Electronic Notes in Theoretical Computer Science 6 (1997)
URL: `http://www.elsevier.nl/locate/entcs/volume6.html` ?? pages

A Modal μ-Calculus and a Proof System for Value Passing Processes

D.B.Gurov [a] S.Berezin [b] B.M.Kapron [a]

[a] *Department of Computer Science, University of Victoria, Canada*

[b] *School of Computer Science, Carnegie Mellon University, USA*

Abstract

A first-order modal μ-calculus is introduced as a convenient logic for reasoning about processes with value passing. For this logic we present a proof system for model checking sequential processes defined in the value passing CCS. Soundness of the proof system is established. The use of the system is demonstrated on two small but instructive examples.

1 Introduction

The propositional modal μ-calculus is a particularly expressive logic for reasoning about branching-time properties of communicating systems. Many other logics, like dynamic logic and CTL, have uniform encodings in this logic [11]. Over the last decade, many proof systems for checking validity of formulae of this logic w.r.t. particular states (or sets of states) of particular models have been proposed. Since the semantics of the logic is given w.r.t. labelled transition systems (LTS), some of these proof systems [1,4,6,7,15] refer directly to LTS, while other, *compositional* approaches [2,8] refer to descriptions in some process language like CCS [12], whose operational semantics is again defined via LTS. The latter approaches, although being more specialized, are of a high practical importance when full automation is not feasible; compositional proof systems are more intuitive, require less semantic reasoning when applying their proof rules, and are hence easier to be machine assisted.

Some relevant properties of communicating systems with value passing cannot be expressed in the propositional modal μ-calculus. These are properties which depend on the values being communicated. For example, consider process P defined in the value passing CCS:

$$\begin{aligned} P &\triangleq in(x).Q(x) \\ Q(x) &\triangleq \textbf{if } x > 0 \textbf{ then } \overline{out}(x).Q(x-1) \end{aligned}$$

This process has the property (for any integer x) of not being able to engage in an infinite sequence of the form

$$s_0 \xrightarrow{in(x)} s_1 \xrightarrow{\overline{out}(x)} s_2 \xrightarrow{\overline{out}(x-1)} s_3 \xrightarrow{\overline{out}(x-2)} s_4 \xrightarrow{\overline{out}(x-3)} \cdots$$

If the process and the property are abstracted from the communicated values the property does not hold anymore. Motivated by such considerations, we present a (first-order) extension of the propositional modal μ-calculus in which such properties are expressible. In this logic one can formalize the above property of process P as Φ defined by:

$$\Phi \stackrel{\Delta}{\equiv} \forall x.\, [in(x)]\, \Pi(x)$$
$$\Pi \stackrel{\Delta}{\equiv} \mu Z.\lambda x.\, [\overline{out}(x)]\, Z(x-1)$$

Here Π is a (recursively defined) one-argument predicate over the domain of values. The logic we propose is partially based on earlier treatments of message passing [10] (a more recent development of which is [13]) and name passing [7]. The syntax we have chosen aims to improve the readability of formulae by distinguishing syntactically between formulae and predicates over the domain of values, which reflects the semantic importance of the latter. The expressive power of such logics depends on the language of Boolean and value expressions chosen. As in standard value passing CCS [12], we do not confine ourselves to a particular such language.

An important problem for such a logic is how to check whether a particular process possesses a certain property. In this paper we present a proof system for model checking processes, defined in the value passing CCS without parallel composition. Following an approach of C.Stirling [14], parallel composition may be handled with the help of an additional proof system. A step in this direction is the system presented in [3].

In designing our proof system the highest challenge and interest is presented by the treatment of fix-point formulae. Different approaches to this problem have been proposed for the propositional case. Some of these employ *tagging* of fix-point formulae to keep inference rules local, which simplifies both their use and theoretical treatment. For greatest fix-points this technique has been proposed by G.Winskel in [17], while H.Andersen [1] extends it to least fix-points by encoding into tags the inductive reasoning required for dealing with such formulae. We generalize these approaches to handle fix-point predicates. One difficulty here is to find what constitute tags and to choose an appropriate semantics for tagged predicates. Another challenge is to generalize the approach of [1] without having to introduce infinitary rules. This is possible due to the semantics of sequents we have chosen. It is accomplished by performing induction on the domain of values rather than explicitly on sets of processes. One has, however, to introduce into the language a new syntactic category, namely *arbitrary constants*.

All other rules have been designed to guarantee that no proof power has been sacrificed. To facilitate machine-assisted proofs, the rules are kept as syntactic as possible by minimizing the semantic reasoning required for their application. This compensates to a great extent for the (for compositional proof systems) inevitably high number of proof rules since it minimizes the need for human intervention when using a proof assistant and delegates more responsibility to the machine. The many examples on which we have tried our proof system so far seem to justify our effort; the proofs are relatively short and follow intuition nicely.

The paper is organized as follows. In Section 2 we present our extension of the modal μ-calculus for value passing systems. We next give a compositional proof system for value passing CCS without parallel composition, establish soundness of the system, and present two small but instructive example proofs. The last section contains some conclusions and directions for further research.

2 A Modal μ-Calculus for Value Passing Systems

In this section we present an extension of the modal μ-calculus for value passing systems. We first present labelled transition systems, then the syntax, and finally the semantics of the logic.

2.1 Labelled Transition Systems

We assume a set $\mathcal{A}$ of *names*, ranged over by a, each name having a non-negative arity. Let $\mathcal{L}$ denote the set $\mathcal{A} \cup \overline{\mathcal{A}}$ of *labels*, ranged over by l, and let $\overline{\overline{a}} \triangleq a$. We also assume a set D of *values*, and variables $x, y, z, \ldots$ (possibly indexed), ranging over D.

Note 1 *We use $\vec{D}, \vec{D}', \ldots$ to denote D^n for some non-negative integer n, being either arbitrary or understood from the context. In the same way, we use $\vec{x}$ to stand for a vector $(x_1, x_2, \ldots, x_n)$ of variables over D.*

Definition 2.1 *A* labelled transition system *over a set $\mathcal{A}$ of names and a set D of values is defined as a triple*

$$\mathcal{T} = (S, Act, \longrightarrow)$$

where:
(i) S is a set of states*;*
(ii) $Act \triangleq \{l(\vec{d}) \mid l \in \mathcal{L}, \vec{d} \in \vec{D}\} \cup \{\tau\}$ is a set of actions*, ranged over by α;*
(iii) $\longrightarrow \subseteq S \times Act \times S$ is a transition relation.

