Encoding Transition Systems in Sequent Calculus

Raymond McDowell
Department of Mathematics and Computer Science
Kalamazoo College
1200 Academy Street
Kalamazoo, MI 49006-3295 USA

Dale Miller and **Catuscia Palamidessi**
Department of Computer Science and Engineering
220 Pond Laboratory
The Pennsylvania State University
University Park, PA 16802-6106 USA

19 July 1999

Abstract

Intuitionistic and linear logics can be used to specify the operational semantics of transition systems in various ways. We consider here two encodings: one uses linear logic and maps states of the transition system into formulas, and the other uses intuitionistic logic and maps states into terms. In both cases, it is possible to relate transition paths to proofs in sequent calculus. In neither encoding, however, does it seem possible to capture properties, such as simulation and bisimulation, that need to consider all possible transitions or all possible computation paths. We consider augmenting both intuitionistic and linear logics with a proof theoretical treatment of definitions. In both cases, this addition allows proving various judgments concerning simulation and bisimulation (especially for noetherian transition systems). We also explore the use of infinite proofs to reason about infinite sequences of transitions. Finally, combining definitions and induction into sequent calculus proofs makes it possible to reason more richly about properties of transition systems completely within the formal setting of sequent calculus.

Keywords: Transition systems, definitions, logic specification, bisimulation, linear logic.

This paper is to appear in *Theoretical Computer Science.*

1 Introduction

Structural operational semantics [30] or natural semantics [19] are familiar approaches to the specification of computations. These semantic specifications can generally be encoded into logic so that atomic formulas are used to encode judgments about the computation (such as "program M has value V" or "process P has a c transition to process Q") and formulas are used to encode inference rules. Such encodings generally yield first-order Horn clauses, although richer sets of clauses have also been used, such as subsets of intuitionistic logic with quantification at higher-types (like those implemented in Isabelle [29] and λProlog [28]). These encodings have been successfully used in the specification of a wide range of computations, including the evaluation of functional programming languages [4, 15, 27], abstract machines [16], and process calculi [26]. Various recent papers suggest that linear logic [11] can be used as well to make this style of specification more expressive. For example, specifications of imperative and concurrent programming language features [5, 6, 9, 23, 24] and the sequential and concurrent (pipe-line) semantics of a RISC processor [6] have been modeled using linear logic.

A key property of such encodings is that there exists a computation in a certain system if and only if there is a proof of a certain judgment from the set of clauses that encodes that computation system. If we were interested in capturing properties based on all possible transitions or computations, the conventional proof theory techniques for intuitionistic and linear logic do not appear to suffice since they do not provide a direct way to manage the notion of "for all proofs" within the logic itself. Using terminology from Hennessy [17], conventional uses of proof theory provide means to capture *may* behavior but not *must* behavior of a transition system.

In this paper we investigate the use of a recently studied extension to proof systems that will allow us to capture certain kinds of *must* behaviors. This extension is based on the notion of *definition*. While definitions and theories are similar in that they both contain formulas useful for constructing a proof, the notion of definition carries the added intention that there are no other ways by which the defined concept can be established. Such a notion of definition has been investigated in proof systems in recent years: Hallnäs considered similar notions in the context of "partial inductive definitions" [13]; Hallnäs and Schroeder-Heister considered classical and intuitionistic logic with proof rules incorporating definitions [14, 31], and in the note [12], Girard independently developed a similar use of definitions for linear logic. More recently, McDowell and Miller have incorporated definitions into an intuitionistic proof system that also includes natural number induction [21, 22]. In all of these cases, it can be shown that if certain restrictions are placed on the structure of definitions, 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 [12, 31], forms of program completion in logic programming [14, 32], the GCLA language project [3], and for meta-level reasoning about logical inference [21].

To provide a general setting for our analysis of the use of definitions within proofs, we shall work with *abstract transition systems* (ats) instead of structural operational semantics or natural semantics, since these can be seen as special cases of abstract transition systems. Section 2 provides some background on the proof theory notions that we shall need, including the concept of definitions. Section 3 contains background notions surrounding ats's. We then consider two ways to encode an ats into logic. In Section 4, they are encoded directly into linear logic in such a way that atomic formulas denote members of the ats and the rules of linear logic are used to mimic actual transitions. Section 5 considers a different approach to encoding an ats: there, members of an ats are encoded as terms and the basic judgment about labeled transitions is represented by an atomic formula. In both of these encodings, the use of definitions allows such *must* behaviors as simulation to be captured for noetherian ats's. Section 6 briefly considers using infinite proofs to deal with non-noetherian transition systems. Finally, in Section 7, we consider adding induction and discuss how that allows establishing richer properties of ats's.

2 Adding definitions to sequent calculus

A *definition* is a countable set of *clauses*, and clauses are expressions of the form $\forall\bar{x}[p(\bar{t}) \triangleq B]$, where the free variables of B are free in some term of the list of terms $\bar{t}$, and all variables free in (some term in) $\bar{t}$ are contained in the list of variables $\bar{x}$. The formula B is the *body* and $p(\bar{t})$ is the *head* of that clause. We do not assume that distinct clauses have distinct predicates p in their head: definitions act to define predicates by mutual recursion. The symbol $\triangleq$ is not a logical connective: it is used just to denote definitional clauses. In Section 4 we shall consider the addition of definitions to linear logic and in Section 5 through the end of the paper we consider adding definitions to intuitionistic logic. In either case, we shall assume that our logic is typed, say, in the style of the Church's Simple Theory of Types [7]. Quantification over predicates will not be used in this paper; we shall assume that formulas have type o, and other types will be introduced when needed. We shall generally treat types implicitly, but shall include types in examples to help make the specifications more readable.

We shall assume that the reader is familiar with the usual two-sided sequent calculus presentation of intuitionistic logic [10] and linear logic [11]. In this paper, we consider sequents of the form $\Delta \longrightarrow B$, where B is a formula and Δ is either a multiset or a set of formulas, depending on whether we are working in linear or intuitionistic logic. We shall also assume that the reader is familiar with substitutions and their basic properties. Given two expressions A and A', a *unifier* for A and A' is a substitution θ such that $A\theta = A'\theta$. Unifiers may map variables to open terms. The set of unifiers for A and A' is denoted as $\mathit{unif}(A, A')$. For the sake of generality, we will assume an underlying equality theory, and $A\theta = A'\theta$ here means that $A\theta$ and $A'\theta$ are equal in that equality theory. (In general, this will be the intended meaning of $=$ whenever written between two terms or two atomic formulas.) A *complete set of unifiers* (csu) for A and A' is a set S of substitutions such that (i) for each $\theta \in S$, θ is a unifier of A and A', and (ii) for every unifier σ of A and A', there exists a $\theta \in S$ which is more general than σ; namely, there exists a substitution ρ such that $\theta\rho = \sigma$. In general, there can be many different csu's for A and A'. We use the function $\mathit{csu}(A, A')$ to pick one of these csu's in an arbitrary but fixed fashion. In some cases, like when the equality is syntactic identity, there exists a most general unifier and, therefore, the csu can be taken to be a singleton. All the examples in this paper fall into this category except for some examples in Sections 5 and 7, where the α, β, η-equality theory of simply typed λ-terms is used: in that theory, it is possible that two unifiable terms have no singleton csu [18].

