Electronic Notes in Theoretical Computer Science 3 (1996)
URL: http://www.elsevier.nl/locate/entcs/volume3.html 15 pages

Encoding Transition Systems in Sequent Calculus: Preliminary Report

Raymond McDowell

Computer and Information Science Department, University of Pennsylvania
Philadelphia, PA 19104-6389 USA

Dale Miller

Computer and Information Science Department, University of Pennsylvania
Philadelphia, PA 19104-6389 USA

Catuscia Palamidessi

Dipartimento di Informatica e Scienze dell'Informazione
Via Dodecaneso, 35, 16146 Genova ITALY

Abstract

Linear logic has been used to specify the operational semantics of various process calculi. In this paper we explore how meta-level judgments, such as simulation and bisimulation, can be established using such encodings. In general, linear logic is too weak to derive such judgments and we focus on an extension to linear logic using definitions. We explore this extension in the context of transition systems.

1 Proof theory preliminaries

In a recent note [5], Girard extended linear logic with a notion of *definitions*. If certain restrictions are placed on the structure of definitions then defined concepts have left and right introduction rules that enjoy a cut-elimination theorem. Some examples of using such a definition mechanism have been given for equality reasoning [5,9], forms of program completion in logic programming [6,10], and in the GCLA language project [1].

Given that linear logic has been successful in specifying various transition systems used in concurrency theory [3,7], it is natural to ask what such a definition facility adds to specifications written in linear logic. In this paper, we show that if the specification of a transition system is made into a definition (instead of just a theory), then it is possible to go beyond operational semantics and also prove judgments such as simulation and bisimulation.

We shall assume that the reader is familiar with the two-sided sequent calculus presentation of linear logic [4]. For the purposes of this paper, we need only the connectives $1, \top, \otimes, \&, \multimap, \forall, \exists$. Our meta-logic is provided with simple types: the type of logical formulas, for example, is o (following [2]). *Definitions* will be written in the following style:

$$\forall \bar{x}_1[p_1(\bar{t}_1) \circ\!\!- H_1] \quad \cdots \quad \forall \bar{x}_n[p_n(\bar{t}_n) \circ\!\!- H_n] \quad (n \geq 0)$$

Here, we assume that the formulas $H_1, \ldots, H_n$ do not contain exponentials (in our case, this is immediate since we have not admitted either ! or ?), all free variables of H_i are free in some term of the list of terms $\bar{t}_i$, and all variables free in some $\bar{t}_i$ are contained in the list of variables $\bar{x}_i$ $(i = 1, \ldots, n)$. We do not assume that the predicates $p_1, \ldots, p_n$ are distinct. For $i = 1, \ldots, n$, the expression $\forall \bar{x}_i[p_i(\bar{t}_i) \circ\!\!- H_i]$ is a *clause* of the definition and H_i is the *body* and $p_i(\bar{t}_i)$ is the *head* of that clause.

Definitions can be used to give both left and right introduction rules for atomic formulas. If suitable restrictions are made on these definitions (such as the above mentioned one that no modals appear in the body of definitions), then cut-elimination can be proved for the resulting logic extended with these defined non-logical constants (for a proof, see [9]). The right and left introduction rules for definitions can be given as follows (here, A is atomic, Δ is a multiset of formulas, and B is a formula).

$$\frac{\Delta \longrightarrow \theta H_i}{\Delta \longrightarrow A}\ BC, \text{ where } \theta \text{ is a substitution such that } A \text{ is } \theta p_i(\bar{t}_i)$$

$$\frac{\{\theta\Delta, \theta H_i \longrightarrow \theta C \mid \theta \text{ is the mgu for } A \text{ and } p_i(\bar{t}_i)\}}{\Delta, A \longrightarrow C}\ DR$$

Specifying a set of sequents as the premise in the left introduction rule should be understood to mean that each sequent in the set is a premise of the rule. Here we name the right introduction rule for definitions as *backchaining* (BC), since it has that name in the logic programming literature, and we name the left introduction rule for definitions as *definitional reflection* (DR), following Schroeder-Heister [9]. Notice the different "quantificational" interpretation of these two rules when reading them bottom-up: BC replaces A with the body of *some* clause in the definition whose head matches with A, while DR replaces A with the body of *all* the clauses whose heads unify with A. These different quantificational aspects play an important role in our uses of DR and BC below.

If D is a definition, we write $D \vdash \Delta \longrightarrow C$ to mean that $\Delta \longrightarrow C$ is provable using the inference rules of linear logic plus BC for clauses in D, and $D \Vdash \Delta \longrightarrow C$ to mean that $\Delta \longrightarrow C$ is provable using the inference rules of linear logic plus BC and DR for clauses in D. The first notion can be reduced to provability in linear logic directly: If D is the definition displayed above and if $\hat{D}$ is the formula

$$\forall \bar{x}_1[H_1 \multimap p_1(\bar{t}_1)] \,\&\, \ldots \,\&\, \forall \bar{x}_n[H_n \multimap p_n(\bar{t}_n)]$$

then $D \vdash \Delta \longrightarrow C$ if and only if $!\hat{D}, \Delta \longrightarrow C$ has a proof in the usual sequent calculus for linear logic. The notion of $D \Vdash \Delta \longrightarrow C$ cannot be so reduced to provability in linear logic. In this case, D is treated as a definition and we consider the proof system of linear logic to be extended to include left and right rules (DR and BC) for the defined predicates in D. We write $D \vdash B$ and $D \Vdash B$ as abbreviations for $D \vdash\longrightarrow B$ and $D \Vdash\longrightarrow B$, respectively.

2 Linear logic presentations of two process calculi

In this section, definitions will be treated just as theories, or, so to say, logic programs. In Section 3 we will start using the inference rule of definitional reflection.