Definition 2.2 *For any labelled transition system $\mathcal{T} = (S, Act, \longrightarrow)$ and any action $\alpha \in Act$ we define the* state transformers

$$[\![\langle\alpha\rangle]\!]^{\mathcal{T}} \triangleq \lambda S' \subseteq S.\, \{s \in S \mid \exists s' \in S'.\, s \xrightarrow{\alpha} s'\}$$
$$[\![[\alpha]]\!]^{\mathcal{T}} \triangleq \lambda S' \subseteq S.\, \{s \in S \mid \forall s' \in S.\, s \xrightarrow{\alpha} s' \textit{ implies } s' \in S'\}$$

2.2 Syntax of the Logic

The logic we introduce uses the following syntactic categories: vectors $\vec{e}$ of value expressions over D, formulae Φ, and predicates Π. Formulae Φ of the logic are generated by the grammar:

$$\begin{array}{lcl} \Phi & ::= & b \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid [\pi]\ \Phi \mid \langle\pi\rangle\,\Phi \mid \forall x.\Phi \mid \exists x.\Phi \mid \Pi\,\vec{e} \\ \Pi & ::= & Z \mid \lambda\vec{x}.\Phi \mid \nu Z.\Pi \mid \mu Z.\Pi \end{array}$$

where Z ranges over a set $Pred$ of predicate variables, x over a set Var of object variables, $\vec{e}$ has the arity of Π, and π are actions generated by:

$$\pi \ ::= \ a(\vec{e}) \mid \overline{a}(\vec{e}) \mid \tau$$

where $a, \overline{a} \in \mathcal{L}$, and the arities of a and $\overline{a}$ respect the arity of $\vec{e}$. Zero-ary predicates are identified with formulae. The exact syntax of Boolean and value expressions is left free in order to allow for a general treatment of the logic; we only assume Boolean expressions of the form $\vec{e}_1 = \vec{e}_2$.

Free and bound variables are defined as usual. Unlike [10,7], we assume an early semantics for input actions and these have no binding power in our logic. We also assume the usual notion $\Phi[\vec{e}/\vec{x}]$ of (capture avoiding) substitution.

2.3 Semantics of the Logic

We start by defining the notion of *model*, or *Kripke structure*, for the logic.

Definition 2.3 *A* model *for the logic over $\mathcal{A}$ and D is a pair*

$$\mathcal{M} = (\mathcal{T}, \mathcal{V})$$

where:
(i) $\mathcal{T} \triangleq (S, Act, \longrightarrow)$ *is a labelled transition system over $\mathcal{A}$ and D;*
(ii) $\mathcal{V} \triangleq (\mathcal{V}_{Var}, \mathcal{V}_{Pred})$ *is a* valuation*, such that*
$\mathcal{V}_{Var} : Var \to D$
$\mathcal{V}_{Pred} : Pred \to [\vec{D} \to \wp(S)]$

We assume the usual notion of updating an environment: for example, the valuation $\mathcal{V}[\vec{d}/\vec{x}]$ agrees with $\mathcal{V}$ on all variables other than the ones in $\vec{x}$ and assigns d_i to x_i. Let $b[\mathcal{V}_{Var}]$ and $e[\mathcal{V}_{Var}]$ denote the substitution of all free variables in b and e according to $\mathcal{V}_{Var}$.

Given a model $\mathcal{M}$, the semantics of a formula Φ is defined inductively by the denotation $\|\Phi\|_{\mathcal{M}}$ of Φ w.r.t. $\mathcal{M}$, i.e. by the set of states in S satisfying Φ. We write $\|\Phi\|_{\mathcal{V}}^{\mathcal{T}}$ for $\|\Phi\|_{\mathcal{M}}$ and often omit the superscript when understood from the context.

The denotation of the different syntactic categories is of the following type: $\|\vec{e}\|_{\mathcal{V}} \in \vec{D}$, $\|\pi\|_{\mathcal{V}} \in Act$, $\|\Phi\|_{\mathcal{V}} \subseteq S$, and $\|\Pi\|_{\mathcal{V}} : \vec{D} \to \wp(S)$. We assume that closed Boolean expressions b have a fixed truth value $[\![b]\!] \in \{\mathsf{tt}, \mathsf{ff}\}$, and closed value expressions e have a fixed value $[\![e]\!] \in D$.

Let $\mathcal{E}_{\vec{D}}$, $\mathcal{E}'_{\vec{D}}$, etc. range over predicates, i.e. over the set $[\vec{D} \to \wp(S)]$ of mappings from $\vec{D}$ to subsets of S, and let $\mathcal{E}_{\vec{d}}$ stand for $\mathcal{E}_{\vec{D}}(\vec{d}\,)$. Let $\mathcal{E}_{\vec{D}} \sqsubseteq \mathcal{E}'_{\vec{D}}$ denote that $\mathcal{E}_{\vec{d}} \subseteq \mathcal{E}'_{\vec{d}}$ for all $\vec{d} \in \vec{D}$. Let $E \subseteq [\vec{D} \to \wp(S)]$. We define

$$\sqcup E \triangleq \lambda \vec{d}.\ \textstyle\bigcup_{\mathcal{E}_{\vec{D}} \in E}\ \mathcal{E}_{\vec{d}} \qquad \sqcap E \triangleq \lambda \vec{d}.\ \textstyle\bigcap_{\mathcal{E}_{\vec{D}} \in E}\ \mathcal{E}_{\vec{d}}$$

$$\|\vec{e}\|_{\mathcal{V}} \triangleq [\![\vec{e}[\mathcal{V}_{Var}]]\!] \quad \|Z\|_{\mathcal{V}} \triangleq \mathcal{V}_{Pred}(Z)$$

$$\|a(\vec{e})\|_{\mathcal{V}} \triangleq a(\|\vec{e}\|_{\mathcal{V}}) \quad \|\overline{a}(\vec{e})\|_{\mathcal{V}} \triangleq \overline{a}(\|\vec{e}\|_{\mathcal{V}}) \quad \|\tau\|_{\mathcal{V}} \triangleq \tau$$

$$\|b\|_{\mathcal{V}} \triangleq \begin{cases} S & \text{if } [\![b[\mathcal{V}_{Var}]]\!] = \mathsf{tt} \\ \{\} & \text{otherwise} \end{cases}$$

$$\|\Phi_1 \wedge \Phi_2\|_{\mathcal{V}} \triangleq \|\Phi_1\|_{\mathcal{V}} \cap \|\Phi_2\|_{\mathcal{V}} \quad \|\Phi_1 \vee \Phi_2\|_{\mathcal{V}} \triangleq \|\Phi_1\|_{\mathcal{V}} \cup \|\Phi_2\|_{\mathcal{V}}$$

$$\|[\pi]\,\Phi\|_{\mathcal{V}} \triangleq [\![[\|\pi\|_{\mathcal{V}}]]\!]\ \|\Phi\|_{\mathcal{V}} \quad \|\langle\pi\rangle\,\Phi\|_{\mathcal{V}} \triangleq [\![\langle\|\pi\|_{\mathcal{V}}\rangle]\!]\ \|\Phi\|_{\mathcal{V}}$$

$$\|\forall x.\Phi\|_{\mathcal{V}} \triangleq \bigcap_{d \in D} \|\Phi\|_{\mathcal{V}[d/x]} \quad \|\exists x.\Phi\|_{\mathcal{V}} \triangleq \bigcup_{d \in D} \|\Phi\|_{\mathcal{V}[d/x]}$$

$$\|\Pi\ \vec{e}\|_{\mathcal{V}} \triangleq \|\Pi\|_{\mathcal{V}}\ \|\vec{e}\|_{\mathcal{V}} \quad \|\lambda\vec{x}.\Phi\|_{\mathcal{V}} \triangleq \lambda\vec{d}.\,\|\Phi\|_{\mathcal{V}[\vec{d}/\vec{x}]}$$

$$\|\nu Z.\Pi\|_{\mathcal{V}} \triangleq \sqcup\{\mathcal{E}_{\vec{D}} \mid \mathcal{E}_{\vec{D}} \sqsubseteq \|\Pi\|_{\mathcal{V}[\mathcal{E}_{\vec{D}}/Z]}\}$$

$$\|\mu Z.\Pi\|_{\mathcal{V}} \triangleq \sqcap\{\mathcal{E}_{\vec{D}} \mid \mathcal{E}_{\vec{D}} \sqsupseteq \|\Pi\|_{\mathcal{V}[\mathcal{E}_{\vec{D}}/Z]}\}$$

Fig. 1. Denotation of formulae.