Left and right introduction rules for atomic formulas will be given for a fixed definition and equality theory. The right rule is given as

$$\frac{\Delta \longrightarrow B\theta}{\Delta \longrightarrow A}\ \mathit{def}\mathcal{R}, \text{ where } A = H\theta \text{ for some clause } \forall\bar{x}.[H \triangleq B].$$

If we think of a definition as a logic program, then this rule is essentially the same as *backchaining*.

Left introduction of defined atoms requires considering unifiers between two atoms. We consider various formulations of this rule. The first version [31] is given as

$$\frac{\{B\theta, \Delta\theta \longrightarrow C\theta \mid \theta \in \mathit{unif}(A, H) \text{ for some clause } \forall\bar{x}.[H \triangleq B]\}}{A, \Delta \longrightarrow C}\ \mathit{def}\mathcal{L}_{\mathit{unif}}.$$

The variables $\bar{x}$ in this rule need to be chosen so that they are not free in any formula of the lower sequent. Specifying a set of sequents as the premise in the left introduction rule means that each sequent in the set is a premise of the rule. This rule uses all unifiers between the defined atom A that is introduced and the heads of clauses.

Another form of the rule, using csu's, was given in [8].

$$\frac{\{B\theta, \Delta\theta \longrightarrow C\theta \mid \theta \in csu(A, H) \text{ for some clause } \forall\bar{x}.[H \triangleq B]\}}{A, \Delta \longrightarrow C}\ def\mathcal{L}.$$

Again the variables $\bar{x}$ need to be chosen so that they are not free in any formula of the lower sequent.

Notice that for both of these rules, the set of premises can be infinite since definitions and the sets $unif(A, H)$ and $csu(A, H)$ can be infinite. The advantage of the second rule is that there are situations where $csu(A, H)$ is finite while the corresponding set $unif(A, H)$ is infinite.

These two left-introduction rules for defined atoms can be proved equivalent in the following strong sense.

Proposition 1 *A proof using $def\mathcal{L}_{unif}$ can be converted to a proof using $def\mathcal{L}$ by replacing each occurrence of $def\mathcal{L}_{unif}$ with $def\mathcal{L}$ and by pruning away the proofs of unnecessary premises. Vice versa, a proof using $def\mathcal{L}$ can be converted into a proof using $def\mathcal{L}_{unif}$ by replacing each occurrence of $def\mathcal{L}$ with $def\mathcal{L}_{unif}$ and adding proofs of the additional premises.*

Proof The first part is obvious. The second part is based on the observation that the proofs for the additional premises can be built as instances of the proofs of the premises of $def\mathcal{L}$. An analogous result was proved in [20, 22]. The only difference is that in $def\mathcal{L}_{unif}$ we consider unifiers instead of pairs of substitutions σ, ρ such that $A\rho = H\sigma$. However the two formulation can be easily shown to be equivalent: we can always α-convert the variables $\bar{x}$ in $\forall\bar{x}.[H \triangleq B]$ so that $\sigma \cup \rho$ is a unifier for A and H. ∎

In this paper we will consider systems based on the $def\mathcal{L}$ rule. As a consequence of Proposition 1, it does not matter which complete set of unifiers is chosen by the function *csu* (used in the definition of $def\mathcal{L}$).

The left and right introduction rules for defined atoms have different "quantificational" interpretations when read bottom-up: $def\mathcal{R}$ replaces A with the body of *some* clause in the definition whose head matches with A, while $def\mathcal{L}$ 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 definitions.

We now show a basic result about $def\mathcal{L}$: The order in which atoms are introduced on the left does not matter. In other words, two applications of $def\mathcal{L}$ permute over each other. In order to prove this result, we first need to establish a basic property of unifiers.

Proposition 2 *Consider the expressions A_1, A_2, H_1, H_2. Let $unif((A_1, A_2), (H_1, H_2))$ be the set of unifiers of the pairs (A_1, A_2) and (H_1, H_2) (that is, the substitutions which unify simultaneously A_1 with H_1, and A_2 with H_2. We have*

$$unif((A_1, A_2), (H_1, H_2)) = \{\theta_1\theta_2 \mid \theta_1 \in unif(A_1, H_1), \theta_2 \in unif(A_2\theta_1, H_2\theta_1)\}$$

where $\theta_1\theta_2$ is the usual composition of the substitutions θ_1 and θ_2.

Proof To show the forward inclusion, let $\theta \in unif((A_1, A_2), (H_1, H_2))$. Let ε denote the empty substitution. The result follows by observing that $\theta \in unif(A_1, H_1)$, $\varepsilon \in unif(A_2\theta, H_2\theta)$, and $\theta = \theta\varepsilon$.

To show the converse inclusion, let $\theta_1 \in \mathit{unif}(A_1, H_1)$ and $\theta_2 \in \mathit{unif}(A_2\theta_1, H_2\theta_1)$. By definition of θ_2, we have $\theta_1\theta_2 \in \mathit{unif}(A_2, H_2)$. Furthermore, since equality is preserved under substitution, we have $\theta_1\theta_2 \in \mathit{unif}(A_1, H_1)$, which concludes the proof. ▮

By applying Proposition 2 two times we obtain the following corollary:

Corollary 3 *Given the expressions A_1, A_2, H_1, H_2, we have*

$$\{\theta_1\theta_2 \mid \theta_1 \in \mathit{unif}(A_1, H_1), \theta_2 \in \mathit{unif}(A_2\theta_1, H_2\theta_1)\} = \\ \{\theta_2'\theta_1' \mid \theta_2' \in \mathit{unif}(A_2, H_2), \theta_1' \in \mathit{unif}(A_1\theta_2', H_1\theta_2')\}.$$

We are now able to show that two instances of $\mathit{def}\mathcal{L}$ permute over each other.

Proposition 4 *If we have a proof of the sequent $A_1, A_2, \Delta \longrightarrow C$ using $\mathrm{def}\mathcal{L}$ with respect to A_1 as the last rule, and $\mathrm{def}\mathcal{L}$ for each of the premises of that rule with respect to (an instance of) A_2, then we also have a proof using $\mathrm{def}\mathcal{L}$ with respect to A_2 as the last rule, and $\mathrm{def}\mathcal{L}$ for each of the premises with respect to (an instance of) A_1.*

Proof Given Proposition 1, it is sufficient to prove this result for $\mathit{def}\mathcal{L}_{\mathit{unif}}$. Suppose that the hypothesis of the proposition (reformulated in terms of $\mathit{def}\mathcal{L}_{\mathit{unif}}$) holds. Then the set of premises at the second level is:

$$\{B_1\theta_1\theta_2, B_2\theta_2, \Delta\theta_1\theta_2 \longrightarrow C\theta_1\theta_2 \mid \theta_1 \in \mathit{unif}(A_1, H_1) \text{ for some clause } \forall\bar{x}.[H_1 \triangleq B_1], \text{ and} \\ \theta_2 \in \mathit{unif}(A_2\theta_1, H_2) \text{ for some clause } \forall\bar{y}.[H_2 \triangleq B_2]\}.$$

Since the list $\bar{y}$ can be chosen so as not to intersect with the domain of θ_1, we can replace H_2 with $H_2\theta_1$, and B_2 with $B_2\theta_1$. By Corollary 3 we have that the above set is equal to

$$\{B_1\theta_2'\theta_1', B_2\theta_2'\theta_1', \Delta\theta_2'\theta_1' \longrightarrow C\theta_2'\theta_1' \mid \theta_2' \in \mathit{unif}(A_2, H_2) \text{ for some clause } \forall\bar{y}.[H_2 \triangleq B_2], \text{ and} \\ \theta_1' \in \mathit{unif}(A_1\theta_2', H_1\theta_2') \text{ for some clause } \forall\bar{x}.[H_1 \triangleq B_1]\}.$$

This set of sequents corresponds to the set of premises obtained by switching the order of the last two $\mathit{def}\mathcal{L}_{\mathit{unif}}$ inference rules in the original proof. ▮

Admitting definitions in this fashion does not necessarily yield a proof system from which cut can be eliminated: to achieve a cut-elimination result, certain restrictions on definitions are needed. We present these restrictions on definitions when we deal specifically with linear logic (Section 4) and intuitionistic logic (Section 5 through the end of the paper).