2.1 Abstract Transition Systems

The triple $\mathcal{T} = \langle \Lambda, S, \delta \rangle$ is an *abstract transition system* (*ats*) if Λ is a non-empty set of *actions*, S is a non-empty set of *states*, and $\delta \subseteq S \times \Lambda \times S$ (Λ and S are assumed to be disjoint). We write $p \xrightarrow{a} q$ if $\langle p, a, q \rangle \in \delta$. If $w \in \Lambda^*$ then we write $p \stackrel{w}{\Longrightarrow} q$ to mean that p makes a transition to q along a path of actions given by w. More formally, this relation is defined by induction on the length of w: thus $p \stackrel{\epsilon}{\Longrightarrow} p$ and if $p \xrightarrow{a} r$ and $r \stackrel{w}{\Longrightarrow} q$ then $p \stackrel{aw}{\Longrightarrow} q$. The ats $\mathcal{T}$ is *finite* if both Λ and S are finite; is *noetherian* if it contains no infinite length paths; is *determinate* if for every state p and every action a, the set $\{q \mid (p, a, q) \in \delta\}$ is either empty or a singleton.

Transitions as logical formulas. Our first encoding represents states as propositional constants and actions as propositional functions of type $o \to o$. A transition $\langle p, a, q \rangle \in \delta$ is encoded as the clause

$$p \circ\!- \exists W(aW \otimes (W \multimap q)),$$

which could be written equivalently as the formula $\forall W((W \multimap q) \multimap (aW \multimap p))$. Let $ats_1(\delta)$ be the definition composed of the clauses encoding all the transitions in δ. Finally, we represent a word $a_1 \cdots a_m$ ($m \geq 0$) from Λ^* as the proposition $a_1(\ldots(a_m 1)\ldots)$ (the empty word is thus encoded as 1).

Proposition 2.1 *Let $\langle \Lambda, S, \delta \rangle$ be an ats. Then $ats_1(\delta) \vdash q, w \longrightarrow p$ if and only if $p \stackrel{w}{\Longrightarrow} q$.*

This proposition can be proved by showing that a path in an ats can faithfully match a sequence of inference rules in a proof. For example, consider proving the sequent $q, a(w') \longrightarrow p$, where p and q denote states and a denotes an action. Up to permutations of inference rules, this proof must end with the following inference rules:

$$\dfrac{\dfrac{\dfrac{\overline{a(w') \longrightarrow a(w')} \quad \dfrac{q, w' \longrightarrow r}{q \longrightarrow w' \multimap r}\multimap R}{q, a(w') \longrightarrow a(w') \otimes (w' \multimap r)}\otimes R}{q, a(w') \longrightarrow \exists W(aW \otimes (W \multimap r))}\exists R}{q, a(w') \longrightarrow p} BC$$

We have reduced proving $q, a(w') \longrightarrow p$ to proving $q, w' \longrightarrow r$: this corresponds to the state transition $p \xrightarrow{a} r$. The empty path then corresponds to the proof

$$\dfrac{\overline{q \longrightarrow q}}{q, 1 \longrightarrow q}\ 1L$$

Transitions given as a table. In this second encoding we use the primitive type σ to denote elements of S and α to denote elements of Λ. Let $one\colon \sigma \to \alpha \to \sigma \to o$ be a predicate of three arguments denoting the one step transition relation and let the definition $ats_2(\delta)$ contain the clause $one(p, a, q) \mathrel{\circ\!-} 1$ for every $\langle p, a, q\rangle \in \delta$. This definition is simple in the sense that it does not use variables or (non-trivial) bodies. We also need the following definition, named $path$:

$$\begin{array}{ll}
\forall P & [\quad multi(P, nil, P) \mathrel{\circ\!-} 1]. \\
\forall A, P, Q, W & [multi(P, A :: W, Q) \mathrel{\circ\!-} \exists R(one(P, A, R) \otimes multi(R, W, Q))].
\end{array}$$

Here, members of Λ^* are represented as terms of type $list(\alpha)$ using $nil : list(\alpha)$ for the empty list and $::$ of type $\alpha \to list(\alpha) \to list(\alpha)$ for the list constructor. The following proposition should be contrasted to the proposition above; its proof is similar.

Proposition 2.2 *Let $\langle \Lambda, S, \delta\rangle$ be an ats. Then $ats_2(\delta), path \vdash multi(p, w, q)$ if and only if $p \stackrel{w}{\Longrightarrow} q$.*

2.2 CCS

We use types σ and α here for CCS [8] expressions and for actions, respectively. The combinators for CCS that we consider here (ignoring renaming and hiding for convenience) are prefixing (of type $\alpha \to \sigma \to \sigma$), plus and parallel composition (both of type $\sigma \to \sigma \to \sigma$), recursion given by the μ operator (of type $(\sigma \to \sigma) \to \sigma$), and 0, the inactive process. The expression $\mu_x P$ will be used to abbreviate $\mu(\lambda x.P)$. Also, the over-bar constructor is of type $\alpha \to \alpha$. Compare the following definitions with the usual CCS transition system given in Figure 1.

$$\begin{array}{rlll}
\forall A, P\, [& one(A.P, A, P) & \mathrel{\circ\!-} & 1]. \\
\forall A, P, Q, R\, [& one(P \mid R, A, Q \mid R) & \mathrel{\circ\!-} & one(P, A, Q)]. \\
\forall A, P, Q, R\, [& one(R \mid P, A, R \mid Q) & \mathrel{\circ\!-} & one(P, A, Q)]. \\
\forall A, P, Q, R\, [& one(P + R, A, Q) & \mathrel{\circ\!-} & one(P, A, Q)]. \\
\forall A, P, Q, R\, [& one(P + R, A, Q) & \mathrel{\circ\!-} & one(R, A, Q)]. \\
\forall A, P, Q\, [& one(\mu P, A, Q) & \mathrel{\circ\!-} & one(P(\mu P), A, Q)]. \\
\forall P, Q, R, S\, [& one(P \mid Q, \tau, R \mid S) & \mathrel{\circ\!-} & \exists A, B(comp(A, B) \otimes \\
 & & & \qquad one(P, A, R) \otimes one(Q, B, S))].
\end{array}$$

Let $\mathcal{A}$ be a finite, non-empty set of actions (assume that $\tau : \alpha$ is not a member of $\mathcal{A}$) and let $ccs(\mathcal{A})$ be the definition composed of all those formulas displayed above plus all formulas of the form $comp(a, \bar{a}) \mathrel{\circ\!-} 1$ and $comp(\bar{a}, a) \mathrel{\circ\!-} 1$ for every $a \in \mathcal{A}$.