Definition 2.4 *The* denotation *of formulae Φ of the logic is defined (inductively) as shown in Figure 1.*

Using the well-known Tarski-Knaster fix-point theorem [16] one can show that the semantics of the fix-point predicates is as expected:

$$\|\sigma Z.\Pi\|_{\mathcal{V}} = \sigma\mathcal{E}_{\vec{D}}.\ \|\Pi\|_{\mathcal{V}[\mathcal{E}_{\vec{D}}/Z]}$$

for any predicate Π and any valuation $\mathcal{V}$, where σ stands for either μ or ν. As a consequence, we have for any predicate Π and any valuation $\mathcal{V}$,

$$\|\sigma Z.\Pi\|_{\mathcal{V}} = \|\Pi[\sigma Z.\Pi/Z]\|_{\mathcal{V}}$$

which justifies the denotation chosen for the fix-point predicates and provides a notion of *unfolding*.

The denotation of closed formulae does not depend on valuations; we can hence write $\|\Phi\|$ instead of $\|\Phi\|_{\mathcal{V}}$ when Φ is closed.

3 A Proof System for the Value Passing CCS

In this section we present a compositional proof system for proving properties of sequential processes described in the value passing CCS [12]. First, we define agent expressions, then we present the proof rules of our system, state soundness of the system, and finally, give two example proofs.

3.1 Agent Expressions

Agent expressions E over $\mathcal{A}$ and D of the value passing calculus are generated by the grammar

$$\begin{array}{rcl} E & ::= & \mathbf{0} \mid \pi.E \mid E+E \mid \Sigma\vec{x}\,E \mid E|E \mid E\backslash U \mid E\{\Xi\} \mid \mathbf{if}\, b\, \mathbf{then}\, E \mid A(\vec{e}) \\ \pi & ::= & a(\vec{x}) \mid \overline{a}(\vec{e}) \mid \tau \end{array}$$

Here A are *agent constants*, each having a defining equation

$$A(\vec{x}) \triangleq E$$

where the right-hand side E may contain no free variables except the ones in $\vec{x}$. $U \subseteq \mathcal{L}$ are *restriction* sets, and $\Xi : \mathcal{L} \to \mathcal{L}$ are *relabelling* functions (i.e. functions satisfying $\Xi(\overline{l}) = \overline{\Xi(l)}$ and $\Xi(\tau) = \tau$). The notion of *free variables* in agent expressions is as usual (input actions and infinite summation having binding power), and so is the notion of *closed* agent expressions, which are termed *processes* and are ranged over by $P, Q, \ldots$ For processes, an *operational semantics* is given as usual by a set of transition rules as shown on Figure 2.

Let $\mathcal{V} = (\mathcal{V}_{Var}, \mathcal{V}_{Pred})$ be a valuation. We define the *denotation* $\|E\|_{\mathcal{V}}$ of an agent expression w.r.t. $\mathcal{V}$ as the process which is obtained from E by substituting all free variables according to $\mathcal{V}_{Var}$, i.e.

$$\|E\|_{\mathcal{V}} \triangleq E[\mathcal{V}_{Var}]$$

Let $\mathcal{P}$ range over *transition closed sets* (t.c.s.) of processes (i.e. sets closed under the rules of Figure 2). Given process P, let $\mathcal{P}(P)$ denote the smallest t.c.s. containing P, and let $\mathcal{T}(P) \triangleq (\mathcal{P}(P), Act, \longrightarrow)$ be the corresponding

$$
\mathrm{R}(\tau)\ \frac{\cdot}{\tau.P \xrightarrow{\tau} P} \qquad \mathrm{R}(in)\ \frac{\cdot}{a(\vec{x}).E \xrightarrow{a(\vec{d})} E[\vec{d}/\vec{x}]} \qquad \mathrm{R}(out)\ \frac{\cdot}{\overline{a}(\vec{e}).P \xrightarrow{\overline{a}([\![\vec{e}]\!])} P}
$$

$$
\mathrm{R}(+l)\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'} \qquad \mathrm{R}(+r)\ \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 + P_2 \xrightarrow{\alpha} P_2'}
$$

$$
\mathrm{R}(\Sigma)\ \frac{E[\vec{d}/\vec{x}] \xrightarrow{\alpha} P}{\Sigma\vec{x}\,E \xrightarrow{\alpha} P} \qquad \mathrm{R}(|)\ \frac{P_1 \xrightarrow{l(\vec{d})} P_1' \qquad P_2 \xrightarrow{\overline{l}(\vec{d})} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'}
$$

$$
\mathrm{R}(|l)\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1|P_2 \xrightarrow{\alpha} P_1'|P_2} \qquad \mathrm{R}(|r)\ \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1|P_2 \xrightarrow{\alpha} P_1|P_2'}
$$

$$
\mathrm{R}(\backslash)\ \frac{P \xrightarrow{l(\vec{d})} P'}{P\backslash U \xrightarrow{l(\vec{d})} P'\backslash U}\ l,\overline{l} \notin U \qquad \mathrm{R}(\Xi)\ \frac{P \xrightarrow{l(\vec{d})} P'}{P\{\Xi\} \xrightarrow{l'(\vec{d})} P'\{\Xi\}}\ \Xi(l) = l'
$$

$$
\mathrm{R}(\mathbf{if})\ \frac{P \xrightarrow{\alpha} P'}{\mathbf{if}\ b\ \mathbf{then}\ P \xrightarrow{\alpha} P'}\ [\![b]\!] = \mathtt{tt} \qquad \mathrm{R}(\triangleq)\ \frac{E[\vec{d}/\vec{x}] \xrightarrow{\alpha} P}{A(\vec{d}) \xrightarrow{\alpha} P}\ A(\vec{x}) \triangleq E
$$

Fig. 2. Transition rules for processes.

LTS. Given also a valuation $\mathcal{V}$, the pair $(\mathcal{T}(P), \mathcal{V})$ is a model for our logic. We consider satisfaction with respect to such models.