If $\mathcal{D}$ is a definition, we write $\mathcal{D} \vdash \Delta \longrightarrow C$ to mean that $\Delta \longrightarrow C$ is provable in the underlying logic (intuitionistic or linear) possibly using the right introduction rule for definitions, and we write $\mathcal{D} \Vdash \Delta \longrightarrow C$ to mean that $\Delta \longrightarrow C$ is provable possibly using the left and right introduction rules for definitions. We write $\mathcal{D} \vdash B$ and $\mathcal{D} \Vdash B$ as abbreviations for $\mathcal{D} \vdash\longrightarrow B$ and $\mathcal{D} \Vdash\longrightarrow B$, respectively.

3 Abstract transition systems

The triple $\mathcal{T} = (\Lambda, S, \delta)$ 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 $(p, a, q) \in \delta$. 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

$$\frac{}{a.p \xrightarrow{a} p} \qquad \frac{p \xrightarrow{a} q}{p \mid r \xrightarrow{a} q \mid r} \qquad \frac{p \xrightarrow{a} q}{r \mid p \xrightarrow{a} r \mid q} \qquad \frac{p[\mu_x p/x] \xrightarrow{a} q}{\mu_x p \xrightarrow{a} q}$$

$$\frac{p \xrightarrow{a} q}{p + r \xrightarrow{a} q} \qquad \frac{r \xrightarrow{a} q}{p + r \xrightarrow{a} q} \qquad \frac{p \xrightarrow{a} r \quad q \xrightarrow{b} s}{p \mid q \xrightarrow{\tau} r \mid s} \text{ where } b = \bar{a} \text{ or } a = \bar{b}$$

Figure 1: CCS transition rules

$p \stackrel{\epsilon}{\Longrightarrow} p$ and if $p \xrightarrow{a} r$ and $r \stackrel{w}{\Longrightarrow} q$ then $p \stackrel{aw}{\Longrightarrow} q$. For a state p, define $\langle\langle p \rangle\rangle = \{(a, q) \mid (p, a, q) \in \delta\}$. The ats $\mathcal{T}$ is *finitely branching* if, for each p, the set $\langle\langle p \rangle\rangle$ is finite. $\mathcal{T}$ 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. $\mathcal{T}$ is *noetherian* if it contains no infinite paths. In a noetherian ats we can define the *measure* of a state p, denoted by $meas(p)$, as the ordinal number given by

$$meas(p) = \mathrm{lub}(\{meas(q) + 1 \mid p \xrightarrow{a} q \text{ for some } a\}),$$

where we assume $\mathrm{lub}(\emptyset) = 0$. If the ats is also finitely branching then all its states have finite measure.

The notions of simulation and bisimulation provide important judgments on pairs of states in an abstract transition system. A good overview of relations on transitions systems (including simulation and bisimulation) can be found in [33]. A relation $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is a *simulation* 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 $p'\mathcal{R}q'$. The union of all simulations in an ats is also a simulation in the ats and we denote it by $\sqsubseteq$; that is, $p \sqsubseteq q$ (read "q simulates p") holds if and only if there exists a simulation $\mathcal{R}$ such that $p\mathcal{R}q$. If $p \sqsubseteq q$ and $q \sqsubseteq p$ both hold, then p and q are *similar*.

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 union of all bisimulations in an ats is also a bisimulation in the ats and we denote it by $\equiv$; that is, $p \equiv q$ (read "p is *bisimilar* to q") holds if and only if there exists a bisimulation $\mathcal{R}$ such that $p\mathcal{R}q$. It is well-known that bisimilarity implies similarity but not vice-versa: for a counterexample, see Example 9.

To illustrate our results, we will consider throughout the paper a more concrete example of an abstract transition system: the operational semantics of the concurrent language CCS [26]. For convenience, we ignore the renaming and hiding combinators, and concentrate on the sublanguage described by the grammar

$$p \ ::= \ 0 \ \mid \ x \ \mid \ a.p \ \mid \ p + p \ \mid \ p \mid p \ \mid \ \mu_x p,$$

where x ranges over process variables and a ranges over a non-empty set of actions $\mathcal{A}$, the set of the complementary actions $\bar{\mathcal{A}} = \{\bar{a} \mid a \in \mathcal{A}\}$, and $\{\tau\}$, where $\tau \notin \mathcal{A} \cup \bar{\mathcal{A}}$.

The intended meaning of these symbols is as follows: 0 represents the inactive process, $a.p$ represents a process prefixed by the action a, $+$ and $\mid$ are choice and parallel composition, respectively, and μ_x is the least fixed point operator, providing recursion. The operational semantics of CCS is specified by the transition rules in Figure 1.

CCS can be seen as an abstract transition system where $\Lambda = \mathcal{A} \cup \bar{\mathcal{A}} \cup \{\tau\}$, S is the set of all CCS expressions, and δ is the set of transitions which are derivable by the rules above. A *finite CCS process* is a CCS expression that does not contain μ. If S is restricted to the set of all finite CCS processes, then the resulting ats is noetherian.

$$\frac{}{B \longrightarrow B}\ \text{initial} \qquad \frac{\Delta \longrightarrow B}{\Delta, 1 \longrightarrow B}\ 1\mathcal{L} \qquad \frac{}{\longrightarrow 1}\ 1\mathcal{R}$$

$$\frac{\Delta, B, C \longrightarrow E}{\Delta, B \otimes C \longrightarrow E}\ \otimes\mathcal{L} \qquad \frac{\Delta_1 \longrightarrow B \qquad \Delta_2 \longrightarrow C}{\Delta_1, \Delta_2 \longrightarrow B \otimes C}\ \otimes\mathcal{R}$$

$$\frac{\Delta, B[y/x] \longrightarrow E}{\Delta, \exists x B \longrightarrow E}\ \exists\mathcal{L} \qquad \frac{\Delta \longrightarrow B[t/x]}{\Delta \longrightarrow \exists x B}\ \exists\mathcal{R}$$

$$\frac{\Delta_1, C \longrightarrow E \quad \Delta_2 \longrightarrow B}{\Delta_1, \Delta_2, B \multimap C \longrightarrow E}\ \multimap\mathcal{L} \qquad \frac{\Delta, B \longrightarrow C}{\Delta \longrightarrow B \multimap C}\ \multimap\mathcal{R}$$

Figure 2: A proof system for a fragment of linear logic. The rule $\exists$L has the proviso that y is not free in the lower sequent.

4 Linear logic specification of abstract transition systems

In this section, we will consider encoding abstract transition systems using linear logic and definitions. We will, in fact, use only the connectives $1, \otimes, \multimap$, and $\exists$, whose inference rules are given in Figure 2, along with $def\mathcal{L}$ and $def\mathcal{R}$ from Section 2. Before presenting our encoding, we state some properties of this combination of linear logic and definitions.