$$\frac{}{A.P \xrightarrow{A} P} \qquad \frac{P \xrightarrow{A} Q}{P \mid R \xrightarrow{A} Q \mid R} \qquad \frac{P \xrightarrow{A} Q}{R \mid P \xrightarrow{A} R \mid Q} \qquad \frac{P \xrightarrow{A} R \quad Q \xrightarrow{\bar{A}} S}{P \mid Q \xrightarrow{\tau} R \mid S}$$

$$\frac{P \xrightarrow{A} Q}{P + R \xrightarrow{A} Q} \qquad \frac{R \xrightarrow{A} Q}{P + R \xrightarrow{A} Q} \qquad \frac{P[\mathrm{fix}(X = P)/X] \xrightarrow{A} Q}{\mathrm{fix}(X = P) \xrightarrow{A} Q}$$

Fig. 1. CCS transition rules

Proposition 2.3 *Let $\mathcal{A}$ be a finite set of actions. Then $p \xrightarrow{a} q$ if and only if $ccs(\mathcal{A}) \vdash one(p, a, q)$, and $p \stackrel{w}{\Longrightarrow} q$ if and only if $ccs(\mathcal{A}), path \vdash multi(p, w, q)$.*

CCS can be seen as an abstract transition system where $\Lambda = \mathcal{A} \cup \bar{\mathcal{A}} \cup \{\tau\}$ and S is the set of all expressions denoting CCS expressions (of type σ). A *finite CCS process* is a CCS process that does not contain μ. If S is instead the set of all finite CCS processes, then the resulting ats is noetherian. Also the judgment $p \xrightarrow{a} q$ is decidable when p is a finite process.

3 Simulation and bisimulations for finite behaviors

A relation $\mathcal{R}$ is a *simulation* between p and q (notation $p\mathcal{R}q$, to be read as "q simulates p") if and only if for every transition $p \xrightarrow{a} p'$, there exists a transition $q \xrightarrow{a} q'$, such that $p'\mathcal{R}q'$. The largest such relation is written $\sqsubseteq$; that is, $p \sqsubseteq q$ if and only if there exists a simulation $\mathcal{R}$ such that $p\mathcal{R}q$. A relation $\mathcal{R}$ is a *bisimulation* between p and q if and only if for every transition $p \xrightarrow{a} p'$, there exists a transition $q \xrightarrow{a} q'$ such that $p'\mathcal{R}q'$, and for every transition $q \xrightarrow{a} q'$, there exists a transition $p \xrightarrow{a} p'$ such that $q'\mathcal{R}p'$. The largest such relation is called the *bisimulation equivalence* and is denoted by $\equiv$; that is, $p \equiv q$ (read as "p is *bisimilar* to q") if and only if there exists a bisimulation $\mathcal{R}$ such that $p\mathcal{R}q$.

Given a finite ats $M = \langle \Lambda, S, \delta \rangle$, define $\langle\langle p \rangle\rangle = \{(a, q) \mid (p, a, q) \in \delta\}$. Now consider a proof of the sequent $p \longrightarrow q$ using the definition $ats_1(\delta)$. If p and q are the same state then this sequent is initial. Otherwise, the only inference rules that can be applied to prove such a sequent are either BC or DR. It is easy to show that occurrences of DR can be permuted down over BC and the right introduction rules for $\exists$ and $\otimes$. So we can assume that the last inference rule used to prove this sequent is DR, and thus the proof has the form

$$\frac{\exists W(a_1 W \otimes (W \multimap p_1)) \longrightarrow q \quad \ldots \quad \exists W(a_m W \otimes (W \multimap p_m)) \longrightarrow q}{p \longrightarrow q} \; DR,$$

where $m \geq 0$ and $\langle\langle p \rangle\rangle = \{(a_1, p_1), \ldots, (a_m, p_m)\}$. Because of the presence of enough permutations, each of the premises here can be proved by left introduction rules for $\exists$ and $\otimes$, yielding the premises

$$a_1 w_1, w_1 \multimap p_1 \longrightarrow q \quad \ldots \quad a_m w_m, w_m \multimap p_m \longrightarrow q,$$

where $w_1, \ldots, w_m$ are variables. At this point, these sequents can only be proved by using BC. For example,

$$a_i w_i, w_i \multimap p_i \longrightarrow \exists W(a_j W \otimes (W \multimap q_j)),$$

where $\langle a_j, q_j\rangle \in \langle\langle q\rangle\rangle$. This proof will be successful only if $i = j$ and $\exists W$ is instantiated with w_i (using the $\exists$ right introduction rule). Thus, the above sequents would then become

$$a_1 w_1, w_1 \multimap p_1 \longrightarrow a_1 w_1 \otimes (w_1 \multimap q_1) \;\ldots\; a_m w_m, w_m \multimap p_m \longrightarrow a_m w_m \otimes (w_m \multimap q_m).$$

Finally, using the right introduction rules for $\otimes$ and $\multimap$ and the left introduction rule for $\multimap$, these sequents reduce to

$$p_1 \longrightarrow q_1 \quad \ldots \quad p_m \longrightarrow q_m.$$

Since these proof steps are forced, these several inference rules can be arranged into the following derived rule of inference

$$\frac{p_1 \longrightarrow q_1 \quad \ldots \quad p_m \longrightarrow q_m}{p \longrightarrow q}\; SIM_1$$

provided that $m \geq 0$, $\{(a_1, p_1), \ldots, (a_m, p_m)\} = \langle\langle p\rangle\rangle$, and

$$\{(a_1, q_1), \ldots, (a_m, q_m)\} \subseteq \langle\langle q\rangle\rangle.$$

Let ${}_{SIM_1} \vdash p \longrightarrow q$ denote the proposition that the sequent $p \longrightarrow q$ can be proved using only the SIM_1 inference rule (again, for the noetherian case; for the non-noetherian case, see Section 4). We have now provided the proof outline of the following proposition.