3.2 Proof Rules

Our proof system is designed for proving satisfaction of a formula w.r.t. some particular agent expression relative to an additional condition on the values of the free object variables occurring in the formula and agent expression. The *judgements* (or *sequents*) of the proof system (following in part [10,7]), have the form $B \vdash E : \Phi$, where B is a finite (possibly empty) sequence $b_1, b_2, \ldots, b_n$ of Boolean expressions over D. The intended semantics of such judgements is given by

$$
B \models E : \Phi \ \stackrel{\Delta}{\equiv}\ \textit{for any}\ \mathcal{V},\ \ \|B\|_{\mathcal{V}} = \mathtt{tt}\ \textit{implies}\ \ \|E\|_{\mathcal{V}} \in \|\Phi\|_{\mathcal{V}}^{\mathcal{T}(\|E\|_{\mathcal{V}})}
$$

where $\|B\|_{\mathcal{V}}$ is defined inductively by (i) $\|\epsilon\|_{\mathcal{V}} = \mathtt{tt}$ holds (ϵ denotes the empty sequence), and (ii) $\|b, B\|_{\mathcal{V}} = \mathtt{tt} \stackrel{\Delta}{\equiv} [\![b[\mathcal{V}_{Var}]]\!] = \mathtt{tt} \wedge \|B\|_{\mathcal{V}} = \mathtt{tt}$. By abuse of notation we extend all Boolean connectives (like $\wedge$, $\vee$ and $\Rightarrow$) over such sequences, with the meaning that the corresponding relation holds for any valuation. Note that a free variable occurring in more that one part of the sequent denotes the same value; symbolic proofs are thus greatly facilitated, at the (inevitable) expense of having more complicated rules.

The proof rules can be grouped according to the parts of a judgement on whose structure they depend. There are also some rules which do not look into the structure of any of these parts; they are referred to as *general rules*. All rules are given in Figure 3; they are presented in a "goal-oriented" fashion. *Axioms* are presented as rules with an empty conclusion (denoted by a dot).

$$\mathrm{E}(\mathbf{if})\ \frac{B\ \vdash\ \mathbf{if}\ b\ \mathbf{then}\ E\ :\ \Phi}{B\ \vdash\ E\ :\ \Phi}\ B \Rightarrow b \quad \mathrm{E}(\triangleq)\ \frac{B\ \vdash\ A(\vec{e})\ :\ \Phi}{B\ \vdash\ E[\vec{e}/\vec{x}]\ :\ \Phi}\ A(\vec{x}) \triangleq E$$

$$\Phi(\wedge)\ \frac{B\ \vdash\ E\ :\ \Phi_1 \wedge \Phi_2}{B\ \vdash\ E\ :\ \Phi_1 \qquad B\ \vdash\ E\ :\ \Phi_2}$$

$$\Phi(\vee l)\ \frac{B\ \vdash\ E\ :\ \Phi_1 \vee \Phi_2}{B\ \vdash\ E\ :\ \Phi_1} \quad \Phi(\vee r)\ \frac{B\ \vdash\ E\ :\ \Phi_1 \vee \Phi_2}{B\ \vdash\ E\ :\ \Phi_2}$$

$$\Phi(\forall)\ \frac{B\ \vdash\ E\ :\ \forall x.\Phi}{B\ \vdash\ E\ :\ \Phi[y/x]}\ y - fresh \quad \Phi(\exists)\ \frac{B\ \vdash\ E\ :\ \exists x.\Phi}{B\ \vdash\ E\ :\ \Phi[e/x]}$$

$$\Phi(b)\ \frac{B\ \vdash\ E\ :\ b}{\cdot}\ B \Rightarrow b \quad \Phi(\beta)\ \frac{B\ \vdash\ E\ :\ (\lambda\vec{x}.\Phi)\vec{e}}{B\ \vdash\ E\ :\ \Phi[\vec{e}/\vec{x}]}$$

$$\Phi(\nu 0)\ \frac{B\ \vdash\ E\ :\ (\nu Z\{L\}.\Pi)\vec{e}}{\cdot}\ \mathcal{C}_{\nu 0} \quad \Phi(\nu 1)\ \frac{B\ \vdash\ E\ :\ (\nu Z\{L\}.\Pi)\vec{e}}{B\ \vdash\ E\ :\ (\Pi[\nu Z\{l,L\}.\Pi/Z])\vec{e}}\ \mathcal{C}_{\nu 1}$$

$$\Phi(\mu 0)\ \frac{B\ \vdash\ E\ :\ (\mu Z\{L\}.\Pi)\vec{e}}{\cdot}\ \mathcal{C}_{\mu 0} \quad \Phi(\mu 1)\ \frac{B\ \vdash\ E\ :\ (\mu Z\{L\}.\Pi)\vec{e}}{B'\ \vdash\ E'\ :\ (\Pi[\mu Z\{l,L\}.\Pi/Z])\vec{e}\,'}\ \mathcal{C}_{\mu 1}$$

$$\mathrm{E}\Phi(\mathbf{0},[\,])\ \frac{B\ \vdash\ \mathbf{0}\ :\ [\pi]\,\Phi}{\cdot} \quad \mathrm{E}\Phi(\pi,[\,])\ \frac{B\ \vdash\ \pi.E\ :\ [\pi']\,\Phi}{\cdot}\ \pi \not\sim \pi'$$

$$\mathrm{E}\Phi(\tau,\langle\rangle)\ \frac{B\ \vdash\ \tau.E\ :\ \langle\tau\rangle\,\Phi}{B\ \vdash\ E\ :\ \Phi} \quad \mathrm{E}\Phi(\tau,[\,])\ \frac{B\ \vdash\ \tau.E\ :\ [\tau]\,\Phi}{B\ \vdash\ E\ :\ \Phi}$$

$$\mathrm{E}\Phi(a,\langle\rangle)\ \frac{B\ \vdash\ a(\vec{x}).E\ :\ \langle a(\vec{e})\rangle\,\Phi}{B\ \vdash\ E[\vec{e}/\vec{x}]\ :\ \Phi} \quad \mathrm{E}\Phi(a,[\,])\ \frac{B\ \vdash\ a(\vec{x}).E\ :\ [a(\vec{e})]\,\Phi}{B\ \vdash\ E[\vec{e}/\vec{x}]\ :\ \Phi}$$

$$\mathrm{E}\Phi(\overline{a},\langle\rangle)\ \frac{B\ \vdash\ \overline{a}(\vec{e}).E\ :\ \langle\overline{a}(\vec{e})\rangle\,\Phi}{B\ \vdash\ E\ :\ \Phi} \quad \mathrm{E}\Phi(\overline{a},[\,])\ \frac{B\ \vdash\ \overline{a}(\vec{e}).E\ :\ [\overline{a}(\vec{e}\,')]\,\Phi}{\vec{e}=\vec{e}\,',B\ \vdash\ E\ :\ \Phi}$$

$$\mathrm{E}\Phi(+l,\langle\rangle)\ \frac{B\ \vdash\ E_1+E_2\ :\ \langle\pi\rangle\,\Phi}{B\ \vdash\ E_1\ :\ \langle\pi\rangle\,\Phi} \quad \mathrm{E}\Phi(+r,\langle\rangle)\ \frac{B\ \vdash\ E_1+E_2\ :\ \langle\pi\rangle\,\Phi}{B\ \vdash\ E_2\ :\ \langle\pi\rangle\,\Phi}$$

