Proving Properties of Dynamic Process Networks

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

[*]Work supported by ESPRIT BRA project 8130 "LOMAPS". Preliminary version appeared as "Compositional Proof Systems for Model Checking Infinite State Processes", Proc. CONCUR'95, Lecture Notes in Computer Science, vol. 962, pp. 12–26.

Abstract

We present the first compositional proof system for checking processes against formulas in the modal μ-calculus which is capable of handling dynamic process networks. The proof system is obtained in a systematic way from the operational semantics of the underlying process algebra. A non-trivial proof example is given, and the proof system is shown to be sound in general, and complete for finite-state processes.

1 Introduction

In this paper we address the problem of verifying modal μ-calculus properties of general infinite-state processes, and we present what we believe to be the first genuinely compositional solution to this problem.

The value of compositionality in program logics is well established. Compositionality allows better structuring and decomposition of the verification task, it allows reuse of proofs, and it allows reasoning about partially instantiated programs, thus supporting program synthesis. Even more fundamentally it allows, at least in principle, verification exercises to be undertaken which are beyond the scope of more global approaches because the set of reachable global states grows in an unbounded manner. The problem of how to build compositional proof systems for concurrent systems, however, has long been recognised as a very difficult one. Many techniques have been suggested in the literature, including the assumption-guarantee paradigm (cf. (Jones, 1983; Pnueli, 1985; Stirling, 1988; Abadi and Lamport, 1993; Grumberg and Long, 1994)), quotienting and reduction (cf. (Larsen and Liu, 1991; Andersen et al, 1994)), techniques exploiting environment-relativised transition semantics (cf. (Abrahamson, 1979; Barringer et al, 1984)), and simulation relations (cf. (Jonsson, 1994; Grumberg and Long, 1994)). All these techniques, however, give only partial and ad-hoc solutions in that they work only for particular concurrency primitives, static process networks and, most often, linear time logic.

Much recent research in the area has focused on process algebra and the modal μ-calculus. The modal (or propositional) μ-calculus, L_μ, due to Kozen (Kozen, 1983), is a powerful temporal logic obtained by adding least and greatest solutions of equations to minimal modal logic. Many important temporal logics such as CTL and CTL* can be represented in L_μ (c.f. (Emerson and Lei, 1986; Dam, 1994, A)), and a large number of algorithms, tableau systems, and proof systems for verifying processes against modal μ-calculus specifications by some form of global state space exploration have been given (c.f. (Bradfield and Stirling, 1992; Cleaveland et al, 1992; Emerson and Lei, 1986; Larsen, 1992; Stirling and Walker, 1991; Winskel, 1991) and many others). Compositional accounts have been developed based on some form of quotienting, or reduction (cf. (Larsen and Liu, 1991; Andersen et al, 1994)). These approaches, however, are only applicable for finite-state processes, or at least when the holding of a property depends only on a finite portion of a potentially infinite-state process.

Finite-state processes, however, are inadequate as modelling tools in many practical situations. Value- or channel passing, for instance, can cause even the simplest processes to become infinite state. While some decidability results can be obtained in the absence of process spawning (c.f. (Dam, 1994, B)), in general the model checking problem becomes undecidable, even in very sparse fragments of, e.g., CCS (Esparza, 1997). Process spawning, however, is needed in many applications: Unbounded buffers, dynamic resource or process creation/forking, data types and higher order features. In fact it is hard to conceive of useful program logics for modern concurrent functional languages such as CML, Facile, Erlang, or PICT that can not deal with process spawning, and indeed the development of such logics is one long-term aim of the research reported here.

Because of undecidability, finitary proof systems for proving temporal properties (like: termination) of general infinite state processes will necessarily be incomplete. This, however, does not make the problem go away! The currently prevailing finite-state approaches (iterative or local) provide little assistance: They are inadequate for even rather simple infinite state problems such as the "counter" example considered below. Here we explore instead a compositional approach. Our aim is to obtain a compositional proof system which is (1) sound, (2) practically useful, (3) powerful enough to prove the kinds of infinite state problems we would hope to be able to address, and (4) complete for the finite state fragment. For (1) and (4) we have positive answers. More work is needed to answer (2) and (3).

Compositionality is addressed by taking a more general view of model checking. Instead of focusing on closed assertions like $\models P : \phi$ we look at sequents of the form $x_1 : \phi_1, ..., x_n : \phi_n \models P(x_1, ..., x_n) : \phi$. That is, properties of the open process term $P(x_1, ..., x_n)$ are relativised to properties of its free variables $x_1, ..., x_n$. This provides a more general proof-theoretical setting which can be exploited to give a structural account of recursive properties. This is a fairly easy task for those connectives like $\wedge$, $\vee$, or the modal operators, that depend only on "local" behaviour. For the fixed point operators the problem is much more difficult. Here we offer an approach based on loop detection. To guide us towards a general solution we offer in this paper a formal proof to show that the CCS process $\mathit{Counter} = \mathit{up}.(\mathit{Counter} \mid \mathit{down}.\mathbf{0})$ after any sequence of consecutive *up* transitions can only perform a finite sequence of consecutive *down* transitions.

An important feature of our approach is that, in contrast to other existing compositional accounts, the sequent style proof system we obtain is constructed from the operational semantics in quite a general and systematic manner. The proof system contains four separate elements: Structural rules, including a cut-rule, to account for sequent structure; logical rules that deal with boolean connectives and recursive formulas; dynamical rules that deal with the modal operators; and finally a single rule of discharge that is responsible for detecting "safe" recurrences of sequents. Only the dynamical rules are dependent upon the specific process algebra under consideration. Moreover the dynamical rules are constructed in a way that one can easily foresee being automated for a range of process algebras.

2 CCS and the Modal μ-Calculus

Our use of CCS follows (Milner, 1989) closely. An *action*, α, is either the invisible action τ or a label l. A *label* is either a (port- or channel-) *name* a, b, say, or a *co-name* $\overline{a}$, $\overline{b}$. Generally $\overline{\overline{a}}$ and a are identified. We assume that the set of labels is finite and ranged over by $l_0, \ldots, l_m$. Sets of labels are ranged over by L, K. *Agent* *expressions*, E, F, are given as follows:

$$E ::= \mathbf{0} \bigm| \alpha.E \bigm| E+E \bigm| E \mid E \bigm| E \setminus L \bigm| x \bigm| \mathrm{fix}x.E$$

where x (and y) range over agent variables. An agent expression is an *agent* if it contains no free agent variables. Agents are ranged over by P, Q, and $\mathcal{A}$ is the set of all agents. The CCS renaming operator is omitted since it adds little of interest to the present account. We refer to (Milner, 1989) for the operational semantics rules.

The syntax of the modal μ-calculus is augmented by equality and inequality of actions which are useful (though not required), primarily to give a reasonable account of the τ-indexed box operator. We consider formulas given in positive form, generated by the grammar

$$\phi ::= \alpha = \beta \bigm| \alpha \neq \beta \bigm| \phi \wedge \phi \bigm| \phi \vee \phi \bigm| [\alpha]\phi \bigm| \langle\alpha\rangle\phi \bigm| X \bigm| \nu X.\phi \bigm| \mu X.\phi$$

where X (Y, Z) ranges over propositional variables. A De-Morganised negation can be defined in this language, by clauses like $\neg[\alpha]\phi = \langle\alpha\rangle\neg\phi$, $\neg\nu X.\phi = \mu X.\neg\phi[\neg X/X]$, $\neg\mu X.\phi = \nu X.\neg\phi[\neg X/X]$, and $\neg\neg X = X$. If α is a name a we can introduce a universal quantifier over actions by abbreviation:

$$\forall\alpha.\phi = \bigwedge\{\phi[\alpha'/\alpha] \mid \alpha' \text{ an action}\}$$

where it is required that ϕ does not have free occurrences of the action $\overline{a}$ so that action terms like $\overline{\tau}$ are avoided.