The cut rule
$$\frac{\Delta_1 \longrightarrow B \quad B, \Delta_2 \longrightarrow C}{\Delta_1, \Delta_2 \longrightarrow C}$$
is known to be admissible for this fragment of linear logic with definitions. This cut-elimination result is given by Girard in [12] or from the main result in [31]. In general, we shall only consider cut-free proofs when considering encodings of ats's.

Generally, there are many proofs for the same sequent, and many of these differ only in the order in which inference rules are applied. In particular, many inference rules permute over each other: if the order in which they are applied is switched, the proof would still be achieved in essentially the same way. Consider, for example, a proof that ends with the following two inference rules.

$$\frac{\dfrac{\Delta_1, C \longrightarrow D \quad \Delta_2 \longrightarrow B}{\Delta_1, \Delta_2, B \multimap C \longrightarrow D}\ \multimap\mathcal{L} \qquad \Delta_3 \longrightarrow E}{\Delta_1, \Delta_2, \Delta_3, B \multimap C \longrightarrow D \otimes E}\ \otimes\mathcal{R}.$$

These instances of the $\multimap\mathcal{L}$ and $\otimes\mathcal{R}$ can be permuted to yield

$$\frac{\dfrac{\Delta_1, C \longrightarrow D \quad \Delta_3 \longrightarrow E}{\Delta_1, \Delta_3, C \longrightarrow D \otimes E}\ \otimes\mathcal{R} \qquad \Delta_2 \longrightarrow B}{\Delta_1, \Delta_2, \Delta_3, B \multimap C \longrightarrow D \otimes E}\ \multimap\mathcal{L}.$$

In the case where the formula $B \multimap C$ appears in the right premise of the $\otimes\mathcal{R}$ inference rule, a corresponding permutation is also possible.

There are various pairs of inference rules that do not permute over each other. In the following Lemma, we write R_1/R_2 to mean an occurrence of the inference rule R_1 over the inference rule R_2. We do not need to consider permutations of two right-introduction rules since sequents have only one formula on their righthand-side.

Lemma 5 *The following pairs of inference rules do not permute over each other.*

$$1\mathcal{R}/1\mathcal{L} \quad \otimes\mathcal{R}/\otimes\mathcal{L} \quad \multimap\mathcal{L}/\multimap\mathcal{R} \quad \multimap\mathcal{L}/\otimes\mathcal{L} \quad \exists\mathcal{R}/\exists\mathcal{L}$$

$$\multimap\mathcal{L}/def\mathcal{L} \quad \otimes\mathcal{R}/def\mathcal{L} \quad \exists\mathcal{R}/def\mathcal{L} \quad def\mathcal{R}/def\mathcal{L}$$

All other pairs of inference rules permute.

Proof We do not examine explicitly here all pairings of inference rules. Instead we show only a few cases: most cases are similar and simple.

The case of nonpermutability of $\otimes\mathcal{R}/\otimes\mathcal{L}$ can be illustrated with the sequent $p \otimes q \longrightarrow q \otimes p$: this has a proof with an occurrence of $\otimes\mathcal{L}$ below $\otimes\mathcal{R}$ but there is no proof with $\otimes\mathcal{R}$ at the bottom. The other cases of nonpermutability can be shown by presenting similar counterexamples.

Above, it has already been shown that the pair $\multimap\mathcal{L}/\otimes\mathcal{R}$ permutes. Most of the other cases of permutability can be reasoned about as simply. The only difficult case involves showing that two atomic formulas on the left of a sequent can be introduced by the $def\mathcal{L}$ rule in either order: fortunately, this has already been proved in Section 2 (Proposition 4). The cases for $def\mathcal{L}/\otimes\mathcal{R}$ and $def\mathcal{L}/\otimes\mathcal{L}$ require applying substitutions to proofs, a notion formally defined in [20, 22]. ∎

To encode an ats in linear logic, we represent states as propositional constants. Actions will be represented by functional constants of type $\gamma \to \gamma$, where γ is a new type. The sequence of actions, or *trace*, $w = a_1 \cdots a_m$ $(m \geq 0)$ from Λ^* will be encoded as the term $\tilde{w} = \lambda w.a_1(\ldots(a_m\ w)\ldots)$. (Here, the symbol denoting an action is also used to denote the corresponding function symbol.) Function composition represents the concatenation of traces: $w_1 w_2$ is encoded as $\tilde{w_1} \circ \tilde{w_2} = \lambda w.\tilde{w_1}(\tilde{w_2}\ w)$. Besides the propositions that encode states, we need one other predicate tr of type $(\gamma \to \gamma) \to o$. Let $ats_1(\delta)$ be the definition given by the following set of clauses:

$$ats_1(\delta) = \{p \triangleq \exists W(tr(a \circ W) \otimes (tr(W) \multimap q)) \mid (p, a, q) \in \delta\}.$$

Notice that the symbol denoting a state is also used to denote the corresponding logical atom.

The following proposition shows how paths can be represented logically.

Proposition 6 *Let (Λ, S, δ) be an ats. Then $p \stackrel{w}{\Longrightarrow} q$ if and only if*

$$ats_1(\delta) \vdash \forall k((tr(k) \multimap q) \multimap (tr(\tilde{w} \circ k) \multimap p)).$$

Proof First notice that $ats_1(\delta) \vdash \forall k((tr(k) \multimap q) \multimap (tr(\tilde{w} \circ k) \multimap p))$ if and only if $ats_1(\delta) \vdash tr(\tilde{w} \circ k), tr(k) \multimap q \longrightarrow p$ for any constant k of type $\gamma \to \gamma$.

We prove the reverse direction by induction on the length of w. Assume w is ϵ and that $ats_1(\delta) \vdash tr(k), tr(k) \multimap q \longrightarrow p$. The proof of this sequent cannot end with $def\mathcal{L}$, since the predicate tr is not assumed to be defined. The proof cannot end with $def\mathcal{R}$, since that would require the proof of $tr(k) \longrightarrow tr(a \circ w')$ for some w', which is not provable. So the sequent can only be proved if it is the conclusion of an $\multimap\mathcal{L}$; thus, p and q are equal and $p \stackrel{\epsilon}{\Longrightarrow} q$ holds immediately.

If w is not empty then it is of the form au. The proof of the sequent $tr(a \circ \tilde{u} \circ k), tr(k) \multimap q \longrightarrow p$ cannot end with $def\mathcal{L}$, since $tr(a \circ \tilde{u} \circ k)$ is not a defined atom. It cannot end with $\multimap\mathcal{L}$, since that would require a proof of either $\longrightarrow tr(k)$ or $tr(a \circ \tilde{u} \circ k) \longrightarrow tr(k)$, neither of which are provable. So the sequent must be the consequence of $def\mathcal{R}$, there must be an r such that $p \stackrel{a}{\longrightarrow} r$, and the proof must be structured as follows:

$$\dfrac{\dfrac{\dfrac{\overline{tr(a \circ \tilde{u} \circ k) \longrightarrow tr(a \circ \tilde{u} \circ k)} \quad \dfrac{tr(\tilde{u} \circ k), tr(k) \multimap q \longrightarrow r}{tr(k) \multimap q \longrightarrow tr(\tilde{u} \circ k) \multimap r}\multimap\mathcal{R}}{tr(a \circ \tilde{u} \circ k), tr(k) \multimap q \longrightarrow tr(a \circ \tilde{u} \circ k) \otimes (tr(\tilde{u} \circ k) \multimap r)}\otimes\mathcal{R}}{tr(a \circ \tilde{u} \circ k), tr(k) \multimap q \longrightarrow \exists W(tr(a \circ W) \otimes (tr(W) \multimap r))}\exists\mathcal{R}}{tr(a \circ \tilde{u} \circ k), tr(k) \multimap q \longrightarrow p}def\mathcal{R}.$$

This proof contains a subproof of $tr(\tilde{u} \circ k), tr(k) \multimap q \longrightarrow r$. By the inductive hypothesis, we have $r \stackrel{u}{\Longrightarrow} q$ and thus $p \stackrel{au}{\Longrightarrow} q$.

