]. As well as demonstrating the adaptability of the methods of the present paper, this accumulating body of work does seem to confirm that sequent-based reasoning is a viable approach to the formal verification of programs.

References

[1] L. Aceto, B. Bloom, and F. Vaandrager. Turning SOS rules into equations. *Information and Computation*, 111:1–52, 1994.

[2] H.R. Anderson, C.P. Stirling, and G. Winskel. A compositional proof system for the modal μ-calculus. In *Proceedings of 9th IEEE Symposium on Logic in Computer Science*, pages 144–153, 1994.

[3] G. Barthe, D. Gurov, and M. Huisman Compositional Verification of Secure Applet Interactions. In *Proceedings of FASE'02*, Springer LNCS 2306, pages 15–32, 2002.

[4] J. Beattie. *A Sequent Calculus for Proving Properties of Processes.* MSc dissertation, Division of Informatics, University of Edinburgh, 1997.

[5] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *J. Assoc. Comput. Mach.*, 42:232–268, 1995.

[6] L. Caires and L. Cardelli. A spatial logic for concurrency (part II). In *CONCUR 2002*, Proceedings of 13th International Conference on Concurrency Theory, Springer LNCS 2421, pp. 209–225, 2002.

[7] M. Dam. Compositional proof systems for model checking infinite state processes. In *International Conference on Concurrency Theory*, pages 12–26, 1995.

[8] M. Dam. Proving properties of dynamic process networks. *Information and Computation*, 140:95–114, 1998.

[9] M. Dam. Proof systems for π-calculus logics. In R. de Queiroz, editor, *Logic for Concurrency and Synchronisation.* OUP, 2001.

[10] M. Dam, L. Fredlund, and D. Gurov. Toward parametric verification of open distributed systems. In A. Pnueli H. Langmaack and W.-P. de Roever, editors, *Compositionality: the Significant Difference.* Springer, 1998.

[11] M. Dam and D. Gurov. μ-calculus with explicit points and approximations. *Journal of Logic and Computation*, 12:255–269, 2002.

[12] J. Esparza. Decidability of model-checking for infinite-state concurrent systems. *Acta Informatica,* 34:85–107, 1997.

[13] L. Fredlund. A framework for reasoning about Erlang code. PhD Thesis, Swedish Institute of Computer Science, 2001.

[14] G. Gentzen. Investigations into logical deduction. In M.E. Szabo, editor, *The collected papers of Gerhard Gentzen*, pages 68–128. North Holland Publishing Company, 1969. Originally published in German, 1935.

[15] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control*, 68:125–145, 1986.

[16] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. Assoc. Comput. Mach.*, 32:137–161, 1985.

[17] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[18] R. Milner. *Communication and Concurrency.* Prentice Hall international series in computer science. Prentice Hall, 1989.

[19] A. Pnueli. The temporal logic of programs. In *Proceedings of 19th IEEE Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[20] U. Schöpp. Formal verification of processes. MSc Dissertation, Division of Informatics, University of Edinburgh, 2001.

[21] U. Schöpp and A.K. Simpson. Verifying temporal properties using explicit approximants: completeness for context-free processes. In *Proceedings of FOSSACS 2002*, Springer LNCS 2303, pages 372-386, 2002.

[22] U. Schöpp and A.K. Simpson. Verifying temporal properties using explicit approximants: completeness for pushdown processes. In preparation, 2003.

[23] A.K. Simpson. Compositionality via cut-elimination: Hennessy-Milner logic for an arbitrary GSOS. In *Proceedings of 10th IEEE Symposium on Logic in Computer Science*, pages 420–430, 1995.

[24] C. Sprenger and M. Dam A note on global induction mechanisms in a μ-calculus with explicit approximations. In *Proceedings of FICS'02*, 2002.

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

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

[27] D. Turi. *Functorial Operational Semantics and its Denotational Dual.* PhD thesis, Free University, Amsterdam, 1996.

[28] D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proceedings of 12th IEEE Symposium on Logic in Computer Science*, pages 280–291, 1997.

[29] I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation*, 157:142–182, 2000.

[30] G. Winskel. A complete proof system for SCCS with modal assertions. *Fundamenta Informaticae*, IX:401–420, 1986.