The semantics of formulas, $||\phi||\mathcal{V} \subseteq \mathcal{A}$, where $\mathcal{V}$ is a valuation assigning sets of agents to propositional variables is quite standard:

$$||\alpha = \beta||\mathcal{V} = \begin{cases} \mathcal{A} & \text{if } \alpha = \beta \\ \emptyset & \text{otherwise} \end{cases}$$

$$||\alpha \neq \beta||\mathcal{V} = \begin{cases} \emptyset & \text{if } \alpha = \beta \\ \mathcal{A} & \text{otherwise} \end{cases}$$

$$||\phi \wedge \psi||\mathcal{V} = ||\phi||\mathcal{V} \cap ||\psi||\mathcal{V}$$

$$||\phi \vee \psi||\mathcal{V} = ||\phi||\mathcal{V} \cup ||\psi||\mathcal{V}$$

$$||[\alpha]\phi||\mathcal{V} = \{P \mid \forall P'. \text{ if } P \xrightarrow{\alpha} P' \text{ then } P' \in ||\phi||\mathcal{V}\}$$

$$||\langle\alpha\rangle\phi||\mathcal{V} = \{P \mid \exists P'.P \xrightarrow{\alpha} P', P' \in ||\phi||\mathcal{V}\}$$

$$||X||\mathcal{V} = \mathcal{V}(X)$$

$$||\nu X.\phi||\mathcal{V} = \bigcup\{A \mid A \subseteq ||\phi||\mathcal{V}[X \mapsto A]\}.$$

$$||\mu X.\phi||\mathcal{V} = \bigcap\{A \mid ||\phi||\mathcal{V}[X \mapsto A] \subseteq A\}.$$

Instead of $P \in ||\phi||\mathcal{V}$ we sometimes write $\models_{\mathcal{V}} P : \phi$, or $\models P : \phi$ if ϕ is closed.

3 Sequents

The basic judgment of the proof system is the sequent.

Definition 3.1 (Sequents, declarations) A *sequent* is an expression of the form $\Gamma \vdash E : \phi$ where Γ is a sequence of *declarations* of one of the forms $x : \phi$ or $X = \phi$.

Sequents are ranged over by s. Declarations of the form $X = \phi$ are called *namings*, and if s contains the naming $X = \phi$ then X is said to *name* ϕ in s. An occurrence of a variable X to the left of the equality sign in a naming $X = \phi$ is regarded as binding. Namings are used as constants in (Stirling and Walker, 1991), and serve to keep track of the unfoldings of fixed point formula occurrences in the proof system. We use σ as a meta-variable over $\{\nu, \mu\}$. If X names a formula of the form $\sigma Y.\psi$ in s then X is called a *σ-variable.*

Declaration sequences and sequents are subject to an inductively defined *well-formedness constraint* in order to ensure that (proposition and process) variables are properly declared. This condition states that variables can be declared at most once, and that for a sequent $\Gamma \vdash E : \phi$, if a variable occurs freely in E or in ϕ then it is declared in Γ, and, for a sequent $\Gamma_1, x : \phi', \Gamma_2 \vdash E : \phi$, if a variable occurs freely in ϕ' then it is declared in Γ_1. Henceforth attention is restricted to sequents that are well-formed. If $s = \Gamma \vdash E : \phi$ then ϕ is called the *conclusion formula* of s, and, if the process variable x is declared in Γ, the formula ϕ associated to x by the declaration in Γ is called the *assumption on x*.

Definition 3.2 (Sequent semantics)

1. The sequent $\vdash P : \phi$ is $\mathcal{V}$-true if and only if $P \in \|\phi\|\mathcal{V}$
2. The sequent $\Gamma, x : \phi \vdash F : \psi$ is $\mathcal{V}$-true if and only if for all agent expressions E, if $\Gamma \vdash E : \phi$ is $\mathcal{V}$-true then so is $\Gamma \vdash F[E/x] : \psi$.
3. The sequent $\Gamma, X = \phi \vdash E : \psi$ is $\mathcal{V}$-true if and only if $\Gamma \vdash E : \psi$ is $\mathcal{V}[X \mapsto \|\phi\|\mathcal{V}]$-true.

If the sequent s is well-formed then the $\mathcal{V}$-truthhood of s is well-defined and independent of $\mathcal{V}$. Notice that the quantification over agent expressions in def. 3.2.3 could equivalently be replaced by a quantification over agents.

4 Local Rules

We are now in a position to present the proof system. It consists of two subsystems, a *local* and a *global* one. We first introduce the local subsystem. The local subsystem is subdivided into three groups of rules: *Structural rules* governing the use of declarations, *logical rules* responsible for the left and right introduction of logical operators, and finally *dynamical rules* for the modal operators which depend on process structure.

Structural rules:

$$\textsc{Declaration}\quad \frac{\cdot}{\Gamma_1, x : \phi, \Gamma_2 \vdash x : \phi}$$

$$\textsc{Cut}\quad \frac{\Gamma_1, \Gamma_2 \vdash E : \phi \quad \Gamma_1, x : \phi, \Gamma_2 \vdash F : \psi}{\Gamma_1, \Gamma_2 \vdash F[E/x] : \psi}$$

Logical rules:

$$=\text{-}\textsc{Right}\quad \frac{\cdot}{\Gamma \vdash E : \alpha = \alpha} \qquad =\text{-}\textsc{Left}\quad \frac{\cdot}{\Gamma_1, x : \alpha = \beta, \Gamma_2 \vdash E : \psi} \quad (\alpha \neq \beta)$$

$$\neq\text{-}\textsc{Right}\quad \frac{\cdot}{\Gamma \vdash E : \alpha \neq \beta} \quad (\alpha \neq \beta) \qquad \neq\text{-}\textsc{Left}\quad \frac{\cdot}{\Gamma_1, x : \alpha \neq \alpha, \Gamma_2 \vdash E : \psi}$$

$$\wedge\text{-}\textsc{Right}\quad \frac{\Gamma \vdash E : \phi \quad \Gamma \vdash E : \psi}{\Gamma \vdash E : \phi \wedge \psi} \qquad \wedge\text{-}\textsc{Left}\quad \frac{\Gamma_1, x : \phi, \Gamma_2 \vdash E : \gamma}{\Gamma_1, x : \phi \wedge \psi, \Gamma_2 \vdash E : \gamma}$$

$$\vee\text{-}\textsc{Right}\quad \frac{\Gamma \vdash E : \phi}{\Gamma \vdash E : \phi \vee \psi} \qquad \vee\text{-}\textsc{Left}\quad \frac{\Gamma_1, x : \phi, \Gamma_2 \vdash E : \gamma \quad \Gamma_1, x : \psi, \Gamma_2 \vdash E : \gamma}{\Gamma_1, x : \phi \vee \psi, \Gamma_2 \vdash E : \gamma}$$

$$\sigma\text{-}\textsc{Right}\quad \frac{\Gamma, Y = \sigma X.\phi \vdash E : Y}{\Gamma \vdash E : \sigma X.\phi} \qquad \sigma\text{-}\textsc{Left}\quad \frac{\Gamma_1, Y = \sigma X.\phi, x : Y, \Gamma_2 \vdash E : \psi}{\Gamma_1, x : \sigma X.\phi, \Gamma_2 \vdash E : \psi}$$

$$Y\text{-}\textsc{Right}\quad \frac{\Gamma_1, Y = \sigma X.\phi, \Gamma_2 \vdash E : \phi[Y/X]}{\Gamma_1, Y = \sigma X.\phi, \Gamma_2 \vdash E : Y}$$

$$Y\text{-}\textsc{Left}\quad \frac{\Gamma_1, Y = \sigma X.\phi, \Gamma_2, x : \phi[Y/X], \Gamma_3 \vdash E : \psi}{\Gamma_1, Y = \sigma X.\phi, \Gamma_2, x : Y, \Gamma_3 \vdash E : \psi}$$