The proof in the forward direction is also by induction of the length of w. If w is empty, then $p \stackrel{w}{\Longrightarrow} q$ implies that p and q are equal, and thus the sequent $tr(\tilde{w} \circ k), tr(k) \multimap q \longrightarrow p$ is provable (via $\multimap\mathcal{L}$ and two uses of the initial rule). If w is nonempty then it is of the form au and there is some r such that $p \stackrel{a}{\longrightarrow} r \stackrel{u}{\Longrightarrow} q$. By the inductive hypothesis, there is a proof of $tr(\tilde{u} \circ k), tr(k) \multimap q \longrightarrow r$. If we add to that proof the inference rules displayed above, we then get a proof of $tr(a \circ \tilde{u} \circ k), tr(k) \multimap q \longrightarrow p$. ∎

We prove now that with this encoding (along with $def\mathcal{L}$), simulation corresponds to reverse linear implication. To do so, we introduce two new items. First, given a fixed ats (Λ, S, δ) and $(p, q) \in S \times S$, a *premise set for* (p, q) is a set $P \subseteq S \times S$ such that for every $a \in \Lambda$ and $p' \in S$ such that $p \stackrel{a}{\longrightarrow} p'$ there exists a $q' \in S$ such that $q \stackrel{a}{\longrightarrow} q'$ and $(p', q') \in P$. Premise sets need not exist, but if there is a simulation $\mathcal{R}$ that contains (p, q) then there is a premise set P for that pair such that $P \subseteq \mathcal{R}$. We restrict premise sets to be minimal (assuming the axiom of choice). Notice also that premise sets can be empty. Second, we introduce the following class of inference rules:

$$\frac{\{p' \longrightarrow q' \mid (p', q') \in P\}}{p \longrightarrow q}\ SIM_1$$

where P is a premise set for (p, q). Notice that this rule is finitary if the ats is finitely branching. Let $\vdash_{SIM_1} \Delta \longrightarrow C$ denote the proposition that the sequent $\Delta \longrightarrow C$ can be proved using only the SIM_1 inference rule.

Lemma 7 *Let (Λ, S, δ) be a noetherian ats and let $p, q \in S$. Then $ats_1(\delta) \Vdash p \longrightarrow q$ if and only if $\vdash_{SIM_1} p \longrightarrow q$.*

Proof Assume $ats_1(\delta) \Vdash p \longrightarrow q$. If the proof of this sequent uses only the initial rule, then p and q are the same state. In a noetherian ats, a proof of $p \longrightarrow p$ using only the SIM_1 inference rule can always be constructed, a result that follows from a simple induction on $meas(p)$. (This is the only use of the noetherian assumption in this proof.) Otherwise, the only inference rules that can be applied to prove such a sequent are either $def\mathcal{R}$ or $def\mathcal{L}$ since p and q are atomic formulas. By Lemma 5, occurrences of $def\mathcal{L}$ can be permuted down over $def\mathcal{R}$ and the right introduction rules for $\exists$, $\otimes$, and $\multimap$. So we can assume that the last inference rule used to prove this sequent is $def\mathcal{L}$, and thus the proof has the form

$$\frac{\{\exists W (tr(a \circ W) \otimes (tr(W) \multimap p')) \longrightarrow q \ \mid \ p \stackrel{a}{\longrightarrow} p'\}}{p \longrightarrow q}\ def\mathcal{L}.$$

Consider one of these premises, say, $\exists W (tr(a \circ W) \otimes (tr(W) \multimap p')) \longrightarrow q$. Given the permutations of the inference rules listed in Lemma 5, we can assume that this premise is obtained by left introduction rules for $\exists$ and $\otimes$, yielding the sequent $tr(a \circ w), tr(w) \multimap p' \longrightarrow q$ where w is a variable. The proof of this sequent cannot end with $def\mathcal{L}$ since $tr(a \circ w)$ is not a defined atom. It also cannot end with $\multimap\mathcal{L}$ since this would require a proof of either $\longrightarrow tr(w)$ or $tr(a \circ w) \longrightarrow tr(w)$, neither of which are provable. Therefore, the proof must end with $def\mathcal{R}$, which yields the sequent

$$tr(a \circ w), tr(w) \multimap p' \longrightarrow \exists W (tr(b \circ W) \otimes (tr(W) \multimap q')),$$

where $q \stackrel{b}{\longrightarrow} q'$. There is a proof of this sequent, however, only if $a = b$ and $\exists W$ is instantiated with w (using the $\exists$ right introduction rule), bringing us to the sequent

$$tr(a \circ w), tr(w) \multimap p' \longrightarrow tr(a \circ w) \otimes (tr(w) \multimap q').$$

A proof of this sequent must end with the right introduction rules for $\otimes$ and $\multimap$ and the left introduction rule for $\multimap$, and contain a subproof of the sequent $p' \longrightarrow q'$. Thus, if we collect all such pairs (p', q') into the set P, we have a premise set for (p, q) and we have established an instance of the SIM_1 rule. Since all these proof steps are forced, this proves the completeness of SIM_1.

The converse follows simply by noting that each instance of the SIM_1 inference rule can be built using several instances of inference rules from linear logic, one instance of $def\mathcal{L}$, and possibly several instances of $def\mathcal{R}$. ∎

Notice that for a noetherian ats we do not need instances of the initial sequent rule to be used with defined atoms: a proof of $p \longrightarrow p$ can be obtained using only the SIM_1 rule or the corresponding combination of $def\mathcal{L}$, $def\mathcal{R}$, $\exists\mathcal{L}$, $\exists\mathcal{R}$, $\otimes\mathcal{L}$, $\otimes\mathcal{R}$, $\multimap\mathcal{L}$, $\multimap\mathcal{R}$, and the initial rule for undefined atoms. 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 $ats_1(\delta)$ as a definition, states become logical constants (in the sense that they have left and right introduction rules), and hence we do not need any instance of the initial rule. Notice, however, if we did admit the initial inference rule along with the SIM_1 inference rule, the previous lemma could be extended to the non-noetherian case.

We can now establish our first proof theoretic connection to simulation.

Theorem 8 *Let (Λ, S, δ) be a 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 Lemma 7, we need only show that $\vdash_{SIM_1} p \longrightarrow q$ if and only if q simulates p. First, assume that the sequent $p \longrightarrow q$ has a proof that contains only instances of the SIM_1 inference rule. Let $\mathcal{R}$ be the set of all pairs (r, s) such that the sequent $r \longrightarrow s$ has an occurrence in that proof. It is easy to verify that $\mathcal{R}$ is a simulation.