Proposition 3.1 *Let $\langle \Lambda, S, \delta\rangle$ be a finite, noetherian ats and let $p, q \in S$. Then $ats_1(\delta) \Vdash p \longrightarrow q$ if and only if ${}_{SIM_1} \vdash p \longrightarrow q$.*

Notice that in the SIM_1 proof system, we do not need instances of the initial sequent rule: a proof of $p \longrightarrow p$ using only the SIM_1 rule is always possible for a noetherian ats. This observation is similar to the one that holds of most proof systems: the initial rule is needed to prove $A \longrightarrow A$ only when A is atomic; that is, when A has a non-logical symbol as its head symbol. When using $ats_1(\delta)$, states become logical constants, in a sense, and, hence, we do not need any instance of the initial rule. We can now establish our first proof theoretic connection to simulation.

Proposition 3.2 *Let $\langle \Lambda, S, \delta\rangle$ be a finite, noetherian ats and let $p, q \in S$. Then $ats_1(\delta) \Vdash p \longrightarrow q$ if and only if q simulates p.*

Proof Given Proposition 3.1, we need only show that ${}_{SIM_1} \vdash p \longrightarrow q$ if and only if q simulates p. First, assume that the sequent $p \longrightarrow q$ has a proof that contains only the SIM_1 inference rule. Let $\mathcal{R}$ be the set of all pairs $\langle r, s\rangle$ such that the sequent $r \longrightarrow s$ has an occurrence in that proof. It is an easy matter to verify that $\mathcal{R}$ is a simulation.

Conversely, assume that q simulates p. Thus there is a simulation $\mathcal{R}$ such that $p\mathcal{R}q$. We construct a proof tree which has the property that, for every sequent $r \longrightarrow s$ in the proof, $r\mathcal{R}s$. We begin with a single node labeled with the sequent $p \longrightarrow q$. We then repeat the following as long as there is a sequent $r \longrightarrow s$ in the tree that is not the conclusion of an inference rule. By construction, $r\mathcal{R}s$, so $\mathcal{R}$ contains pairs $\langle r_1, s_1\rangle, \ldots, \langle r_m, s_m\rangle$, where $\{(a_1, r_1), \ldots, (a_m, r_m)\} = \langle\langle r\rangle\rangle$ and $\{(a_1, s_1), \ldots, (a_m, s_m)\} \subseteq \langle\langle s\rangle\rangle$ for some actions $a_1, \ldots, a_m$ ($m \geq 0$). Now the sequent $r \longrightarrow s$ can be placed at the conclusion of a SIM_1 rule, whose premises are $r_1 \longrightarrow s_1, \ldots, r_m \longrightarrow s_m$. This process terminates since the given ats is noetherian and has a bound on the length of its paths. ∎

A consequence of this theorem is that logical equivalence of two processes, namely $(p \multimap q) \mathbin{\&} (q \multimap p)$, which is the finest equivalence relation definable using logic, is "two-way simulation". Logical equivalence, however, is courser than bisimulation, as the following example shows.

Example 3.3 *Consider the transition system with one label a, states $\{p_1, p_2, p_3, p_4, q_1, q_2, q_3\}$, and transitions $p_1 \xrightarrow{a} p_2, p_2 \xrightarrow{a} p_3, p_1 \xrightarrow{a} p_4, q_1 \xrightarrow{a} q_2$, $q_2 \xrightarrow{a} q_3$. The relations*

$$\{\langle p_1, q_1\rangle, \langle p_2, q_2\rangle, \langle p_3, q_3\rangle, \langle p_4, q_2\rangle\} \text{ and } \{\langle q_1, p_1\rangle, \langle q_2, p_2\rangle, \langle q_3, p_3\rangle\}$$

witness the fact that q_1 simulates p_1 and p_1 simulates q_1, respectively. It is easy to check, however, that there is no bisimulation that contains the pair $\langle p_1, q_1\rangle$.

Using the encoding $ats_1(\delta)$, simulation can be identified with the logical connective $\multimap$. As a consequence of this example, bisimulation cannot in general be a logical connective: that is, it cannot be a predicate (of type $o \to o \to o$) with left and right introduction rules that enjoy cut-elimination. However, if we consider an ats that is also determinate then $p \sqsubseteq q$ and $q \sqsubseteq p$ imply $p \equiv q$, so logical equivalence and bisimulation coincide.

Proposition 3.4 *Let $\langle \Lambda, S, \delta\rangle$ be a finite, noetherian, and determinate ats and let $p, q \in S$. Then $ats_1(\delta) \Vdash (p \multimap q) \mathbin{\&} (q \multimap p)$ if and only if p is bisimilar to q.*

If we switch representations of transitions systems to use the definition $ats_2(\delta)$, then it is possible to characterize simulation and bisimulation as predicates *sim* and *bisim* given by the following definition, named *sims*.

$$\begin{array}{l}
\forall P, Q\ [sim(P,Q) \circ\!\!- \forall A \forall P'.\ one(P,A,P') \multimap \\
\qquad\qquad\qquad\qquad \exists Q'.\ one(Q,A,Q') \otimes sim(P',Q')] \\
\forall P, Q\ [bisim(P,Q) \circ\!\!- [\forall A \forall P'.\ one(P,A,P') \multimap \\
\qquad\qquad\qquad\qquad \exists Q'.\ one(Q,A,Q') \otimes bisim(P',Q')]\ \& \\
\qquad\qquad\quad [\forall A \forall Q'.\ one(Q,A,Q') \multimap \\
\qquad\qquad\qquad\qquad \exists P'.\ one(P,A,P') \otimes bisim(Q',P')]].
\end{array}$$

We introduce an inference rule similar to the SIM_1 given above, namely

$$\frac{\longrightarrow sim(p_1, q_1) \quad \ldots \quad \longrightarrow sim(p_m, q_m)}{\longrightarrow sim(p, q)}\ SIM_2$$

provided that $m \geq 0$,

$$\{(a_1, p_1), \ldots, (a_m, p_m)\} = \langle\langle p \rangle\rangle, \text{ and } \{(a_1, q_1), \ldots, (a_m, q_m)\} \subseteq \langle\langle q \rangle\rangle.$$

We also present an inference rule $BISIM_2$ for bisimulation:

$$\frac{\longrightarrow bisim(p_1, q_1) \quad \longrightarrow bisim(q_1, p_1) \quad \ldots \longrightarrow bisim(p_m, q_m) \quad \longrightarrow bisim(q_m, p_m)}{\longrightarrow bisim(p, q)}$$

where $m \geq 0$,

$$\{(a_1, p_1), \ldots, (a_m, p_m)\} = \langle\langle p \rangle\rangle, \text{ and } \{(a_1, q_1), \ldots, (a_m, q_m)\} = \langle\langle q \rangle\rangle.$$

Let ${}_{SIM_2} \vdash \Delta \longrightarrow C$ (respectively, ${}_{BISIM_2} \vdash \Delta \longrightarrow C$) denote the proposition that the sequent $\Delta \longrightarrow C$ can be proved using only the SIM_2 (respectively, $BISIM_2$) inference rule.