$$\mathrm{E}\Phi(+,[\,])\ \frac{B\ \vdash\ E_1+E_2\ :\ [\pi]\,\Phi}{B\ \vdash\ E_1\ :\ [\pi]\,\Phi \qquad B\ \vdash\ E_2\ :\ [\pi]\,\Phi}$$

$$\mathrm{E}\Phi(\Sigma,\langle\rangle)\ \frac{B\ \vdash\ \Sigma\vec{x}\ E\ :\ \langle\pi\rangle\,\Phi}{B\ \vdash\ E[\vec{e}/\vec{x}]\ :\ \langle\pi\rangle\,\Phi} \quad \mathrm{E}\Phi(\Sigma,[\,])\ \frac{B\ \vdash\ \Sigma\vec{x}\ E\ :\ [\pi]\,\Phi}{B\ \vdash\ E[\vec{y}/\vec{x}]\ :\ [\pi]\,\Phi}\ \vec{y} - fresh$$

$$\mathrm{E}\Phi(\mathbf{if},[\,])\ \frac{B\ \vdash\ \mathbf{if}\ b\ \mathbf{then}\ E\ :\ [\pi]\,\Phi}{b,B\ \vdash\ E\ :\ [\pi]\,\Phi}$$

$$\mathrm{G}(\mathsf{ff})\ \frac{B\ \vdash\ E\ :\ \Phi}{\cdot}\ B \equiv \mathsf{ff} \quad \mathrm{G}(\mathsf{US})\ \frac{B[\vec{e}/\vec{x}]\ \vdash\ E[\vec{e}/\vec{x}]\ :\ \Phi[\vec{e}/\vec{x}]}{B\ \vdash\ E\ :\ \Phi}$$

$$\mathrm{G}(\mathsf{Cut})\ \frac{B\ \vdash\ E\ :\ \Phi}{B_1\ \vdash\ E\ :\ \Phi \qquad B_2\ \vdash\ E\ :\ \Phi}\ B \Rightarrow B_1 \vee B_2$$

$$\mathrm{G}(\equiv)\ \frac{B\ \vdash\ E\ :\ \Phi}{B\ \vdash\ E'\ :\ \Phi'}\ E \equiv_B E',\ \Phi \equiv_B \Phi'$$

Fig. 3. Proof Rules.

The rules for restriction and renaming, following the approach advocated in [5], are as in the propositional case and can therefore be omitted here for brevity. In $\Phi(\forall)$, y-fresh means that y does not occur free in B, E or $\forall x.\Phi$. In rule $\mathrm{E}\Phi(\pi, [\,])$, $\pi \not\sim \pi'$ means that π and π' are not *compatible*: either exactly one of them is τ, or otherwise if $\pi = l(\vec{e})$ and $\pi' = l'(\vec{e}\,')$, then $l \neq l'$. Rule $\mathrm{G}(\mathsf{US})$ is a rule of *universal substitution*, corresponding to the *widening* rule(s) in other systems. In rule $\mathrm{G}(\equiv)$, the side condition requires E to be identical to E' up to equivalence of terms under B, e.g. $E(x^2) \equiv_{x=1} E(x)$, and the same for Φ and Φ'.

To allow for a simpler treatment of fix-point formulae, we impose the restriction on the syntax of formulae that all fix-point sub-predicates of the root formula are closed. Note that all rules preserve this property. This restriction does not affect the expressive power of the logic, since every formula containing fix-point sub-predicates with free object variables can easily be converted into an equivalent formula in which all fix-point sub-predicates are closed. For example, $\sigma Z. \langle a(x)\rangle\, Z$ is equivalent to $(\sigma Z.\lambda x.\, \langle a(x)\rangle\, Z(x))(x)$.

For dealing with fix-point formulae we use *tags*, generalizing the approaches of [1,2,17]. The idea of using tags for fix-point formulae is to allow for the detection of loops of a certain kind when traversing the (symbolic) LTS of the process. Detecting such a loop would guarantee the validity of the corresponding sequent. In our case, a tag is a finite (possibly empty) list $L = l_1, l_2, \ldots, l_n$, where each l_i is a triple $(B_i, E_i, \vec{e}_i)$. With a triple $l = (B, E, \vec{e})$ we associate the indexed set of processes

$$\mathcal{E}^l_{\vec{D}} \triangleq \lambda\vec{d}.\{P \mid \exists\mathcal{V}.(\|B\|_{\mathcal{V}} = \mathsf{tt} \;\wedge\; \|E\|_{\mathcal{V}} = P \;\wedge\; \|\vec{e}\|_{\mathcal{V}} = \vec{d})\}$$

The semantics of tagged predicates is defined as follows:

$$\|\nu Z\{L\}.\Pi\|_{\mathcal{V}} \;\triangleq\; \sqcup\{\mathcal{E}_{\vec{D}} \mid \mathcal{E}_{\vec{D}} \sqsubseteq \sqcup L \sqcup \|\Pi\|_{\mathcal{V}[\mathcal{E}_{\vec{D}}/Z]}\}$$

$$\|\mu Z\{L\}.\Pi\|_{\mathcal{V}} \;\triangleq\; \sqcap\{\mathcal{E}_{\vec{D}} \mid \mathcal{E}_{\vec{D}} \sqsupseteq \sqcup L \sqcup \|\Pi\|_{\mathcal{V}[\mathcal{E}_{\vec{D}}/Z]}\}$$

where $\sqcup L \triangleq \bigsqcup_{l\in L} \mathcal{E}^l_{\vec{D}}$. If L is empty, $\sqcup L = \lambda\vec{d}.\{\}$, and hence $\sigma Z.\Pi \equiv \sigma Z\{\}.\Pi$.

In rule $\Phi(\nu 0)$, $\mathcal{C}_{\nu 0}$ demands $(B, E, \vec{e}) \in L$, while in $\Phi(\nu 1)$, $l = (B, E, \vec{e})$ and $\mathcal{C}_{\nu 1}$ demands $(B, E, \vec{e}) \notin L$. The side conditions are purely syntactical and require no semantic reasoning. Note that $l \in L$ implies $\mathcal{E}^l_{\vec{D}} \sqsubseteq \sqcup L$. Condition $\mathcal{C}_{\nu 0}$ might seem rather strong but, as our first example proof below shows, when used in combination with the general rules rule $\Phi(\nu 0)$ becomes sufficiently powerful.