Dynamical rules:

$$\mathbf{0}\text{-}\Box\quad \frac{}{\Gamma \vdash \mathbf{0} : [\alpha]\phi} \qquad \alpha.\text{-}\Diamond\quad \frac{\Gamma \vdash x : \psi}{\Gamma \vdash \alpha.x : \langle\alpha\rangle\psi}$$

$$\alpha.\text{-}\Box\text{-}1\quad \frac{\Gamma \vdash x : \psi}{\Gamma \vdash \alpha.x : [\alpha]\psi} \qquad \alpha.\text{-}\Box\text{-}2\quad \frac{\cdot}{\Gamma \vdash \alpha.E : [\beta]\phi} \quad (\alpha \neq \beta)$$

$$+\text{-}\Diamond\quad \frac{\Gamma_1, x : \phi, \Gamma_2 \vdash x : \psi}{\Gamma_1, x : \langle\alpha\rangle\phi, \Gamma_2 \vdash x + F : \langle\alpha\rangle\psi}$$

$$+\text{-}\Box\quad \frac{\Gamma_1, x : \phi_1, \Gamma_2, \Gamma_3 \vdash x : \psi \quad \Gamma_1, \Gamma_2, y : \phi_2, \Gamma_3 \vdash y : \psi}{\Gamma_1, x : [\alpha]\phi_1, \Gamma_2, y : [\alpha]\phi_2, \Gamma_3 \vdash x + y : [\alpha]\psi}$$

$$|\text{-}\langle\alpha\rangle\quad \frac{\Gamma_1, x : \phi, \Gamma_2, y : \psi, \Gamma_3 \vdash x \mid y : \gamma}{\Gamma_1, x : \langle\alpha\rangle\phi, \Gamma_2, y : \psi, \Gamma_3 \vdash x \mid y : \langle\alpha\rangle\gamma}$$

$$|\text{-}\langle\tau\rangle\quad \frac{\Gamma_1, x : \phi, \Gamma_2, y : \psi, \Gamma_3 \vdash x \mid y : \gamma}{\Gamma_1, x : \langle l\rangle\phi, \Gamma_2, y : \langle\bar{l}\rangle\psi, \Gamma_3 \vdash x \mid y : \langle\tau\rangle\gamma}$$

$$|\text{-}[\alpha]\quad \frac{\begin{array}{c}\Gamma_1, x : \phi_1, \Gamma_2, y : \psi_1, \Gamma_3 \vdash x : [\alpha]\phi_2 \\ \Gamma_1, x : \phi_1, \Gamma_2, y : \psi_1, \Gamma_3 \vdash y : [\alpha]\psi_2 \\ \Gamma_1, x : \phi_1, \Gamma_2, y : \psi_2, \Gamma_3 \vdash x \mid y : \gamma \\ \Gamma_1, x : \phi_2, \Gamma_2, y : \psi_1, \Gamma_3 \vdash x \mid y : \gamma\end{array}}{\Gamma_1, x : \phi_1, \Gamma_2, y : \psi_1, \Gamma_3 \vdash x \mid y : [\alpha]\gamma}\quad (\alpha \neq \tau)$$

$$\frac{\begin{array}{c}\Gamma_1, x : \phi_1, \Gamma_2, y : \psi_1, \Gamma_3 \vdash x : \forall\alpha.[\alpha]\phi_2(\alpha) \\ \Gamma_1, x : \phi_1, \Gamma_2, y : \psi_1, \Gamma_3 \vdash y : \forall\beta.[\beta]\psi_2(\beta) \\ \Gamma_1, x : \phi_2(\tau), \Gamma_2, y : \psi_1, \Gamma_3 \vdash x \mid y : \gamma \\ \Gamma_1, x : \phi_1, \Gamma_2, y : \psi_2(\tau), \Gamma_3 \vdash x \mid y : \gamma \\ \Gamma_1, x : \phi_2(l_0), \Gamma_2, y : \psi_2(\overline{l_0}), \Gamma_3 \vdash x \mid y : \gamma \\ \vdots \\ \Gamma_1, x : \phi_2(l_m), \Gamma_2, y : \psi_2(\overline{l_m}), \Gamma_3 \vdash x \mid y : \gamma\end{array}}{\Gamma_1, x : \phi_1, \Gamma_2, y : \psi_1, \Gamma_3 \vdash x \mid y : [\tau]\gamma}$$

$|\text{-}[\tau]$

$$\backslash\text{-}\Box\text{-}1\quad \frac{\cdot}{\Gamma \vdash E \setminus K : [\alpha]\psi}\quad (\alpha \in K) \qquad \backslash\text{-}\Box\text{-}2\quad \frac{\Gamma_1, x : \phi, \Gamma_2 \vdash x \setminus K : \psi}{\Gamma_1, x : [\alpha]\phi, \Gamma_2 \vdash x \setminus K : [\alpha]\psi}$$

$$\backslash\text{-}\Diamond\quad \frac{\Gamma_1, x : \phi, \Gamma_2 \vdash x \setminus K : \psi}{\Gamma_1, x : \langle\alpha\rangle\phi, \Gamma_2 \vdash x \setminus K : \langle\alpha\rangle\psi}\quad (\alpha \notin K) \qquad \text{FIX}\quad \frac{\Gamma \vdash E[\text{fix}x.E/x] : \phi}{\Gamma \vdash \text{fix}x.E : \phi}$$

The first two sets of rules require little comment, coming, as they do, straight from proof theory. The only noteworthy points are the use of variables to name fixed point formulas, and that symmetric versions of the $\wedge$-LEFT and $\vee$-RIGHT rules have been omitted. Similarly, symmetric versions of the $+$-$\Diamond$, $|$-$\Diamond$, and rules derived from $+$-$\Box$ and the rules for parallel composition, obtained by systematically exchanging the declarations for x and for y, have been omitted.

The rationale behind the dynamical rules is best explained through a little example. Suppose we wish to prove $\vdash P \mid Q : \langle\alpha\rangle\phi$, because we suspect that (1) $P \xrightarrow{\alpha} P'$ and (2) $\models P' \mid Q : \phi$. Our task is to

1. guess a property ϕ_1 of P' and a property ϕ_2 of Q,
2. prove $\vdash P : \langle\alpha\rangle\phi_1$ and $\vdash Q : \phi_2$,
3. prove $x_1 : \phi_1, x_2 : \phi_2 \vdash x_1 \mid x_2 : \phi$, and finally,
4. put (2) and (3) together using $|$-$\langle\alpha\rangle$ and two cuts to conclude $\vdash P \mid Q : \langle\alpha\rangle\phi$.

Comparing with local model checking systems such as Stirling and Walker's (Stirling and Walker, 1991) this account has sacrificed a subformula property ($\vdash P \mid Q : \langle\alpha\rangle\phi$ is proved in terms of processes having the property ϕ) in favour of a subprocess property ($\vdash P \mid Q : \langle\alpha\rangle\phi$ is proved in terms of properties holding of the processes P and Q). We regard this as quite natural and reflecting closely the *compositional* nature of the proof system. We do not expect that any of the tasks (1)–(3) can be automated, although this is possible in special cases, in particular for the case of finite state processes considered later.