Conversely, assume that q simulates p. Thus there is a simulation $\mathcal{R}$ such that $p\mathcal{R}q$. The proof is by complete induction on the measure of p, $meas(p)$. Since $p\mathcal{R}q$, there is a premise set $P \subseteq \mathcal{R}$ for (p, q). If $(p', q') \in P$, then $p'\mathcal{R}q'$ and $meas(p') < meas(p)$, so we have by the induction hypothesis $\vdash_{SIM_1} p' \longrightarrow q'$. Thus, we have proved $\vdash_{SIM_1} p \longrightarrow q$. ∎

This theorem states that in a noetherian ats, simulation can be identified with the logical connective $\multimap$ via the encoding $ats_1(\delta)$. As a consequence, logical equivalence of p and q, namely the provability of both $p \multimap q$ and $q \multimap p$, corresponds to *similarity* of p and q, i.e. simulation in both directions. Notice that bisimilarity implies similarity but does not coincide with it, as the following example shows. Hence logical equivalence is coarser than bisimulation equivalence.

Example 9 *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*

$$\{(p_1, q_1), (p_2, q_2), (p_3, q_3), (p_4, q_2)\} \text{ and } \{(q_1, p_1), (q_2, p_2), (q_3, p_3)\}$$

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 (p_1, q_1).

As a consequence of this example, we see no way to make bisimilarity into a logical connective; that is, into a constant of type $o \to o \to o$ with left and right introduction rules that enjoys cut-elimination. In the special case of noetherian and determinate ats, however, similarity implies bisimilarity, and hence logical equivalence and bisimulation equivalence coincide.

$$\frac{}{\Delta, B \longrightarrow B}\ \text{initial} \qquad \frac{}{\Delta \longrightarrow \top}\ \top\mathcal{R}$$

$$\frac{\Delta, B, C \longrightarrow E}{\Delta, B \wedge C \longrightarrow E}\ \wedge\mathcal{L} \qquad \frac{\Delta \longrightarrow B \qquad \Delta \longrightarrow C}{\Delta \longrightarrow B \wedge C}\ \wedge\mathcal{R}$$

$$\frac{\Delta, B[t/x] \longrightarrow C}{\Delta, \forall x B \longrightarrow C}\ \forall\mathcal{L} \qquad \frac{\Delta \longrightarrow B[y/x]}{\Delta \longrightarrow \forall x B}\ \forall\mathcal{R}$$

$$\frac{\Delta, B[y/x] \longrightarrow E}{\Delta, \exists x B \longrightarrow E}\ \exists\mathcal{L} \qquad \frac{\Delta \longrightarrow B[t/x]}{\Delta \longrightarrow \exists x B}\ \exists\mathcal{R}$$

$$\frac{\Delta, C \longrightarrow E \quad \Delta \longrightarrow B}{\Delta, B \supset C \longrightarrow E}\ \supset\mathcal{L} \qquad \frac{\Delta, B \longrightarrow C}{\Delta \longrightarrow B \supset C}\ \supset\mathcal{R}$$

Figure 3: A proof system for a fragment of intuitionistic logic. The rules ∀R and ∃L have the proviso that y is not free in the lower sequent of that inference rule. The structural rules of exchange and contraction are implicit in the use of sets on the left-hand side of the sequent arrow. The weakening rule is implicit in the presence of Δ in the axioms.

Proposition 10 *Let (Λ, S, δ) be a noetherian and determinate ats and let $p, q \in S$. Then $ats_1(\delta) \Vdash p \multimap q$ and $ats_1(\delta) \Vdash q \multimap p$ if and only if p is bisimilar to q.*

Proof We prove that if the ats is noetherian and determinate then similarity and bisimilarity coincide. The proposition then follows from Theorem 8. We show that similarity implies bisimularity by well-founded induction on $meas(p)$. Consider a transition $p \xrightarrow{a} p'$. Since $p \sqsubseteq q$, there must exist a q' such that $q \xrightarrow{a} q'$ and $p' \sqsubseteq q'$. Since the ats is determinate, $q \sqsubseteq p$ implies $q' \sqsubseteq p'$. Since $meas(p') < meas(p)$, the inductive hypothesis yields $p' \equiv q'$. Symmetrically, for any $q \xrightarrow{a} q'$, there exist $p \xrightarrow{a} p'$ such that $q' \equiv p'$. Therefore $p \equiv q$. ▮

5 Encoding one-step transitions as atomic judgments

While using linear logic directly to encode transitions was rather natural and immediate, the resulting encoding will not be able to provide us information about bisimilarity in many situations, as is illustrated in Example 9. To overcome this problem we give a second encoding of abstract transition systems, this time encoding the relations between states as predicates rather than logical connectives. Since this encoding will not need linearity, it uses intuitionistic logic instead. Definitions will play a role rather similar to the one they played in the previous encoding.

Consider intuitionistic logic using the connectives $\top$ for true, $\wedge$ for conjunction, $\supset$ for implication, and $\forall$ and $\exists$ for universal and existential quantification. Sequents for this intuitionistic logic are of the form $\Delta \longrightarrow B$ where Δ is a set of formulas and B is a formula. The sequent calculus for this fragment of intuitionistic logic is given in Figure 3. Schroeder-Heister showed in [31] that if definitions do not contain implications in clause bodies, then cut-elimination can be proved for intuitionistic logic extended with $def\mathcal{L}$ and $def\mathcal{R}$ rules. However, we shall need a strong form of definition that allows certain stratified occurrences of implications. In particular, we assume that each predicate symbol p in the language has associated with it a natural number $lvl(p)$, the *level* of the predicate. The notion of level is then extended to formulas as follows. Given a formula B, its *level* $lvl(B)$ is defined as follows: $lvl(p(\bar{t})) = lvl(p)$, $lvl(\top) = 0$, $lvl(B \wedge C) = \max(lvl(B), lvl(C))$, $lvl(B \supset C) = \max(lvl(B) + 1, lvl(C))$, and $lvl(\forall x.B) = lvl(\exists x.B) = lvl(B)$. We say that a defini-

$$\begin{array}{rrcl} \forall P[& multi(P, nil, P) & \triangleq & \top] \\ \forall A, P, Q, W[& multi(P, A :: W, Q) & \triangleq & \exists R(one(P, A, R) \wedge multi(R, W, Q))] \end{array}$$

Figure 4: The definition *path*.

tion is *stratified* if for every clause $\forall\bar{x}[H \triangleq B]$ of that definition, $lvl(B) \leq lvl(H)$. It is proved in [22] that cut can be eliminated from intuitionistic logic extended with stratified definitions. In the rest of this paper, when we use the term "definition" we shall mean "stratified definition".

The following Lemma is similar to Lemma 5 and is proved similarly.

Lemma 11 *The following pairs of inference rules do not permute over each other. We write R_1/R_2 to mean an occurrence of inference rule R_1 over inference rule R_2.*

$$\forall\mathcal{L}/\forall\mathcal{R} \quad \supset\mathcal{L}/\supset\mathcal{R} \quad \exists\mathcal{R}/\exists\mathcal{L} \quad \forall\mathcal{L}/\exists\mathcal{L} \quad \exists\mathcal{R}/def\mathcal{L} \quad \forall\mathcal{L}/def\mathcal{L} \quad def\mathcal{R}/def\mathcal{L}$$

All other pairs of inference rules permute.