Proposition 3.5 *Let $\langle \Lambda, S, \delta \rangle$ be a finite, noetherian ats and let $p, q \in S$. Then*

- $ats_2(\delta), sims \Vdash sim(p, q)$ *if and only if* ${}_{SIM_2} \vdash sim(p, q)$*, and*
- $ats_2(\delta), sims \Vdash bisim(p, q)$ *if and only if* ${}_{BISIM_2} \vdash bisim(p, q)$*.*

Proof We outline the proof of the first case; the second can be done similarly. Consider a proof of the sequent $\longrightarrow sim(p, q)$. This is provable only by a BC rule using *sims*, and thus the sequents

$$\longrightarrow \forall A \forall P'.\ one(p, A, P') \multimap \exists Q'.\ one(q, A, Q') \otimes sim(P', Q')$$

and

$$one(p, A, P') \longrightarrow \exists Q'.\ one(q, A, Q') \otimes sim(P', Q')$$

must be provable. If this latter sequent is provable, there is a proof of it ending with DR, and thus the sequents

$$\longrightarrow \exists Q'.\ one(q, a_1, Q') \otimes sim(p_1, Q') \quad \ldots \quad \longrightarrow \exists Q'.\ one(q, a_m, Q') \otimes sim(p_m, Q')$$

must be provable, where $\langle\langle p \rangle\rangle = \{(a_1, p_1), \ldots, (a_m, p_m)\}$. Each of these sequents is provable only if each of the $\exists Q'$ is instantiated with q_i where $(a_i, q_i) \in \langle\langle q \rangle\rangle$, for $i = 1, \ldots, m$. Thus, these sequents are provable only if the sequents

$$\longrightarrow sim(p_1, q_1) \quad \ldots \quad \longrightarrow sim(p_m, q_m)$$

are provable. ∎

Now the following can be proved using Proposition 3.5 in a manner analogous to the proof of Proposition 3.2 using Proposition 3.1.

Proposition 3.6 *Let $\langle \Lambda, S, \delta \rangle$ be a finite, noetherian ats, and let p and q be members of S. Then*

- $ats_2(\delta), sims \Vdash sim(p, q)$ *if and only if* $p \sqsubseteq q$*, and*
- $ats_2(\delta), sims \Vdash bisim(p, q)$ *if and only if* $p \equiv q$*.*

The following proposition can be proved similarly to the preceding proposition, although it is a bit more involved, owing to the fact that the definition of one-step transitions in CCS is given by recursion. Notice that CCS over a finite set of actions and restricted to finite processes is not a finite ats.

Proposition 3.7 *The following equivalences hold for finite processes p and q of CCS.*

- $ccs(\mathcal{A}), sims \Vdash sim(p,q)$ *if and only if* $p \sqsubseteq q$.
- $ccs(\mathcal{A}), sims \Vdash bisim(p,q)$ *if and only if* $p \equiv q$.

4 Non-finite behavior cases

In the previous section we have expressed simulation and bisimulation in logic using recursive definitions of the form

$$\forall P_1 \ldots \forall P_n [r(P_1, \ldots, P_n) \mathrel{\circ\!\!-} \Phi], \tag{1}$$

where the formula Φ contained the predicate r and free occurrences of the variables $P_1, \ldots, P_n$. To this clause we associate a function ϕ from n-ary relations to n-ary relations defined as: $\phi(\mathcal{R})$ is the set of all tuples of terms $\langle t^1, \ldots, t^n \rangle$ such that there is a set $\{\langle t_1^1, \ldots, t_1^n \rangle, \ldots, \langle t_m^1, \ldots, t_m^n \rangle\} \subseteq \mathcal{R}$ $(m \geq 0)$ and

$$ats_2(\delta) \Vdash r(t_1^1, \ldots, t_1^n) \,\&\, \ldots \,\&\, r(t_m^1, \ldots, t_m^n) \longrightarrow \Phi[t^1/P_1, \ldots, t^n/P_n].$$

To see that definitional reflection on (1) is sound for all the relations which are fixed points of ϕ, assume that for any relation r there is only one such definition (in case there are more, we group them in one definition which has as body the disjunction of the bodies). Then observe that the use of definitional reflection corresponds to reversing the implication in the definition, that is, to assuming the formula $\forall P_1 \ldots \forall P_n [r(P_1, \ldots, P_n) \multimap \Phi]$.

Let Φ_s and Φ_b be the bodies of the definitions given in $sims$ for sim and $bisim$, respectively, and consider the corresponding functions ϕ_s and ϕ_b on binary relations associated with these formulas. We can see from their definitions that $\sqsubseteq$ and $\equiv$ are the greatest fixed points of ϕ_s and ϕ_b, respectively. Note that in proofs of $\longrightarrow sim(p,q)$ and $\longrightarrow bisim(p,q)$ using definitions $ats_2(\delta)$ and $sims$, DR is used with $ats_2(\delta)$ but not with $sims$. Backchaining on (1) is all that we need to prove the sequent $\longrightarrow r(p_1, \ldots, p_n)$ whenever $p_1, \ldots, p_n$ are related in the least fixed point of the function ϕ.

As the following example shows, when the transition system is not noetherian, the "if parts" of Proposition 3.6 may not hold.