5 An Example Proof

In this section we give an example proof to show the local rules at work, and to serve as a setting for discussing termination conditions. The example proves that the infinite state process $\mathit{Counter} = \mathrm{fix} x.\mathit{up}.(x \mid \mathit{down}.\mathbf{0})$ satisfies the property $\phi = \nu X.(\mu Y.[\mathit{down}]Y) \wedge [\mathit{up}]X$, i.e. after any finite consecutive sequence of *up*'s only a finite number of consecutive *down*'s are possible. We use a goal-directed approach. Thus the initial goal is $\vdash \mathit{Counter} : \phi$. We first name ϕ, obtaining the sequent

$$U{=}\phi \vdash \mathit{Counter} : U. \tag{1}$$

We then unfold *Counter* and U, apply $\wedge$-Right, and arrive at the two subgoals

$$U{=}\phi \vdash \mathit{up}.(\mathit{Counter} \mid \mathit{down}.\mathbf{0}) : (\mu Y.[\mathit{down}]Y) \tag{2}$$

$$U{=}\phi \vdash \mathit{up}.(\mathit{Counter} \mid \mathit{down}.\mathbf{0}) : [\mathit{up}]U. \tag{3}$$

Subgoal (2) is easily handled by naming the μ-formula, unfolding, and then applying $\alpha.$-$\Box$-2, so we proceed refining subgoal (3). First using $\alpha.$-$\Box$-1 we obtain

$$U{=}\phi \vdash \mathit{Counter} \mid \mathit{down}.\mathbf{0} : U \tag{4}$$

Now the idea is to use two applications of Cut to refine 4 to three subgoals of the form

$$U{=}\phi \vdash \mathit{Counter} : \phi' \tag{5}$$

$$U{=}\phi, x : \phi' \vdash \mathit{down}.\mathbf{0} : \psi \tag{6}$$

$$U{=}\phi, x : \phi', y : \psi \vdash x \mid y : U. \tag{7}$$

The problem is how to arrive at good choices for ϕ' and ψ. This can be quite tricky. The problem is to specify the behaviour of each of two components *Counter* and $\mathit{down}.\mathbf{0}$ to such detail that all necessary aspects of their possible interactions can be analysed. For *Counter*, keeping (1) in mind, a first guess would be to choose $\phi' = U$. For $\mathit{down}.\mathbf{0}$ we choose the formula $\psi = [\mathit{down}][\mathit{down}]\mathit{ff} \wedge [\mathit{up}]\mathit{ff}$. Another possible choice would have been $\psi = U$. This would have been useful, for instance, for dealing with the related example $\mathit{Counter}' = \mathrm{fix} x.\mathit{up}.(x \mid \mathit{down}.x)$.

Proceeding with the proof, (6) is now eliminated by $\wedge$-Right, $\alpha.$-$\Box$-1, $\alpha.$-$\Box$-2, and $\mathbf{0}$-$\Box$. For (5) our intention is to terminate because (5) has previously been expanded as (1), and U is a ν-variable so termination is safe. Indeed the termination conditions will allow this. Thus, (7) is all that remains. Now, by unfolding and $\wedge$-Right we obtain the subgoals

$$U{=}\phi, x : U, y : \psi \vdash x \mid y : \mu Y.[\mathit{down}]Y \tag{8}$$

$$U{=}\phi, x : U, y : \psi \vdash x \mid y : [\mathit{up}]U. \tag{9}$$

We delay consideration of (8) and concentrate on (9). First unfold the left hand occurrence of U to obtain

$$U{=}\phi, x : \mu Y.[\mathit{down}]Y \wedge [\mathit{up}]U, y : \psi \vdash x \mid y : [\mathit{up}]U. \tag{10}$$

Now the rule |-$[\alpha]$ applies to reduce to the four subgoals

$$U{=}\phi, x : \mu Y.[down]Y \wedge [up]U, y : \psi \vdash x : [up]U \tag{11}$$
$$U{=}\phi, x : \mu Y.[down]Y \wedge [up]U, y : \psi \vdash y : [up]\mathit{ff} \tag{12}$$
$$U{=}\phi, x : U, y : \psi \vdash x \mid y : U \tag{13}$$
$$U{=}\phi, x : \mu Y.[down]Y \wedge [up]U, y : \mathit{ff} \vdash x \mid y : U \tag{14}$$

Of these, (11) and (12) are easily proved using $\wedge$-L and some simple boolean reasoning. (14) is proved using $\neq$-LEFT. Finally, (13) is discharged using (7) since U is a ν-variable (!). We then need to consider (8). First name the right hand μ-formula, letting $\gamma = \mu Y.[down]Y$, then unfold the left-hand occurrence of U, and after an application of $\wedge$-LEFT we obtain

$$U{=}\phi, x : \gamma, y : \psi, V{=}\gamma \vdash x \mid y : V.$$

Now the left hand μ-formula is named too:

$$U{=}\phi, W{=}\gamma, x : W, y : \psi, V{=}\gamma \vdash x \mid y : V \tag{15}$$

and V is unfolded:

$$U{=}\phi, W{=}\gamma, x : W, y : \psi, V{=}\gamma \vdash x \mid y : [down]V. \tag{16}$$

Unfolding W to the left gives

$$U{=}\phi, W{=}\gamma, x : [down]W, y : \psi, V{=}\gamma \vdash x \mid y : [down]V. \tag{17}$$

which reduces through |-$[\alpha]$ and some logical reasoning to the two subgoals

$$U{=}\phi, W{=}\gamma, x : W, y : \psi, V{=}\gamma \vdash x \mid y : V \tag{18}$$
$$U{=}\phi, W{=}\gamma, x : [down]W, y : [down]\mathit{ff}, V{=}\gamma \vdash x \mid y : V. \tag{19}$$

We now arrive at a key point in the proof where we discharge (18) with reference to subgoal (15), because even though the μ-variable V to the right of the turnstyle has been unfolded from (15) to (18) so has another μ-variable, W, to the left of the turnstyle. Thus, intuitively, if we assume the left hand side to be true this will ensure that in fact W, and hence V, will only be unfolded a finite number of times, and hence termination at the point (18) is safe. Finally the proof is completed, refining (19) by first unfolding V and then using |-$[\alpha]$ in a way very similar to the way (16) was dealt with. This is left as an exercise for the reader.

6 Side-conditions and Global Rules

We proceed to explain the global rules justifying the discharge of hypotheses at steps (5), (13), and (18) in the previous section.

A *basic proof structure* (b.p.s.) $\mathcal{B}$ is a finite proof tree constructed according to the local proof rules. Usual tree terminology such as nodes, paths, leaves applies. Basic proof structures may contain occurrences of *hypotheses*. The global subsystem consists of a single *rule of discharge* that determines which occurrences of hypotheses can be discharged, along the lines suggested in section 5. A *proof*, then, is a basic proof structure for which all occurrences of hypotheses have been discharged.

To arrive at a sound rule of discharge it is necessary to

1. consider the ways formulas are "regenerated" along paths through a basic proof structure, and

2. count the number of unfoldings of ν-variables.

The first problem is familiar from most accounts of local fixed point unfolding in the modal μ-calculus: Streett and Emerson (Streett and Emerson, 1989) and later Cleaveland (Cleaveland, 1990) uses a subformula condition to keep track of fixed point occurrences; Stirling and Walker (Stirling and Walker, 1991) uses propositional constants for the same purpose; and Winskel (Winskel, 1991) uses what has become known as the "tag-set" approach. For reasons we return to later we have chosen to use the constant-based approach. To understand the second problem let us anticipate the soundness proof a little. We prove soundness by assuming a proof to be given and a sequent in the proof to be false. Let the sequent concerned be of the form $\Gamma \vdash E : X$, X a ν-variable. We can then find a substitution σ validating Γ and making $\sigma(X)$ false when X is annotated by some suitable ordinal. By applications of Cut this annotation may cause occurrences of X to the left of the turnstyle in some "later" sequent $\Gamma' \vdash E' : \phi'$ to be annotated too. Unfolding specific occurrences of X may cause the annotation of that occurrence to be decreased. We need to arrive at a contradiction even when $\Gamma' \vdash E' : \phi'$ is the conclusion of a nullary rule, say Declaration. But if the annotation of a X to the left is *less than* the annotation of X to the right then there is no guarantee of a contradiction, and soundness may fail.

6.1 Generation and Activity

We first need some machinery to reflect the way formula occurrences give rise to other formula occurrences as proofs are constructed.

Definition 6.1 (Generation) Let a basic proof structure $\mathcal{B}$ and a sequent s_1 in $\mathcal{B}$ be given. Let $\Pi = s_1, \ldots, s_k$ be a path downwards (towards the root) from s_1 to some other sequent s_k in $\mathcal{B}$. Whenever formula occurrences ϕ_1 in s_1 and ϕ_k in s_k exists such that either ϕ_1 and ϕ_k are both conclusion formulas, or ϕ_1 and ϕ_k are both assumptions on the same x then ϕ_k is said to *generate* ϕ_1 *along* Π.