Let (Λ, S, δ) be an ats and let the primitive types σ and α denote the type of the elements in S and Λ, respectively. Let $one\colon \sigma \to \alpha \to \sigma \to o$ be a predicate denoting the one-step transition relation δ. Let $ats_2(\delta)$ be the definition given by the following set of clauses:

$$ats_2(\delta) = \{one(p, a, q) \triangleq \top \mid (p, a, q) \in \delta\}.$$

To encode paths within a transition system we use the predicate $multi : \sigma \to list(\alpha) \to \sigma \to o$, which is defined by the definition *path* given in Figure 4. The predicates *one* and *multi* are assumed to have level 0. 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 Proposition 6; its proof is simpler.

Proposition 12 *Let (Λ, S, δ) be an ats. Then $ats_2(\delta), path \vdash multi(p, w, q)$ if and only if $p \stackrel{w}{\Longrightarrow} q$.*

The encoding $ats_2(\delta)$ is based on an extensional description of δ, hence the definition will be infinite if δ is infinite. In specific transition systems the transition relation might be described structurally. This is the case for CCS, whose transitions can be encoded as the definition $ccs(\mathcal{A})$, given in Figure 5. Here the combinators of CCS are typed as follow: the action prefix has type $\alpha \to \sigma \to \sigma$, the $+$ and $|$ both have type $\sigma \to \sigma \to \sigma$, and μ has type $(\sigma \to \sigma) \to \sigma$. We assume the silent action $\tau : \alpha$ is not a member of the set $\mathcal{A}$.

Observe that we are using meta-level λ-abstraction to encode $\mu_x P$: such a term is represented as μM, where M is meant to be the abstraction $\lambda x.P$. Thus the term $P[\mu_x P/x]$ can be represented simply by $M(\mu M)$ without introducing an explicit notion of substitution (β-conversion in the meta-logic performs the necessary substitution).

The following result shows that CCS transitions are completely described by logical derivability in $ccs(\mathcal{A})$.

Proposition 13 *The transition $p \stackrel{a}{\longrightarrow} q$ holds if and only if $ccs(\mathcal{A}) \vdash one(p, a, q)$, and $p \stackrel{w}{\Longrightarrow} q$ holds if and only if $ccs(\mathcal{A}), path \vdash multi(p, w, q)$.*

$$
\begin{array}{rcll}
\forall A, P\, [& one(A.P, A, P) & \triangleq & \top] \\
\forall A, P, Q, R\, [& one(P \mid R, A, Q \mid R) & \triangleq & one(P, A, Q)] \\
\forall A, P, Q, R\, [& one(R \mid P, A, R \mid Q) & \triangleq & one(P, A, Q)] \\
\forall A, P, Q, R\, [& one(P + R, A, Q) & \triangleq & one(P, A, Q)] \\
\forall A, P, Q, R\, [& one(P + R, A, Q) & \triangleq & one(R, A, Q)] \\
\forall A, P, Q\, [& one(\mu M, A, Q) & \triangleq & one(M(\mu M), A, Q)] \\
\forall P, Q, R, S\, [& one(P \mid Q, \tau, R \mid S) & \triangleq & \exists A, B(comp(A, B) \wedge one(P, A, R) \wedge one(Q, B, S))]
\end{array}
$$

For each $a \in \mathcal{A}$, the two clauses $comp(a, \bar{a}) \triangleq \top$ and $comp(\bar{a}, a) \triangleq \top$.

Figure 5: The definition $ccs(\mathcal{A})$. Here, $\mathcal{A}$ is a non-empty set of actions.

$$
\begin{array}{rcll}
\forall P, Q\, [& sim(P, Q) & \triangleq & \forall A \forall P'.\, one(P, A, P') \supset \\
 & & & \qquad \exists Q'.\, one(Q, A, Q') \wedge sim(P', Q')] \\
\forall P, Q\, [& bisim(P, Q) & \triangleq & [\forall A \forall P'.\, one(P, A, P') \supset \\
 & & & \qquad \exists Q'.\, one(Q, A, Q') \wedge bisim(P', Q')] \wedge \\
 & & & [\forall A \forall Q'.\, one(Q, A, Q') \supset \\
 & & & \qquad \exists P'.\, one(P, A, P') \wedge bisim(Q', P')]]
\end{array}
$$

Figure 6: The definition *sims*.

This proposition can be proved by simple structural induction by showing that proofs using the inference rules in Figure 1 for CCS are essentially identical to sequent calculus proofs over the corresponding clauses in the definition $ccs(\mathcal{A})$. Notice that the sequent calculus proofs involve only right introduction rules.

If we explicitly represent the transition step by a predicate *one* (defined by $ats_2(\delta)$ or by a system like $ccs(\mathcal{A})$), then it is possible to characterize simulation and bisimulation as predicates *sim* and *bisim* given by the definition *sims*, presented in Figure 6. Since the level of *one* is 0, we need to assign to both *sim* and *bisim* the level 1 (or higher) in order to make this definition stratified.

In order to prove that these encodings correctly capture simulation and bisimulation, we need to show that we can restrict the use of the $def\mathcal{L}$ rule without affecting provability. Toward that end we introduce another version of the left introduction rule for definitions, $def\mathcal{L}_{nc}$. This version is the same as $def\mathcal{L}$ but with the additional restriction that the introduced atom is not a member of the side formulas on the left:

$$
\frac{\{B\theta, \Delta\theta \longrightarrow C\theta \mid \theta \in csu(A, H) \text{ for some clause } \forall \bar{x}.[H \triangleq B]\}}{A, \Delta \longrightarrow C}\ def\mathcal{L}_{nc}, \text{ provided } A \notin \Delta.
$$

As before, the variables $\bar{x}$ in this rule need to be chosen so that they are not free in any formula of the lower sequent. We shall show that in a proof of the sequents $\longrightarrow sim(p, q)$ or $\longrightarrow bisim(p, q)$, this additional restriction does not affect provability. We shall call an instance of $def\mathcal{L}$ *redundant* if it is not an instance of $def\mathcal{L}_{nc}$. A proof in which all instances of $def\mathcal{L}$ are instances of $def\mathcal{L}_{nc}$ is a proof *without redundancies*.

Proposition 14 *Let (Λ, S, δ) be an ats and let $p, q \in S$.*

- *If* $ats_2(\delta), sims \Vdash sim(p,q)$*, then there is also a proof of* $\longrightarrow sim(p,q)$ *without redundancies, and*

- *If* $ats_2(\delta), sims \Vdash bisim(p,q)$*, then there is also a proof of* $\longrightarrow bisim(p,q)$ *without redundancies.*

The same holds if we replace $ats_2(\delta)$ *by* $ccs(\mathcal{A})$ *and restrict* p *and* q *to be finite processes.*

Proof We shall show how to replace a redundant instance of $def\mathcal{L}$ at the root of a proof by an instance of $def\mathcal{L}_{nc}$. By repeatedly removing such redundant instances, we can convert a proof using $def\mathcal{L}$ into a proof using $def\mathcal{L}_{nc}$.