Example 4.1 *Consider a transition system with two states only, p and q, and two transitions $p \xrightarrow{a} p$ and $q \xrightarrow{a} q$. Then $p \sqsubseteq q$ holds, but $sim(p,q)$ cannot be proved. Note that the attempt to prove it would end up in a circularity.*

In fact, both backchaining and definitional reflection on (1) are sound for all the relations which are fixed points of ϕ. If $ats_2(\delta), sims \Vdash r(p_1, \ldots, p_n)$ holds then $p_1, \ldots, p_n$ must be related by every fixed point of ϕ, and thus by its least fixed point. In a noetherian ats, ϕ_s and ϕ_b each have unique fixed points, and this is why $\sqsubseteq$ and $\equiv$ can be completely characterized in a noetherian ats by provability (Proposition 3.6).

One attempt to characterize the greatest fixed point of the relation transformer ϕ proof-theoretically is to introduce a notion of "finite or infinite proof". An *ω-proof* of the sequent $\Delta \longrightarrow C$ with inference rules taken from the set L is a tree whose root is labeled by $\Delta \longrightarrow C$, and such that for every node N there is an instance of an inference rule of L whose conclusion is the label of N, and whose premises are the labels of the children of N. We will denote by ${}_L \vdash_\omega \Delta \longrightarrow C$ the existence of an ω-proof in L for $\Delta \longrightarrow C$. For example, ${}_{SIM_2} \vdash_\omega \Delta \longrightarrow C$ is true if $\Delta \longrightarrow C$ has an ω-proof using only SIM_2. If the set of inference rules L is determined by those in linear logic and those arising from using some definition, say D, then we write $D \Vdash_\omega \Delta \longrightarrow C$. Notice that an ω-proof can have bounded or unbounded height. We now prove two propositions using ω-proofs that generalize Propositions 3.5 and 3.6 by dropping the noetherian condition.

Proposition 4.2 *Let $\langle \Lambda, S, \delta \rangle$ be a finite ats and let $p, q \in S$. Then*

- $ats_2(\delta), sims \Vdash_\omega sim(p,q)$ *if and only if* ${}_{SIM_2} \vdash_\omega sim(p,q)$*, and*
- $ats_2(\delta), sims \Vdash_\omega bisim(p,q)$ *if and only if* ${}_{BISIM_2} \vdash_\omega bisim(p,q)$*.*

Proof We outline the proof of the first case; the second can be done similarly. Since the converse is immediate, we only show the forward direction. Assume that the sequent $\longrightarrow sim(p,q)$ has an ω-proof using the definitions $ats_2(\delta)$. Since this sequent must be proved by one use of BC, two uses of $\forall$R, and one use of $\multimap$R, we have

$$ats_2(\delta) \Vdash_\omega one(p, A, P') \longrightarrow \exists q'[one(q, A, q') \otimes sim(P', q')],$$

where A and P' are variables. At this point, the proof can proceed by either DR or $\exists$R. If the choice is DR, then we quickly get that the proof is essentially an instance of SIM_2 at the root, and we proceed recursively through the ω-proof. Otherwise, a use of $\exists$R would give raise to a tensor, the first component of which is $one(q, A, q_0)$ for some particular $q_0 \in S$. It is not possible, however, to prove this atom using BC since no instance of a clause in the definition $ats_1(\delta)$ has the variable A in its head. Thus, this proof could not be built in that fashion. ∎

Proposition 4.3 *Let $\langle \Lambda, S, \delta \rangle$ be a finite ats. Then $p \sqsubseteq q$ if and only if $ats_2(\delta), sims \Vdash_\omega sim(p,q)$, and $p \equiv q$ if and only if $ats_2(\delta), sims \Vdash_\omega bisim(p,q)$.*

Proof We prove only the first equivalence since the second follows similarly. First, assume that

$$ats_2(\delta), sims \Vdash_\omega sim(p,q).$$

By the preceding Proposition, ${}_{SIM_2} \vdash_\omega sim(p,q)$. We now proceed as in the proof of Proposition 3.2: Let Ξ be a proof of $\longrightarrow sim(p,q)$ that contains just the SIM_2 inference rule and let $\mathcal{R}$ be the binary relation such that $r\mathcal{R}s$ if $r \longrightarrow s$ has an occurrence in Ξ. It is easy to see that $\mathcal{R}$ is a simulation containing $\langle p, q \rangle$.

Assume next that $p \sqsubseteq q$ holds. We can construct a sequence of trees $\{T_k\}_{k \in \omega}$ such that for every k, the root of T_k is labeled by $\longrightarrow sim(p, q)$, all the leaves of T_k are labeled by sequents of the form $sim(p', q')$ where $p' \sqsubseteq q'$ holds, and T_k is a subtree of T_{k+1}. The construction process for such trees follows the same pattern as used in the proof of Proposition 3.2. ∎

For these equivalences to hold, it is important that the ats is finite. If we consider a non-finite ats, say, $ccs(\mathcal{A})$ ($\mathcal{A}$ is finite), then the reverse direction of these equivalences does not necessarily hold, as the following example illustrates.

Example 4.4 *The CCS terms $\mu_x x$ and $\mu_x a.x$ are such that $\mu_x a.x \sqsubseteq \mu_x x$ does not hold. However, both the judgments $ccs(\mathcal{A}) \vdash_\omega one(\mu_x x, a, \mu_x a.x)$, and $ccs(\mathcal{A}), sims \Vdash_\omega sim(\mu_x a.x, \mu_x x)$ hold.*

In a sense, the infinite proof of $one(\mu_x x, a, \mu_x a.x)$ is an "infinite failure". If we restrict to finite CCS processes, then such infinite failures do not occur with respect to the one step transition steps and the only infinite proof behaviors will be those that positively verify the greatest fixed point properties of simulation and bisimulation. Thus, when restricted to finite CCS processes, the generalization of Proposition 4.3 where $ats_2(\delta)$ is replaced with $ccs(\mathcal{A})$ holds.

5 Properties about defined relations

In previous sections we have shown how to encode in sequent calculus various relations over the states of a transition system. We now explore the kinds of properties on the relations that can be proved within the calculus or via some characterization provided by the calculus.