The term "generates" is chosen since we envisage proofs to be constructed in a bottom up fashion from goal to subgoals. Since weakening is not allowed the local check (that both formulas are either the conclusion or assumptions on the same x) suffices. Otherwise an analysis of the entire path instead of just the end-points would be required. The notion of generation is important since it respects activity of variables in a sense which we go on to explain.

Definition 6.2 (Activity) Let $s = \Gamma \vdash E : \phi$ be a well-formed sequent and let ϕ be any formula. A variable X is said to be *active in* ϕ (with respect to s) if either X is free in ϕ or else some Y is free in ϕ, Y names some ψ in s, and X is active in ψ.

Note that well-formedness ensures that the "active-in" relation on variables is a partial order. We impose the following side condition on CUT:

Proviso 6.3 Applications of CUT are subject to the condition: For any ν-variable X, if X is active in ϕ then X is also active in ψ.

The following property concerning "preservation of activity" is crucial for soundness:

Proposition 6.4 *In a b.p.s. $\mathcal{B}$ let a path downwards from s to s' be given. Let ϕ (ϕ') be an occurrence of a formula in s (s'). If ϕ' generates ϕ, X is declared in s', and X is active in ϕ then X is active in ϕ'.* $\square$

PROOF: Induction in length of path $\Pi = s_1, \ldots, s_k$ where $s_1 = s$ and $s_k = s'$. By inspection of the proof system we realise that there are only three cases where X could possibly lose activity. First the case where s_k is the left antecedent of s_{k-1} through a cut is dealt with by our proviso. The other cases apply when X is introduced at s_k through an application of Y-RIGHT or Y-LEFT. But then X is not declared in s_{k-1}, a contradiction. $\square$

6.2 Indexing

We now turn to the counting of unfoldings. An *indexing* is a partial assignment of indices $n \in \omega$ to occurrences of names such that for any sequent $\Gamma \vdash E : \phi$, if two occurrences of X in the same formula in Γ or ϕ are given then one is indexed n iff the other one is. Only the rules σ-LEFT, σ-RIGHT, Y-RIGHT, Y-LEFT, DECLARATION, and CUT are affected by indexing. The modifications needed are the following:

1. σ-LEFT and σ-RIGHT: Y is indexed by 0 in both rules.
2. Y-RIGHT and Y-LEFT: The occurrence of Y in the conclusion is indexed by n and occurrences of Y in the antecedent by $n+1$.

3. Declaration: If X is an n-indexed ν-variable in ϕ to the right of the turnstile then the corresponding occurrence of X in ϕ to the left of the turnstile is indexed by some $n' \leq n$.

4. Cut: For any ν-variable X, if X is active in ϕ then all occurrences of X in ϕ or ψ are indexed by the same index.

As we have explained, indexing is important to ensure that ν-variables are not unfolded "faster" to the left of the turnstile than to the right. The following notion helps to ensure that loops do not cause this to be violated:

Definition 6.5 (Standard sequents) Let $\mathrm{FV}(E) = \{x_1, \ldots, x_n\}$ and let

$$s = \Gamma(x_1 : \phi_1, \ldots, x_k : \phi_k) \vdash E : \phi$$

be a sequent. Then s is said to be *standard* if for all ν-variables X, if X is active in ϕ with index n and if X is active in ϕ_i, $1 \leq i \leq k$, with index n' then $n' \leq n$.

6.3 Regeneration and the Rule of Discharge

We can now state the property of regeneration and the rule of discharge.

Definition 6.6 (Regeneration) In a b.p.s. $\mathcal{B}$ let a path Π downwards from s to s' be given. Suppose that

1. ϕ' generates ϕ along Π,

2. ϕ and ϕ' are identical up to indexing of variables,

3. a variable Y is active in ϕ,

4. Y names a fixed point formula $\sigma X.\psi$ at s',

5. ϕ' generates Y along some strict suffix of Π such that Y results from the application of one of the rules Y-Right or Y-Left.

Then ϕ is *σ-regenerated along* Π *(through Y).*

For the rule of discharge we wish, intuitively, to be able to discharge hypotheses $s = \Gamma \vdash E : \phi$ whenever we below s find another sequent s' which is identical up to the variables free in E (and up to indexing), and such that one of the following two properties hold:

1. ϕ is regenerated along the path from s' to s through a ν-variable.

2. Some formula in a declaration of one of the free variables in E is regenerated from s' to s through a μ-variable.

Condition (1) is the discharge condition found in local model checkers such as (Stirling and Walker, 1991) or (Winskel, 1991): It reflects the intuition that ν-variables express invariant properties that must be refuted by exhibiting progress towards an inconsistency. Condition (2), on the other hand, reflects the fact that μ-variables determine eventuality properties, and that the assumption of an eventuality property on some parameter of E provides a progress measure which, due to regeneration, is strictly decreased along the path concerned.

While this intuition is good as a first approximation the need to take indexing into account complicates matters considerably, in particular due to the potential of loop nesting.

Definition 6.7 (Rule of Discharge) Let $\mathrm{FV}(E) = \{x_1, \ldots, x_k\}$ and let

$$s = \Gamma(x_1 : \phi_1, \ldots, x_k : \phi_k) \vdash E : \phi$$

be an occurrence of a hypothesis in a given basic proof structure $\mathcal{B}$. Suppose that s is standard. Then s can be discharged provided that, below s, there is, up to indexing, an occurrence of a standard sequent

$$s' = \Gamma'(x_1 : \phi_1, \ldots, x_k : \phi_k) \vdash E : \phi$$

such that one of the following conditions hold:

1. ϕ is ν-regenerated along the path from s' to s through some X, say. Then it has to be the case that for all $i : 1 \leq i \leq k$, if ϕ_i is ν-regenerated along the path from s' to s through some Y which is active in X then $Y = X$, and if n (n') is the index of X in ϕ in s (s') and if m (m') is the index of Y in ϕ_i in s (s') then $m - m' \leq n - n'$.

2. ϕ is μ-regenerated along the path from s' to s. Then it has to be the case for some $i : 1 \leq i \leq k$, that ϕ_i is μ-regenerated along the path from s' to s too. Moreover, for all $i : 1 \leq i \leq k$, if ϕ_i is ν-regenerated along the path from s' to s through some Y then Y is not active in ϕ.

7 Soundness

We prove that if $\Gamma \vdash E : \phi$ is provable then it is true. The proof uses ordinal approximations defined in a standard fashion:

$$\|\sigma^0 X.\phi\|\mathcal{V} = \begin{cases} \mathcal{A} & \text{if } \sigma = \nu \\ \emptyset & \text{otherwise} \end{cases}$$

$$\|\sigma^{\kappa+1} X.\phi\|\mathcal{V} = \|\phi\|\mathcal{V}[X \mapsto \|\sigma^\kappa X.\phi\|\mathcal{V}]$$

$$\|\sigma^\lambda X.\phi\|\mathcal{V} = \begin{cases} \bigcap_{\kappa<\lambda} \|\sigma^\kappa X.\phi\|\mathcal{V} & \text{if } \sigma = \nu \\ \bigcup_{\kappa<\lambda} \|\sigma^\kappa X.\phi\|\mathcal{V} & \text{otherwise} \end{cases}$$

Using the well-known fixed point theorem of Knaster-Tarski we see that if $P \in \|\mu X.\phi\|\mathcal{V}$ then $P \in \|\nu^\kappa X.\phi\|\mathcal{V}$ for some ordinal κ, and vice versa, if $P \in \|\nu X.\phi\|\mathcal{V}$ then $P \in \|\nu^\kappa X.\phi\|\mathcal{V}$ for all ordinals κ.

Now, a *partial approximation* ι of a sequent $s = \Gamma \vdash E : \phi$ is a partial annotation of ordinals to names X in s such that if X occurs in ϕ then X is a ν-variable. It is important to keep approximation ordinals and indexing apart. The latter is a pure book-keeping device designed to keep track of the number of times ν-variables are unfolded as one passes upwards in a proof structure. The semantics of formulas is extended slightly to take variable declarations and approximations into account by the clause

$$\|X\|\mathcal{V} = \|\sigma^{(\kappa)}Y.\phi\|\mathcal{V}$$

which should be understood relative to a sequent with left hand side Γ where Γ contains the declaration $X = \sigma Y.\phi$ and where X is annotated by κ.