The treatment of least fixpoint formulae is far more complicated. We employ an idea by H.Andersen [1] to use the tagging technique for inductive reasoning. However, instead of doing induction on sets of processes explicitly we do this implicitly by using induction on the domain of values. In this way we avoid the introduction of infinitary proof rules in our proof system. One has, however, to introduce into the language a new syntactic category, namely *arbitrary constants*. These are to be treated syntactically as constants from the domain of values, but do not denote any particular values. Valuations do not assign values to arbitrary constants, so we obtain an implicit second level of universal quantification which is necessary to conduct an inductive argument over the domain of values. The side conditions for rules $\Phi(\mu 0)$ and $\Phi(\mu 1)$ are as follows. $\mathcal{C}_{\mu 0}$ is exactly as $\mathcal{C}_{\nu 0}$. $\mathcal{C}_{\mu 1}$ requires that there exists a vector $\vec{x}$ of variables and a vector of the same length $\vec{c}$ of fresh arbitrary constants so that B', E' and $\vec{e}\,'$ are obtained by substituting $\vec{c}$ for $\vec{x}$ in B, E and $\vec{e}$, respectively. Furthermore, there exist B'' and $\vec{e}\,''$ having as free variables exactly the ones in $\vec{x}$, so that B'' implies $B[\vec{e}\,''/\vec{x}]$ under any valuation, the relation $\prec\!\cdot \subseteq \vec{D} \times \vec{D}$ defined by:

$$\vec{d}_1 \prec\!\cdot\, \vec{d}_2 \;\stackrel{\Delta}{\equiv}\; \left[\!\left[B''[\vec{d}_2/\vec{x}]\right]\!\right] = \mathsf{tt} \,\wedge\, \vec{d}_1 = \left[\!\left[\vec{e}\,''[\vec{d}_2/\vec{x}]\right]\!\right]$$

is a *well-founded relation* (i.e. has no infinite decreasing chains), and $l = l''[\vec{c}/\vec{x}]$ for $l'' = (B'', E[\vec{e}\,''/\vec{x}], \vec{e}\,[\vec{e}\,''/\vec{x}])$. Note that rule $\Phi(\mu 1)$ can always be applied trivially by choosing $\vec{x}$, $\vec{c}$ and $\vec{e}\,''$ to be null-ary vectors and B'' to be false; this is equivalent to simple unfolding without changing the tag.

A *proof* for a sequent is a proof tree in which this sequent is the root and all leaves are axioms. If such a proof exists we term the sequent *derivable*. In our proof of soundness we refer to the following two results.

Proposition 3.1 *For any B, E, $\vec{e}$ and indexed set $\mathcal{E}_{\vec{D}}$ the following holds:*

$$\mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \mathcal{E}_{\vec{D}} \;\equiv\; \forall \mathcal{V}.(\|B\|_{\mathcal{V}} = \mathsf{tt} \;\rightarrow\; \|E\|_{\mathcal{V}} \in \mathcal{E}_{\|\vec{e}\|_{\mathcal{V}}})$$

Lemma 3.2 (Reduction lemma) *For any monotone $f : [\vec{D} \rightarrow \wp(S)] \rightarrow [\vec{D} \rightarrow \wp(S)]$ and any $U_{\vec{D}} : \vec{D} \rightarrow \wp(S)$:*

$$U_{\vec{D}} \sqsubseteq \nu f \;\equiv\; U_{\vec{D}} \sqsubseteq f(\nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}})))$$

Proof. Our proof is based on the following two relationships:

(1) $U_{\vec{D}} \sqcup f(\nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))) \;=\; \nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))$ {fix-point property}

(2) $\nu f \;\sqsubseteq\; \nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))$ {easy to show}

($\rightarrow$) Follows immediately from (2) and monotonicity of f.
($\leftarrow$) Let $U_{\vec{D}} \sqsubseteq f(\nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}})))$. Then $f(\nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))) = \nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))$ because of (1), and hence $\nu \mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))$ is a fix-point of f and is thereby less or equal to νf since the latter is the greatest fix-point of f. It

follows by (2), that $\nu f = \nu\mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))$ and consequently:

$$U_{\vec{D}} \sqsubseteq f(\nu\mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}}))) = \nu\mathcal{E}_{\vec{D}}.(U_{\vec{D}} \sqcup f(\mathcal{E}_{\vec{D}})) = \nu f$$

which completes the proof. □

Our system is sound, as the following theorem states.

Theorem 3.3 (Soundness) *If sequent $B \vdash E : \Phi$ is derivable in the proof system, then $B \models E : \Phi$ holds.*

Proof. It is sufficient to show (by case analysis), that all rules are individually sound; the result then follows by induction on the height of the derivation tree. For most of the rules this is straightforward; the only interesting cases are the fixpoint rules. The soundness of $\Phi(\nu 0)$ and $\Phi(\nu 1)$ can be established as follows (assuming $\nu Z\{L\}.\Pi$ is closed):

$$\begin{array}{ll}
\quad B \models E : (\nu Z\{L\}.\Pi)\vec{e} & \\
\equiv \forall\mathcal{V}.(\|B\|_{\mathcal{V}} = \mathsf{tt} \rightarrow \|E\|_{\mathcal{V}} \in \|(\nu Z\{L\}.\Pi)\vec{e}\|_{\mathcal{V}}) & \{\text{Def. } \models\} \\
\equiv \forall\mathcal{V}.(\|B\|_{\mathcal{V}} = \mathsf{tt} \rightarrow \|E\|_{\mathcal{V}} \in \|\nu Z\{L\}.\Pi\|\,\|\vec{e}\|_{\mathcal{V}}) & \{\text{Def. } 2.4\} \\
\equiv \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \|\nu Z\{L\}.\Pi\| & \{\text{Prop. } 3.1\} \\
\equiv \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \nu\mathcal{E}_{\vec{D}}.(\sqcup L \sqcup \|\Pi\|_{\mathcal{V}[\mathcal{E}_{\vec{D}}/Z]}) & \{\text{Tarski-Knaster, any } \mathcal{V}\} \\
\equiv \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \sqcup L \sqcup \|\Pi\|_{\mathcal{V}[\nu\mathcal{E}_{\vec{D}}.((\sqcup L \sqcup \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})}) \sqcup \|\Pi\|_{\mathcal{V}[\mathcal{E}_{\vec{D}}/Z]})/Z]} & \{\text{Lemma } 3.2\} \\
\equiv \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \sqcup L \sqcup \|\Pi\|_{\mathcal{V}[\|\nu Z\{l,L\}.\Pi\|_{\mathcal{V}}/Z]} & \{l \triangleq (B,E,\vec{e})\} \\
\equiv \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \sqcup L \sqcup \|\Pi[\nu Z\{l,L\}.\Pi/Z]\| & \{\text{Prop. of substitution}\}
\end{array}$$

From this it follows that:

$$\begin{array}{rll}
(B,E,\vec{e}) \in L \rightarrow & \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \sqcup L & \{\text{Def. } \sqcup L\} \\
\rightarrow & B \models E : (\nu Z\{L\}.\Pi)\vec{e} & \{\text{above equivalence}\}
\end{array}$$

i.e. that $\Phi(\nu 0)$ is sound, and

$$\begin{array}{ll}
\quad B \models E : (\Pi[\nu Z\{l,L\}.\Pi/Z])\vec{e} & \\
\equiv \forall\mathcal{V}.(\|B\|_{\mathcal{V}} = \mathsf{tt} \rightarrow \|E\|_{\mathcal{V}} \in \|(\Pi[\nu Z\{l,L\}.\Pi/Z])\vec{e}\|_{\mathcal{V}}) & \{\text{Def. } \models\} \\
\equiv \forall\mathcal{V}.(\|B\|_{\mathcal{V}} = \mathsf{tt} \rightarrow \|E\|_{\mathcal{V}} \in \|\Pi[\nu Z\{l,L\}.\Pi/Z]\|\,\|\vec{e}\|_{\mathcal{V}}) & \{\text{Def. } 2.4\} \\
\equiv \mathcal{E}_{\vec{D}}^{(B,E,\vec{e})} \sqsubseteq \|\Pi[\nu Z\{l,L\}.\Pi/Z]\| & \{\text{Prop. } 3.1\} \\
\rightarrow B \models E : (\nu Z\{L\}.\Pi)\vec{e} & \{\text{above equivalence}\}
\end{array}$$