Example 5.1 *The property "bisimulation is preserved by the prefix operator" holds in CCS. The corresponding encoding of this property is also provable in the sequent calculus; that is, we have*

$$ccs(\mathcal{A}), sims \Vdash \forall P \forall Q[bisim(P, Q) \multimap bisim(A.P, A.Q)],$$

a proof of which is easy to construct. The property "bisimulation is symmetric" holds in any transition system. The corresponding encoding of this property is also provable in the sequent calculus; that is, we have

$$ats_2(\delta), sims \Vdash \forall P \forall Q[bisim(P, Q) \multimap bisim(Q, P)],$$

which is also easy to verify.

However, as the following examples illustrate, there are plenty of true properties of $\equiv$ and $\sqsubseteq$ that cannot be proved within the logic. One reason for this lack is, intuitively, we can prove properties of *sim* and *bisim* only if they are true for every fixed point of ϕ_s and ϕ_b, but in the non-noetherian case, there is in general more than one fixed point.

Example 5.2 *The property "bisimulation equivalence implies the largest simulation" (or more formally: $\equiv$ is a subset of $\sqsubseteq$) is true in any transition system. This property can be expressed by the formula* $\forall P \forall Q[\mathrm{bisim}(P,Q) \multimap \mathrm{sim}(P,Q)]$ *but, in general, if δ is a non-noetherian transition relation, this formula cannot be proved using the definitions* $\mathrm{ats}_2(\delta)$ *and* sims. *For example, if we take the transition system* $\langle \{a\}, \{p\}, \{\langle p,a,p\rangle\}\rangle$ *it is immediate to see that* $\{\langle p,p\rangle\}$ *is a bisimulation (the greatest fixed point of ϕ_b, namely bisimulation equivalence) and $\emptyset$ is a simulation (the least fixed point of ϕ_s). Hence, this formula cannot be proved for this transition system.*

Example 5.3 *The property "bisimulation equivalence is reflexive" holds in any transition system. The formula* $\forall P[\mathrm{bisim}(P,P)]$ *cannot be proved using the definitions* $\mathrm{ats}_2(\delta)$ *and* sims. *Consider for instance the same transition system as in Example 5.2: the empty set $\emptyset$ is a bisimulation (the least fixed point of ϕ_b), and it is, of course, not reflexive.*

Example 5.4 *The property "bisimulation equivalence is preserved by the + operator" is true in CCS. This property can be expressed as the formula* $\forall P \forall Q \forall R[\mathit{bisim}(P,Q) \multimap \mathit{bisim}(P+R, Q+R))]$. *This sequent cannot be proved using* $\mathit{ccs}(\mathcal{A})$ *and sims. In fact, take* $P = a.0$, $Q = a.0 + a.0$ *and* $R = \mu_x a.x$. *The least fixed point of ϕ_b contains the pair* $\langle a.0, a.0 + a.0\rangle$ *but not the pair* $\langle a.0 + \mu_x a.x, a.0 + a.0 + \mu_x a.x\rangle$.

The notion of "infinite proof", introduced in the previous section, can be helpful to prove properties on the defined relations at the meta (mathematical) level. For instance, the properties of Examples 5.2 and 5.3 can both be proved by using the characterization of $\equiv$ and $\sqsubseteq$ provided at the end of the previous section. It is easy to see, in fact, that $\mathit{ccs}(\mathcal{A}), \mathit{sims} \vdash_\omega \mathit{bisim}(P,P)$. Concerning the implication $\mathit{bisim}(P,Q) \multimap \mathit{sim}(P,Q)$, observe that any infinite proof for $\longrightarrow \mathit{bisim}(P,Q)$ contains an infinite proof of $\longrightarrow \mathit{sim}(P,Q)$.

The characterization of simulation or bisimulation using ω-proofs is not so helpful because the existence of an infinite proof for a given sequent is co-semidecidable but (in general) not semidecidable. A better approach would be to use co-induction: it is well known that, for finitely-branching transition systems, ϕ_s and ϕ_b are downward-continuous. Thus their greatest fixed point can be characterized as $\bigcap_k \phi_s^k(S \times S)$ and $\bigcap_k \phi_b^k(S \times S)$ respectively. We could then encode $\sqsubseteq$ and $\equiv$ by using stratified versions of *sim* and *bisim*: Figure 2 contains just such a definition, which will be named *ssims*.

In order to encode co-induction, we can use inductions on natural numbers. We incorporate induction by introducing natural numbers using z for zero and s for successor and using the predicate *nat*. We introduce now the following "introduction" rules for this new predicate.

$$\frac{}{\longrightarrow \mathit{nat}\ z} \qquad \frac{\Delta \longrightarrow \mathit{nat}\ x}{\Delta \longrightarrow \mathit{nat}\ (sx)} \qquad \frac{\Delta_1 \longrightarrow Qz \quad Qy \longrightarrow Q(sy) \quad \Delta_2, Qx \longrightarrow P}{\Delta_1, \Delta_2, \mathit{nat}\ x \longrightarrow P}$$

Here, x, P, and Q are schematic variables of these inference rule, and y is a variable not free in Q. The first two rules can be seen as right-introduction rules for *nat* while the third rule, encoding induction over natural numbers, can

$$
\begin{aligned}
sim(P,Q) &\mathrel{\circ\!\!-} \forall K\,(nat\ K \multimap ssim(K,P,Q)).\\
ssim(z,P,Q) &\mathrel{\circ\!\!-} 1\\
ssim((sK),P,Q) &\mathrel{\circ\!\!-} [\forall A \forall P'.\ one(P,A,P') \multimap\\
&\qquad\qquad \exists Q'.\ one(Q,A,Q') \otimes ssim(K,P',Q')].\\
bisim(P,Q) &\mathrel{\circ\!\!-} \forall K\,(nat\ K \multimap sbisim(K,P,Q)).\\
sbisim(z,P,Q) &\mathrel{\circ\!\!-} 1.\\
sbisim((sK),P,Q) &\mathrel{\circ\!\!-} [\forall A \forall P'.\ one(P,A,P') \multimap\\
&\qquad\qquad \exists Q'.\ one(Q,A,Q') \otimes sbisim(K,P',Q')]\ \&\\
&\qquad [\forall A \forall Q'.\ one(Q,A,Q') \multimap\\
&\qquad\qquad \exists P'.\ one(P,A,P') \otimes sbisim(K,Q',P')].
\end{aligned}
$$

Fig. 2. Stratified simulation and bisimulation definitions. Free variables are assumed to be universally quantified.

be seen as a left-introduction rule. In the left-introduction rule, Q ranges over formulas with one variable extracted (say, using λ-abstraction) and represents the property that is proved by induction: the third premise of that inference rule witnesses the fact that, in general, Q will express a property stronger than P. Notice that with this formulation of induction, cut-free proofs will not have the subformula property.