Definition 7.1 (Truth for substitution and partial approximation) The sequent $\Gamma \vdash E : \phi$ is *true for a substitution* σ of agent variables to agents, *and a partial approximation* ι if $E\sigma \in \|\phi\|$ provided for all x which are free in E, if $x : \phi_x$ is the declaration of x in Γ then $\sigma(x) \in \|\phi_x\|$.

We now embark on the soundness proof proper. We prove that if there is a proof of $s_0 = \Gamma_0 \vdash E_0 : \phi_0$ then it is true. Assume that in fact s_0 is false. Then we find a substitution σ such that s_0 is false for σ and the empty partial approximation ι_0. We trace an infinite sequence of the form $\Pi = (s_0, \sigma_0, \iota_0), (s_1, \sigma_1, \iota_1), \ldots$ such that for all i, s_i is false for σ_i and ι_i, and s_i is (up to indexing) the conclusion of a proof rule instance for which s_{i+1} is an antecedent. By use of approximation ordinals, and using the fact that infinitely many points along Π must correspond to hypotheses that have been discharged we can then arrive at a contradiction.

Suppose the construction has arrived at the sequent $s_i = \Gamma_i \vdash E_i : \phi_i$. The following properties are maintained invariant:

Property 7.2
1. *Let any two occurrences of a free variable X in ϕ_i be given. If one occurrence is annotated by ι_i they both are, and then the annotations are identical. The same holds for any ψ occurring as part of a declaration $x : \psi$ in Γ_i.*

2. *We assume for all ν-variables X that if X is active in both ϕ_i and Γ_i such that X is active in a declaration in Γ_i of a variable which is free in E_i, the active occurrence of X in ϕ_i is indexed by n and approximated by κ, and the active occurrence of X in Γ_i is indexed by n' and approximated by κ', then $\kappa' + n' \geq \kappa + n$.*

To motivate the condition 7.2.2 note that $n - n'$ counts how many more times X to the right of the turnstile has been unfolded than the corresponding occurrence to the left. In some cases, however, unfolding to the left may outpace unfoldings to the right temporarily, violating the invariant. We postpone discussion of this case until we see it arising.

Since ι_0 is empty the invariant 7.2 holds initially. We show how we can identify $(s_{i+1}, \sigma_{i+1}, \iota_{i+1})$ such that s_{i+1} is false for σ_{i+1} and ι_{i+1} by considering each potential rule in turn.

Structural rules. The only circumstance in which DECLARATION could apply is where some ν-variable occurrence to the left of the turnstile is annotated by a smaller approximation ordinal than its corresponding occurrence to the right. However, the invariant condition gives $\kappa' + n' \geq \kappa + n$ and $n' \leq n$ (where κ, κ', n and n' are determined as in 7.2.2), hence $\kappa' \geq \kappa$.

Suppose then that s_i results from an application of the rule CUT. Consider the instance

$$\textsc{Cut}\quad \frac{\Gamma_1, \Gamma_2 \vdash E : \phi \quad \Gamma_1, x : \phi, \Gamma_2 \vdash F : \psi}{\Gamma_1, \Gamma_2 \vdash F[E/x] : \psi}$$

so that $s_i = \Gamma_1, \Gamma_2 \vdash F[E/x] : \psi$. Assume that

(i) $\Gamma_1, \Gamma_2 \vdash E : \phi$ is true for σ_i and ι_i' where ι_i' annotates variables in Γ_1 or Γ_2 as ι_i, and variables in ϕ as the corresponding variables in ψ in s_i.

(ii) $\Gamma_1, x : \phi, \Gamma_2 \vdash F : \psi$ is true for the substitution σ_i' and ι_i'' where

- σ_i' is the substitution for which $\sigma_i'(y) = \sigma_i(y)$ whenever $y \neq x$ and for which $\sigma_i'(x) = E\sigma_i$.
- ι_i'' annotates variable occurrences in Γ_1, Γ_2, and ψ as the corresponding occurrences in s_i, it annotates no occurrences of μ-variables in ϕ, and it annotates ν-variables in ϕ as they are annotated in ψ by ι_i.

From (i) and (ii) it follows that s_i must be true for σ_i and ι_i, hence one of them must fail, and we pick as $(s_{i+1}, \sigma_{i+1}, \iota_{i+1})$ whichever combination that does fail. Note that, due to the side-condition concerning activity for CUT, we ensure that if all ν-variables are annotated in ψ then the same is true for ϕ. Note also that the invariants are maintained true by this construction. Note thirdly that it is this step in the construction that requires ν-variables to be approximated (hence also indexed) both to the right and to the left of the turnstile. This situation does not arise for μ-variables.

Logical rules. Most of the other local rules are quite trivial. The only rules that add something of interest to the construction are those that involve substitutions or approximations. None of the logical rules involve substitutions in a non-trivial way. Approximations are affected only by the rules σ-RIGHT, σ-LEFT, Y-RIGHT, and Y-LEFT. For σ-RIGHT there are two cases: If σ is μ then nothing is changed. If σ is ν then, since s_i is false, we know that we can find some approximation ordinal for which the antecedent of s_i is false when the ordinal annotates the introduced variable. Thus ι_{i+1} is obtained, and clearly the invariants are maintained. A similar argument applies to σ-Left except the roles of μ and ν are now exchanged.

We then arrive at the situations where we need in some cases to temporarily break the invariant 7.2.2. These concern the rules Y-Right and Y-Left. In neither case is σ_i affected. Suppose first that Y is a μ-variable. The rule Y-Right is trivial since it can not be annotated. For Y-Left, if Y is not annotated, start by annotating Y by the least ordinal possible, and proceed. Now, if (the occurrence of) Y is annotated by a successor ordinal then the annotation is decreased by one in obtaining ι_{i+1}. If Y is annotated by a limit ordinal some strictly smaller ordinal is chosen for which the invariant holds, and the construction continues. Suppose then that Y is a ν-variable. Consider first Y-Right. Again, if Y is not annotated, start by annotating Y by the least ordinal possible, and proceed. If Y is annotated by a successor ordinal it is decremented, and if Y is a limit ordinal some strictly smaller ordinal is chosen, for which the invariant continues to hold. Finally consider Y-Left. If Y is not annotated ι_i is left unchanged. So let instead κ' be the annotation of Y. If κ' is a successor ordinal κ' can be decremented by 1 without affecting the invariant. Suppose then that κ' is a limit ordinal λ. If we can find some $\kappa'' < \kappa'$ such that $\kappa'' + n' + 1 \geq \kappa + n$ then we can choose κ'' to annotate Y after the unfolding, preserving the invariant. If on the other hand no such κ'' can be found we have reached the situation where the invariant will have to be (temporarily) broken. We choose some $\kappa'' < \kappa'$ at random and continue the construction even though the "invariant" fails to hold, arguing that we eventually will reach a situation where it can be reinstated. Since no choice of $\kappa'' < \kappa'$ preserves the invariant we can conclude that $\kappa = \lambda + m$ so that $n' = m + n$, ie. $n' \geq n$. That is, we are in the situation where the left hand side, due to applications of Y-Left, has overtaken the right hand side m times when counting numbers of unfoldings of Y for the particular assumption concerned. Assume that the construction of Π has reached stage $j > i$ and that s_j is false for the substitution and partial approximation concerned. Assume first $(*)$ that s_j is either an instance of Declaration, a discharged occurrence of a hypothesis, or a loop sequent—a sequent occurrence which serves as justification for the discharge of a hypothesis. Observe that if $(*)$ fails to hold the construction of Π can be extended by some s_{j+1} preserving the property of being false for the appropriate substitution and partial approximation. We also know that the construction of Π can not go on forever without one of the three situations $(*)$ arising. For all three we know by the side conditions that at that stage $n \geq n'$. Thus, counting the number of unfoldings of Y for the conclusion formula and for the particular assumption formula concerned, we see that, along the path from s_i to s_j, Y must have been unfolded m times more for the conclusion formula than for the assumption formula. Thus we see that, along the path from s_i to s_j, eventually the annotation of Y for the conclusion formula must have decreased strictly below λ to some $\kappa''' < \lambda$. This κ''' serves as the starting point for backtracking on the choice of κ'': If the invariant turns out not to hold at stage j we now forget about

Π from stage $i+1$ onwards and redo the construction, this time with $\kappa'' := \kappa''' + m$. We know that if no new choices are made in the construction of Π, when we arrive at stage j the invariant will have been reinstated. We also know that no matter what path we follow from stage i onwards eventually one of the three conditions $(*)$ will hold. As the number of choices we may have to redo can be bounded we know that eventually we will reach stage j with the invariant having been reinstated.