i.e. that $\Phi(\nu 1)$ is sound. The case $\Phi(\mu 0)$ is similar to $\Phi(\nu 0)$. The case with rule $\Phi(\mu 1)$ is the most involved one and because of its size can only be sketched here. Assume condition $\mathcal{C}_{\mu 1}$ holds for some $\vec{x}$, $\vec{c}$, B'' and $\vec{e}\,''$, and assume the notation chosen in the explanation of the condition. Let furthermore l' denote $(B, E, \vec{e})$. Assume $B' \models E' : (\Pi[\mu Z\{l, L\}.\Pi/Z])\vec{e}\,'$ holds. Using Proposition 3.1 above, Lemma 5.1. from [1], and the semantics of arbitrary constants, one can show that for any $\vec{d} \in \vec{D}$, $\mathcal{E}_{\vec{D}}^{l''[\vec{d}/\vec{x}]} \sqsubseteq \|\mu Z\{L\}.\Pi\|$ implies $\mathcal{E}_{\vec{D}}^{l'[\vec{d}/\vec{x}]} \sqsubseteq \|\mu Z\{L\}.\Pi\|$. As a consequent, due to the assumed properties of B'' and $\prec\cdot$, $\mathcal{E}_{\vec{D}}^{l'[\vec{d}/\vec{x}]} \sqsubseteq \|\mu Z\{L\}.\Pi\|$ holds for all minimal elements $\vec{d}$ of $\prec\cdot$, and whenever $\vec{d}_1 \prec\cdot\, \vec{d}_2$ then $\mathcal{E}_{\vec{D}}^{l'[\vec{d}_1/\vec{x}]} \sqsubseteq \|\mu Z\{L\}.\Pi\|$ implies $\mathcal{E}_{\vec{D}}^{l'[\vec{d}_2/\vec{x}]} \sqsubseteq \|\mu Z\{L\}.\Pi\|$. Since $\prec\cdot$ is well-founded we have by induction on $\vec{D}$ that $\mathcal{E}_{\vec{D}}^{l'[\vec{d}/\vec{x}]} \sqsubseteq \|\mu Z\{L\}.\Pi\|$ for all $\vec{d} \in \vec{D}$, which is equivalent to $B \models E : (\mu Z\{L\}.\Pi)\vec{e}$. $\square$

3.3 Example Proofs

In this subsection we present two example proofs conducted in the proof system given above. First, consider process P defined as follows:

$$\begin{aligned}
P &\triangleq Q(0) \\
Q(x) &\triangleq R(x, x+1) \\
R(x,y) &\triangleq \overline{out}(x+2, y+3).R(x+1, y+2)
\end{aligned}$$

This process is capable of engaging in the (infinite) sequence of interactions

$$s_0 \xrightarrow{\overline{out}(2,4)} s_1 \xrightarrow{\overline{out}(3,6)} s_2 \xrightarrow{\overline{out}(4,8)} \cdots$$

i.e. of putting out all consecutive combinations $(x, 2x)$ starting with $x = 2$. In our logic this capability (property) could be described as $\Pi(2)$ where:

$$\Pi \triangleq \nu Z.\lambda x.\, \langle \overline{out}(x, 2x) \rangle\, Z(x+1)$$

The proof could be conducted (in a "goal-oriented" fashion) as follows:

$$\begin{array}{cl}
\vdash P : \Pi(2) & \\
\hline
\vdash Q(0) : \Pi(2) & \mathrm{E}(\triangleq) \\
\hline
\vdash R(0, 0+1) : \Pi(2) & \mathrm{E}(\triangleq) \\
\hline
\vdash R(0, 2\times 0+1) : \Pi(0+2) & \mathrm{G}(\equiv) \\
\hline
\vdash R(x, 2x+1) : \Pi(x+2) & \mathrm{G}(\mathsf{US}) \\
\hline
\vdash R(x, 2x+1) : (\lambda x.\, \langle \overline{out}(x, 2x) \rangle\, \Pi_1(x+1))(x+2) & \Phi(\nu 1) \\
\hline
\vdash R(x, 2x+1) : \langle \overline{out}(x+2, 2(x+2)) \rangle\, \Pi_1((x+2)+1) & \Phi(\beta) \\
\hline
\vdash \overline{out}(x+2, (2x+1)+3).R(x+1, (2x+1)+2) : \langle \overline{out}(x+2, 2(x+2)) \rangle\, \Pi_1((x+2)+1) & \mathrm{E}(\triangleq) \\
\hline
\vdash \overline{out}(x+2, 2x+4).R(x+1, 2x+3) : \langle \overline{out}(x+2, 2x+4) \rangle\, \Pi_1(x+3) & \mathrm{G}(\equiv) \\
\hline
\vdash R(x+1, 2x+3) : \Pi_1(x+3) & \mathrm{E}\Phi(\overline{a}, \langle\rangle) \\
\hline
\vdash R(x+1, 2(x+1)+1) : \Pi_1((x+1)+2) & \mathrm{G}(\equiv) \\
\hline
\vdash R(x, 2x+1) : \Pi_1(x+2) & \mathrm{G}(\mathsf{US}) \\
\hline
. & \Phi(\nu 0)
\end{array}$$

where $\Pi_1 \triangleq \nu Z\{(\epsilon, R(x, 2x+1), x+2)\}.\lambda x.\, \langle \overline{out}(x, 2x)\rangle\, Z(x+1)$. It is worth noticing the way in which rule G(US) is used in combination with rule G($\equiv$); in the first case as a widening, allowing the proof to be completed with just one unfolding of the ν-formula; and in the second to allow for rule $\Phi(\nu 0)$ to be applied.