The following inference rules can be derived from these rules for nat.

$$
\frac{\Delta \longrightarrow Qz \quad Qy \longrightarrow Q(sy)}{\Delta, nat\ n \longrightarrow Qn}
\qquad
\frac{\Delta \longrightarrow C}{\Delta, nat\ n \longrightarrow C}
\qquad
\frac{\Delta, nat\ n, nat\ n \longrightarrow C}{\Delta, nat\ n \longrightarrow C}
$$

The fact that a nat-atom can be weakened and contracted on the left follows from observation that it is possible to prove $\forall n(nat\ n \multimap !\,nat\ n)$ using our induction principle.

We can now prove properties of maximal simulation or bisimulation equivalence even if they are not valid for all fixed points of ϕ_s or ϕ_b; the only necessary condition is that they are valid on $\bigcap_k \phi_s^k(S \times S)$ or $\bigcap_k \phi_b^k(S \times S)$. Of course, this encoding of $\sqsubseteq$ and $\equiv$ as sim and $bisim$ is sound only when ϕ_s and ϕ_b are downward-continuous; in general this is true when the transition system is finitely branching. In CCS, for instance, this condition is guaranteed whenever all the recursion variables in μ-expressions are prefixed.

It is now possible to prove, for instance, reflexivity of bisimulation equivalence, and bisimilarity of nodes involved in cyclic transitions. In particular, we can prove all of the following formulas using natural number induction and definitional reflection with the definitions $ccs(\mathcal{A})$ and $ssims$.

$$
\begin{gathered}
bisim(\mu_x\ a.x, \mu_x\ (a.x + a.x)) \qquad\qquad \forall P\ bisim(P,P)\\
\forall P\ bisim(P+0,P) \qquad\qquad \forall P\ bisim(P+P,P)\\
\forall P,Q\ (bisim(P,Q)) \multimap sim(P,Q))\\
\forall P,Q,R\ (bisim(P,Q) \multimap bisim(P+R,Q+R))
\end{gathered}
$$

We are continuing to consider the use of induction along with definitions to capture judgments about properties such as bisimulation.

6 Conclusion

It has been observed before and in this paper that linear logic can be used to specify transition systems. We have shown that if linear logic is extended with definitions, then certain properties about elements of transition systems, namely simulation and bisimulation, can be captured naturally. Furthermore, if induction over integers is added, then we can increase the expressiveness of logic to establish more high-level facts about these properties, such as the fact that bisimulation is an equivalence.

From a high-level point-of-view, we can characterize the experiments we have reported here in two ways. From a (traditional) logic programming point of view, a definition D is generally either a set of (positive) Horn clauses or an extension of them that allows negated atoms in the body of clauses. In that case, sequents in a proof of $D \Vdash A$, for atomic formula A, are either of the form $\longrightarrow B$ or $B \longrightarrow$. In the first case, backchaining is used to establish B and, in the second case, definition reflection is used to build a finite refutation of B. In this paper, we consider richer definitions so that the search for proofs must consider sequents of the form $B \longrightarrow C$; with such sequents, backchaining and definition reflection are used together. From a computational or concurrency point-of-view, proofs using just backchaining only capture the *may* behavior of a system: "there exists a computation such that ..." is easily translated to "there exists a proof (in the sense of $\vdash$) of ...". The addition of the definitional reflection inference rules allows certain forms of *must* behavior to be captured.

Acknowledgments.

We would like to thank Robert Stärk for several helpful discussions and Chuck Liang for providing us with a prototype in λProlog of an interactive theorem prover for doing many of the formal (object-logic) proofs described in this paper. The authors have been funded in part by ONR N00014-93-1-1324, NSF CCR-92-09224, NSF CCR-94-00907, and ARO DAAH04-95-1-0092. Part of this work was done while Palamidessi was visiting the University of Pennsylvania and while Miller was visiting the University of Genova. We both wish to thank these institutions for their hospitality in hosting us.

References

[1] M. Aronsson, L.-H. Eriksson, A. Gäredal, L. Halnäs, and P. Olin. GCLA: a definitional approach to logic programming. *New Generation Computing*, 4:381–404, 1990.

[2] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[3] Vijay Gehlot and Carl Gunter. Normal process representatives. In *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 200–207, Philadelphia, Pennsylvania, June 1990. IEEE Computer Society Press.

[4] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[5] Jean-Yves Girard. A fixpoint theorem in linear logic. A message posted on the linear@cs.stanford.edu mailing listing, `http://www.csl.sri.com/linear/mailing-list-traffic/www/07/mail_3.html`, February 1992.

[6] Lars Hallnäs and Peter Schroeder-Heister. A proof-theoretic approach to logic programming. ii. Programs as definitions. *Journal of Logic and Computation*, pages 635–660, October 1991.

[7] Dale Miller. The π-calculus as a theory in linear logic: Preliminary results. In E. Lamma and P. Mello, editors, *Proceedings of the 1992 Workshop on Extensions to Logic Programming*, number 660 in LNCS, pages 242–265. Springer-Verlag, 1993.

[8] Robin Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[9] Peter Schroeder-Heister. Rules of definitional reflection. In M. Vardi, editor, *Eighth Annual Symposium on Logic in Computer Science*, pages 222–232. IEEE, June 1993.

[10] Robert Stärk. A complete axiomatization of the three-valued completion of logic programs. *Journal of Logic and Computation*, 1(6):811–834, 1991.