Dynamical rules. These cause no real complications and are left to the reader.

Global rules. Finally we need to consider the case where s_i is a discharged occurrence of a hypothesis. We then find a sequent s_j, $j \leq i$, which is the loop sequent justifying the discharge of s_i. s_i will have the form $s_i = \Gamma_i(x_1 : \phi_{i,1}, \ldots, x_k : \phi_{i,k}) \vdash E_i : \phi_i$ and s_j will have the form $s_j = \Gamma_j(x_1 : \phi_{i,1}, \ldots, x_k : \phi_{i,k}) \vdash E_i : \phi_i$ where $\mathrm{FV}(E) = \{x_1, \ldots, x_k\}$. We know that the invariant holds for s_j (and that no subsequent backtracking will modify the annotations of s_j). In identifying $(s_{i+1}, \sigma_{i+1}, \iota_{i+1})$ we wish to replace s_i by s_j keeping σ_i and ι_i unchanged. We need to check that the invariant is maintained. Let X be any ν-variable which is active in ϕ_i, indexed n_j in s_j, n_i in s_i, and annotated by, say, κ in s_i. Further, let X also be active in one of the $\phi_{i,j}$, indexed n'_j in s_j, n'_i in s_i, and annotated by κ' in s_i. Since the invariant holds for s_i we know that $\kappa' + n'_i \geq \kappa + n_i$. There are two cases: Either X is the ν-variable through which ϕ_i is regenerated, and then $\kappa' + n'_j \geq \kappa' + n'_i - n_i + n_j \geq \kappa + n_j$ as desired. Otherwise ϕ_i is regenerated through some other ν- or μ-variable. In this case we know that, since X is active in ϕ_i that $n_i = n_j$. Moreover by conditions 6.7.2 and 3 we know that, since X is also active in $\phi_{i,j}$, that $n'_i = n'_j$ too. Hence also here $\kappa' + n'_j \geq \kappa + n_j$ and we have shown the invariant to be maintained. Now $(s_{i+1}, \sigma_{i+1}, \iota_{i+1})$ can be derived since one of the local rules apply.

Completing the proof. Having built the infinite sequence Π we find a σ-variable X which is infinitely often regenerated to the right of the turnstile along Π. If $\sigma = \nu$ a contradiction is obtained since the initial annotation of X is infinitely often decreased along Π. If $\sigma = \mu$ we find a μ-variable which is infinitely often unfolded to the left of the turnstile along Π and a similar argument applies, completing the soundness proof.

8 Completeness for Finite-state Processes

While we view soundness for general processes as the main contribution of the paper, completeness for finite-state processes is important as a check that no proof power has accidentally been sacrificed.

Theorem 8.1 *If P is a well-guarded finite-state process and $\models P : \phi$ then $\vdash P : \phi$ is provable.*

PROOF: (Outline) Theorem 8.1 can be proved by embedding the tableau based model checker of Stirling and Walker (Stirling and Walker, 1991) into the present setting. Consider the proof system obtained by restricting attention to sequents $\Gamma \vdash P : \phi$ where only namings are allowed in Γ, and where the dynamical rules are replaced by the following two global rules:

$$\langle\alpha\rangle\text{-RIGHT} \quad \frac{\Gamma \vdash P' : \phi}{\Gamma \vdash P : \langle\alpha\rangle\phi} \quad (P \xrightarrow{\alpha} P)$$

$$[\alpha]\text{-RIGHT} \quad \frac{\{\Gamma \vdash P' : \phi \mid P \xrightarrow{\alpha} P'\}}{\Gamma \vdash P : [\alpha]\phi}$$

The rule of discharge is modified by allowing a sequent $s = \Gamma \vdash P : X$ to be discharged whenever X is a ν-variable and there is strictly below s another sequent of the form $\Gamma' \vdash P : X$. Call the proof system ensuing from these changes the *Stirling-Walker system*, and write $\Gamma \vdash_{sw} P : \phi$ for provability in this system. By soundness and completeness (Stirling and Walker, 1991) we know that $\models P : \phi$ iff $\vdash_{sw} P : \phi$. So assume that P is finite-state and that a proof π of $\vdash_{sw} P : \phi$ is given. Assume for simplicity that $P = P_1 \mid P_2$. We derive by induction in the size of π formulas ϕ_1 and ϕ_2 such that $\vdash_{sw} P_1 : \phi_1$ and $\vdash_{sw} P_2 : \phi_2$ by proofs of size not greater than the size of π, and $x_1 : \phi_1, x_2 : \phi_2 \vdash x_1 \mid x_2 : \phi$ is provable in the compositional system. Once a similar result has been proved for restriction (which is quite simple), it is an easy induction in the size of proof of $\vdash_{sw} P : \phi$ to show that $\vdash P : \phi$ in the compositional proof system too, establishing the result. As we traverse π from the root upwards we generate pieces of ϕ_1 and ϕ_2 in a manner which respects the structure of π. The greatest difficulty is to deal with names. Assume that the rule Y-RIGHT is applied in π to a sequent, say, $\Gamma \vdash_{sw} P_1' \mid P_2' : X$, and that $\Gamma \vdash_{sw} X = \nu Y.\psi$, so that the resulting antecedent is $\Gamma \vdash_{sw} P_1' \mid P_2' : \psi[X/Y]$. Pick two fresh variables X_1 and X_2. X_1 will be used in ϕ_1 and X_2 in ϕ_2. These variables are generated whenever we reach a sequent occurrence of the form $\Gamma' \vdash P_1' \mid P_2' : X$ strictly above the sequent occurrence $\Gamma \vdash P_1' \mid P_2' : X$. Let ϕ_1', ϕ_2' be the formulas generated by the sequent occurrence $\Gamma \vdash P_1' \mid P_2' : \psi[X/Y]$. Let then $\phi_i = \nu X_i.\phi_i'$, $i \in \{1, 2\}$. By one part of the induction hypothesis, $\vdash_{sw} P_i' : \phi_i'$ can be established from the assumption $\vdash P_i' : X_i$. Thus $\vdash_{sw} P_i' : \phi_i$. From the second part of the induction hypothesis we have a proof in the compositional system of $x_1 : \phi_1', x_2 : \phi_2' \vdash x_1 \mid x_2 : \psi[X/Y]$ from the assumption $x_1 : X_1, x_2 : X_2 \vdash x_1 \mid x_2 : X$, thus obtaining a proof in the compositional system of $x_1 : \phi_1, x_2 : \phi_2 \vdash x_1 \mid x_2 : \nu Y.\psi$ from no assumptions.