Our second example illustrates the use of rules $\Phi(\mu 0)$ and $\Phi(\mu 1)$. Consider process P

$$
\begin{aligned}
P &\triangleq in(x).Q(x) \\
Q(x) &\triangleq \mathbf{if}\ x > 0\ \mathbf{then}\ \overline{out}(x).Q(x-1)
\end{aligned}
$$

and its property Φ

$$
\begin{aligned}
\Phi &\triangleq \forall x.\, [in(x)]\, \Pi(x) \\
\Pi &\triangleq \mu Z.\lambda x.\, [\overline{out}(x)]\, Z(x-1)
\end{aligned}
$$

discussed in the Introduction. The proof goes as follows:

$$
\begin{array}{cl}
\vdash\ P\ :\ \forall x.\, [in(x)]\, \Pi(x) & \\ \hline
\vdash\ P\ :\ [in(y)]\, \Pi(y) & \Phi(\forall) \\ \hline
\vdash\ in(x).Q(x)\ :\ [in(y)]\, \Pi(y) & \mathrm{E}(\triangleq) \\ \hline
\vdash\ Q(y)\ :\ \Pi(y) & \mathrm{E}\Phi(a, [\,]) \\ \hline
\vdash\ Q(c)\ :\ (\lambda x.\, [\overline{out}(x)]\, \Pi_c(x-1))(c) & \Phi(\mu 1) \\ \hline
\vdash\ Q(c)\ :\ [\overline{out}(c)]\, \Pi_c(c-1) & \Phi(\beta) \\ \hline
\vdash\ \mathbf{if}\ c > 0\ \mathbf{then}\ \overline{out}(c).Q(c-1)\ :\ [\overline{out}(c)]\, \Pi_c(c-1) & \mathrm{E}(\triangleq) \\ \hline
c > 0\ \vdash\ \overline{out}(c).Q(c-1)\ :\ [\overline{out}(c)]\, \Pi_c(c-1) & \mathrm{E}\Phi(\mathbf{if}, [\,]) \\ \hline
c > 0, c = c\ \vdash\ Q(c-1)\ :\ \Pi_c(c-1) & \mathrm{E}\Phi(\overline{a}, [\,]) \\ \hline
c > 0\ \vdash\ Q(c-1)\ :\ \Pi_c(c-1) & \mathrm{G}(\Rightarrow) \\ \hline
. & \Phi(\mu 0)
\end{array}
$$

where $\Pi_c \triangleq \mu Z\{(c > 0, Q(c-1), c-1)\}.\lambda x.\, [\overline{out}(x)]\, Z(x-1)$, and where G($\Rightarrow$) is the obvious derived rule. Rule $\Phi(\mu 1)$ has been applied with y, c, $y > 0$, and $y - 1$ for $\vec{x}$, $\vec{c}$, B'', and $\vec{e}\,''$ in the description of condition $\mathcal{C}_{\mu 1}$, respectively.

This example can be used to explain how the least fix-point rules support inductive arguments: for any *fixed* value of y, by using simple unfolding instead of tagging and by repeating the sequence of inferences between the fourth and the last lines of the proof, we are guaranteed to reach in finitely many steps a sequent whose condition B is false and the (axiom) rule G(ff) is therefore applicable.

More example proofs can be found in [9].

4 Conclusion

In this paper we suggested an extension of the modal μ-calculus for expressing properties of processes with value passing. To allow for convenient verification of processes described in the value passing CCS without parallel composition, we constructed a compositional proof system. We showed that the proof system is sound. This proof system can be considered as a step towards finding an appropriate framework for the verification of value-passing processes. Although fully automatic verification is not achievable due to the expressive power of the presented logic, the verification process can be machine-assisted to a considerable extent since most rules do not need human intervention to be applied and the need for external semantic reasoning has been kept minimal.

An important question is whether the presented proof system is *complete*, i.e. whether $B \models E : \Phi$ implies that $B \vdash E : \Phi$ is derivable in the proof system. When asking this question one has to factor out any reasoning concerning the value domain D like equivalence of value expressions, entailment of Boolean expressions etc. Our proof system has been tailored towards such a *relative completeness*, but concrete completeness results have still to be obtained.

The proof system presented here can be considered, following an approach of C.Stirling [14], as the first part of a more general proof system which can also deal with processes involving parallel composition. The other part of the latter system is for sequents of the form $B \vdash \Phi_1, \Phi_2 : \Phi$. The semantics of such sequents is given by

$$
\begin{aligned}
B \models \Phi_1, \Phi_2 : \Phi \;\stackrel{\Delta}{\equiv}\; & \forall \mathcal{V}.\ \forall P_1, P_2. \\
& (\|B\|_{\mathcal{V}} = \mathsf{tt} \wedge P_1 \in \|\Phi_1\|_{\mathcal{V}}^{\mathcal{T}(P_1)} \wedge P_2 \in \|\Phi_1\|_{\mathcal{V}}^{\mathcal{T}(P_2)} \\
& \longrightarrow P_1 | P_2 \in \|\Phi_1\|_{\mathcal{V}}^{\mathcal{T}(P_1|P_2)})
\end{aligned}
$$

The two systems are connected through the following rule:

$$
\mathrm{E}(|)\ \frac{B \vdash E_1 | E_2 : \Phi}{B \vdash E_1 : \Phi_1 \qquad B \vdash \Phi_1, \Phi_2 : \Phi \qquad B \vdash E_2 : \Phi_2}
$$

We treat the problem of constructing proof systems of the above type separately [3].

Acknowledgement

The authors wish to thank Scott Hazelhurst for many helpful comments on earlier versions of the paper.

References

[1] H.R.Andersen, Verification of Temporal Properties of Concurrent Systems, PhD thesis, Department of Computer Science, Aarhus University, Denmark, June 1993.

[2] H.R.Andersen, C.Stirling and G.Winskel, A Compositional Proof System for the Modal μ-Calculus, in: Proceedings of LICS'94, 1994.

[3] S.Berezin and D.Gurov, A Compositional Proof System for the First Order Modal μ-calculus and Value Passing CCS, unpublished manuscript, March 1996.

[4] J.Bradfield and C.Stirling, Local Model Checking for Infinite State Spaces, *Theoretical Computer Science*, 96:157-174, 1992.

[5] G.Bruns, A Practical Technique for Process Abstraction, in: Proceedings of CONCUR'93, *Lecture Notes in Computer Science*, 715:37-49, 1993.

[6] R.Cleaveland, Tableau-based Model Checking in the Propositional mu-Calculus, *Acta Informatica*, 27:725-747, 1990.

[7] M.Dam, Model Checking Mobile Processes, in: Proceedings of CONCUR'93, *Lecture Notes in Computer Science*, 715:22-36, 1993.

[8] M.Dam, Compositional Proof Systems for Model Checking Infinite State Processes, in: Proceedings of CONCUR'95, *Lecture Notes in Computer Science*, 962:12-26, 1995.

[9] D.Gurov, S.Berezin and B.Kapron, A Compositional Proof System for a Subset of the Value Passing CCS, Report DCS-240-IR, Department of Computer Science, University of Victoria, February 1996.

[10] M.Hennessy and X.Liu, A Modal Logic for Message Passing Processes, Report 3/93, Department of Computer Science, University of Sussex, January 1993.

[11] D.Kozen, Results on the Propositional μ-Calculus, *Theoretical Computer Science*, 27:333-354, 1983.

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

[13] J.Rathke and M.Hennessy, Local Model Checking for a Value-Based Modal μ-Calculus, Report 5/96, Department of Computer Science, University of Sussex, June 1996.

[14] C.Stirling, Modal Logics for Communicating Systems, *Theoretical Computer Science*, 49:311-347, 1987.

[15] C.Stirling and D.Walker, Local Model Checking in the Modal mu-Calculus, *Theoretical Computer Science*, 89(1):161-177, 1991.

[16] A.Tarski, A Lattice-Theoretical Fixedpoint Theorem and its Applications, *Pacific Journal of Mathematics*, 5:285-309, 1955.

[17] G.Winskel, A Note on Model Checking the Modal ν-Calculus, *Theoretical Computer Science*, 83:157-167, 1991.