This deals with the global part of the construction. For the local part we consider the case where $\Gamma \vdash_{sw} P_1 \mid P_2 : \langle\alpha\rangle\phi$. Suppose, e.g., that $P_1 \xrightarrow{\alpha} P_1'$ and $\Gamma \vdash_{sw} P_1' \mid P_2 : \phi$. By induction we find ϕ_1 and ϕ_2 such that $\vdash_{sw} P_1' : \phi_1$, $\vdash_{sw} P_2 : \phi_2$, and $x_1 : \phi_1, x_2 : \phi_2 \vdash x_1 \mid x_2 : \phi$. But then $\vdash_{sw} P_1 : \langle\alpha\rangle\phi_1$ and $x_1 : \langle\alpha\rangle\phi_1, x_2 : \phi_2 \vdash x_1 \mid x_2 : \langle\alpha\rangle\phi$ as required. □

In fact — since model checking in the Stirling-Walker system is decidable for finite-state processes — the proof of Theorem 8.1 gives an effective strategy for building proofs. Other strategies can be devised, based on e.g. characteristic formulas. Notice also that the proof makes only limited use of the global rules. Termination is needed for greatest fixed points only, and the side-conditions concerning activity and indexing can be eliminated altogether in favour of the much simpler side-condition for Cut that ϕ is a closed formula.

9 Conclusion

A precursor of the present work is (Amadio and Dam, 1995) where a proof system for a process passing calculus is presented, though recursive specifications are not addressed.

The main issues left for future work are analyses of the proof power of the general proof system, and of its practical usefulness. The latter is best evaluated through experimentation. Constructed, as they are, in a systematic way, the local rules may turn out to be quite natural once practice is built up. Moreover, being compositional the proof system is well suited to support macros and derived rules. The quite complicated side-conditions, on the other hand, may seem disconcerting. The hope is that in most practical situations the technicalities concerning indexing and activity can in fact be hidden.

To handle fixed points we have chosen to work with constants in the style of (Stirling and Walker, 1991). Other alternatives would be Winskel's tag-set approach (Winskel, 1991), or subformula conditions in the style of (Streett and Emerson, 1989). In the tag-set approach fixed point formulas are tagged with sufficient information concerning the proof to determine locally whether discharge is possible or not. For local model checking it suffices to record the process terms for which the fixed point has so far been unfolded, admitting a very appealing semantical account. For our case of two-sided sequents the proof information needed to entirely localise discharge decisions would be both very syntactical and very substantial indeed, thus losing much of the appeal of the tag-based approach. Concerning subformula conditions the point is less obvious. In such an approach one would drop namings and use a notion of regeneration that requires a fixed point formula to both be a subformula of the right formula occurrence throughout a given path, and to be actually unfolded somewhere along that path. The proof system would be easily adaptable to such a setting, but a direct adaptation of the side conditions of section 6 would be unsound. For instance one would quite easily be able to prove the sequent

$$x : \nu X_1.\mu Y_1.[a]X_1 \wedge [b]Y_1, y : \nu Y_2.\mu X_2.[a]X_2 \wedge [b]Y_2 \vdash x \mid y : \mu Z.[a]Z \wedge [b]Z \quad (20)$$

where the least fixed point formula corresponding to Y_1 would be unfolded along one path of the proof, and the least fixed point formula corresponding to X_2 along the other, thus justifying discharge according to section 6. However, these unfoldings cancel out when the loops begin to be nested, and indeed the sequent

(20) is easily seen to be false. The problem can be remedied, though, by global side-conditions constraining the way fixed point unfoldings can be allowed to interfere.

The only completeness criterion we have considered here is weak completeness for finite-state processes. Stronger completeness results are needed, maybe along the lines of the so-called well-described formulas explored in (Amadio and Dam, 1995). In this case rules which are not needed for weak completeness such as $\wedge$-$\vee$ distribution ($x : \phi \wedge (\psi \vee \gamma) \vdash x : (\phi \wedge \psi) \vee (\phi \wedge \gamma)$) and monotonicity under the modal operators (e.g. $x : \phi \vdash x : \psi$ implies $x : \langle\alpha\rangle\phi \vdash x : \langle\alpha\rangle\psi$) must be added. For unguarded recursive process terms our proof system is in most cases ineffective. It should be possible to remedy this by adding further reasoning principles along the lines of Hungar's "box recurrences" (Hungar, 1994).

Another issue is to investigate the power of our approach for general, say, GSOS definable languages (cf. Simpson's recent work on Hennessy-Milner logic (Simpson, 1995)). For instance it is quite easy based on the present ideas to develop a sound proof system for context-free processes. Is this proof system completeΓ

Acknowledgements. Thanks are due to Roberto Amadio for numerous discussions on related topics, and to the referees for some useful comments.

References

Abadi, M. and Lamport, L. (1993) Composing Specifications. *ACM Transactions on Programming Languages and Systems* **15**, pp. 73–132.

Abrahamson, K. (1979) Modal logic of concurrent programs. *Lecture Notes in Computer Science* **70**, pp. 21–33.

Amadio, R., and Dam., M. (1995) Reasoning about higher-order processes. Proc. CAAP'95, *Lecture Notes in Computer Science* **915**, pp. 202–216.

Andersen, H., Stirling, C., and Winskel, G. (1994) A compositional proof system for the modal μ-calculus. Proc. LICS'94, pp. 144–153.

Barringer, H., Kuiper, R., and Pnueli, A. (1984) Now you may compose temporal logic specifications. Proc. STOC'84, pp. 51–63.

Bradfield, J. and Stirling, C. (1992) Local model checking for infinite state spaces. *Theoretical Computer Science* **96**, pp. 157–174.

Cleaveland, R. (1990) Tableau-based model checking in the propositional mu-calculus. *Acta Informatica* **27**, pp. 725–747.

Cleaveland, R., Dreimüller, M., and Steffen, B. (1992) Faster model checking for the modal mu-calculus. Proc. CAV'92, *Lecture Notes in Computer Science* **663**, pp. 383–394.

Dam, M. (1994, A) CTL* and ECTL* as Fragments of the Modal Mu-Calculus. *Theoretical Computer Science* **126**, pp. 77–96.

Dam, M. (1994, B) Model checking mobile processes (full version). SICS report RR 94:1. Prel. version appeared in Proc. CONCUR'93, *Lecture Notes in Computer Science* **715**, pp. 22–36.

Emerson, E. A., and Lei, C. (1986) Efficient model checking in fragments of the propositional mu-calculus. Proc. LICS'86, pp. 267–278.

Esparza, J. (1997) Decidability of model checking for infinite-state concurrent systems. *Acta Informatica* **34**, pp. 85–107.

Grumberg, O. and Long, D. E. (1994) Model checking and modular verification. *ACM Transactions on Programming Languages and Systems* **16**, pp. 843–871.

Hungar, H. (1994) Model checking of macro processes. Proc. CAV'94, *Lecture Notes in Computer Science* **818**, pp. 169–181.

Jones, C. (1983) Specification and design of (parallel) programs. Proc. IFIP'83, pp. 321–332.

Jonsson, B. (1994) Compositional Specification and Verification of Distributed Systems. *ACM Transactions on Programming Languages and Systems* **16**, pp. 259–303.

Kozen, D. (1983) Results on the Propositional μ-Calculus. *Theoretical Computer Science* **27**, pp. 333–354.

Larsen, K. G. (1992) Efficient local correctness checking. Proc. CAV'92, *Lecture Notes in Computer Science* **663**.

Larsen, K. G., and Liu, X. (1991) Compositionality through an operational semantics of contexts. *Journal of Logic and Computation* **1**, pp. 761–795.

Milner, R. (1989) *Communication and Concurrency.* Prentice Hall International.

Pnueli, A. (1985) In transition from global to modular temporal reasoning about programs. In: *Logics and Models of Concurrent Systems*, NATO ASI Series F, Computer and System Sciences **13**, pp. 123–144.

Simpson, A. (1995) Compositionality via cut-elimination: Hennessy-Milner logic for an arbitrary GSOS. Proc. LICS'95, pp. 420–430.

Stirling, C. (1988) A generalization of Owicki-Gries's Hoare logic for a concurrent while language. *Theoretical Computer Science* **58**, pp. 347–359.

Stirling, C., and Walker, D. (1991) Local model checking in the modal mu-calculus. *Theoretical Computer Science* **89**, pp. 161–177.

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

Winskel, G. (1991) A note on model checking the modal μ-calculus. *Theoretical Computer Science* **83**, pp. 157–187.