Consider a proof of $A, \Delta \longrightarrow C$ in which that sequent is the consequence of a redundant application of $def\mathcal{L}$ that introduces A. Thus, $\Delta = \{A\} \cup \Delta'$ for some Δ' that does not contain A. Also assume that this proof contains no other redundant occurrences of $def\mathcal{L}$. We shall transform this proof into one of $A, \Delta' \longrightarrow C$ that uses only $def\mathcal{L}_{nc}$. The premise of this occurrence of $def\mathcal{L}$ is the set of sequents

$$\{B\theta, \Delta\theta \longrightarrow C\theta \mid \theta \in csu(A, H) \text{ for some clause } \forall\bar{x}[H \stackrel{\triangle}{=} B]\}.$$

For each of the sequents $B\theta, A\theta, \Delta'\theta \longrightarrow C\theta$ we construct a proof of $B\theta, \Delta'\theta \longrightarrow C\theta$ as follows. Since there is a definitional clause $\forall\bar{x}[H \stackrel{\triangle}{=} B]$ such that $A\theta = H\theta$, there is a proof of the sequent $B\theta \longrightarrow A\theta$ using $def\mathcal{R}$ and initial. Using the cut-elimination theorem of [22], we know there is a (cut-free) proof of $B\theta, \Delta'\theta \longrightarrow C\theta$. (Since the left-hand side of the sequent is a set, we can combine the two occurrences of $B\theta$.) We can use all of these proofs of the sequents $B\theta, \Delta'\theta \longrightarrow C\theta$ and an instance of the $def\mathcal{L}$ rule to prove the sequent $A, \Delta' \longrightarrow C$.

To guarantee that this new proof does not contain redundant occurrences of the $def\mathcal{L}$ rule, we need to know that the cut-elimination process does not introduce new redundant occurrences, and also that the implicit contraction on $B\theta$ in the premises does not result in a new redundant occurrence.

The cut-elimination procedure in [22] can only introduce redundant occurrences of the $def\mathcal{L}$ rule if there are occurrences of $\supset\mathcal{L}$ or $nat\mathcal{L}$ in the proof. An examination of the relevant formulas and definitions reveals that the only formulas that will be on the left-hand side of a sequent in a cut-free proof of $\longrightarrow sim(p,q)$ or $\longrightarrow bisim(p,q)$ will be atomic formulas constructed using the predicate *one* (and the predicate *comp* in the CCS case). Thus the proof will not contain occurrences of the $\supset\mathcal{L}$ or $nat\mathcal{L}$ rules, so cut-elimination will not introduce redundant occurrences of $def\mathcal{L}$.

The implicit contraction of $B\theta$ will introduce a redundant occurrence of $def\mathcal{L}$ only if $B\theta$ is the main formula of an occurrence of $def\mathcal{L}$. Remember that B is the body of a definitional clause $\forall\bar{x}[H \stackrel{\triangle}{=} B]$ such that $\theta \in csu(A, H)$, and A is an atomic formula constructed using either *one* or *comp*. With the definition $ats_2(\delta)$, B will always be the formula $\top$, which cannot be the main formula of $def\mathcal{L}$. With the definition $ccs(\mathcal{A})$, however, B may be another atomic formula constructed using *one*, and so may be the main formula of $def\mathcal{L}$. Since we have excluded the μ operator, the first argument to *one* in B is necessarily a subexpression of the first argument to *one* in A. This provides us with a measure that guarantees that any new redundant occurrence of $def\mathcal{L}$ that is introduced is strictly smaller than the one eliminated. Since our definition is finite, the original occurrence of $def\mathcal{L}$ will have a finite number of premises, so at most a finite number of new redundant occurrences are introduced. As a result, we can apply the elimination process to the new occurrences, and eventually the process will terminate. ∎

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

$$\frac{\{\longrightarrow sim(p',q') \mid (p',q') \in P\}}{\longrightarrow sim(p,q)}\ SIM_2$$

where P is a premise set for (p,q). We also present an inference rule for bisimulation:

$$\frac{\{\longrightarrow bisim(p',q') \mid (p',q') \in P\} \cup \{\longrightarrow bisim(q',p') \mid (q',p') \in Q\}}{\longrightarrow bisim(p,q)}\ BISIM_2$$

where P is a premise set for (p,q) and Q is a premise set for (q,p). Again, these rules are finitary if the ats is finitely branching. In the case of CCS, one condition which guarantees this property is that recursion variables in bodies of μ-terms only occur prefixed. Let $\vdash_{SIM_2} \Delta \longrightarrow C$ (respectively, $\vdash_{BISIM_2} \Delta \longrightarrow C$) denote the proposition that the sequent $\Delta \longrightarrow C$ can be proved using only occurrences of the SIM_2 (respectively, $BISIM_2$) inference rule.

The following lemma is the analogue of Lemma 7 for this second encoding.

Lemma 15 *Let (Λ, S, δ) be an ats and let $p, q \in S$. Then*

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

The same holds if we replace $ats_2(\delta)$ by $ccs(\mathcal{A})$ and restrict p and q to be finite processes.

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 $def\mathcal{R}$ rule using $sims$, and thus the sequent

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

must be provable. The only (cut-free) proof of this sequent must end in three occurrences of right-introduction rules and a proof of the sequent

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

(here, A and P' are variables introduced by the $\forall\mathcal{R}$ rule). By permuting inference rules, we can assume that this latter sequent is proved with $def\mathcal{L}$ (or more precisely $def\mathcal{L}_{nc}$) and that its premises, namely the sequents

$$\longrightarrow \exists Q'.\ one(q,a,Q') \wedge sim(p',Q'),$$

where the pair (a,p') ranges over $\langle\langle p \rangle\rangle$, are each provable. Such a sequent is provable only if the quantifier $\exists Q'$ is instantiated with q' where $q \xrightarrow{a} q'$. Let P be the premise set arising from collecting together all pairs (p',q') for such values p' and q'. Thus, our original sequent is provable if and only if for every $(p',q') \in P$ the sequent $\longrightarrow sim(p',q')$ is provable.

The other direction follows by reversing these reasoning steps. ∎

Now the following can be proved using Lemma 15 in a manner analogous to the proof of Theorem 8 using Lemma 7.

Theorem 16 *Let (Λ, S, δ) be a 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* $p \equiv q$.

Concerning CCS, the full language is not noetherian because of the presence of the recursion operator. If we consider only expressions without μ, i.e. finite processes, then the same property holds, as is witnessed by the following theorem. We omit the proof of this theorem since it is essentially the same as the preceding proof: the main difference is that the definition of the one-step transitions for CCS is a recursive definition.

Theorem 17 *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* $p \equiv q$.

6 Infinite behavior

As we mentioned in Section 2, a definition can be seen as defining the predicates in the heads of the clauses by mutual recursion. Notice that we use the technique of stratification to give meaning to definitions containing implications. For a discussion on stratified specifications in the presence of negation (which can be reduced to a use of implication), see, for example, [1]. We now examine how such recursive definitions can be viewed as fixed points of suitable operators.

The clauses for defining the level 0 predicate *one* (both for $ats_2(\delta)$ and for $ccs(\mathcal{A})$) are Horn clauses when $\stackrel{\triangle}{=}$ is replaced with reverse implication. The usual meaning of predicates defined by Horn clauses is given by the least fixed point of the monotone, one-step inference operator associated with the clauses [2]. In our case, this interpretation for the predicate *one* coincides with the transition relation. That is, $one(p, a, p')$ is a member of the least fixed point if and only if $p \xrightarrow{a} p'$. For the definition $ats_2(\delta)$, this is trivial, and for the definition $ccs(\mathcal{A})$, this follows from Proposition 13 and the fact that the least fixed point characterizes the set of atomic consequences [2].

We now define an analogous one-step inference operator associated with the definition *sims*. In that definition, simulation and bisimulation are predicates of level 1 and the clauses are of the form $\forall P \forall Q[r(P, Q) \stackrel{\triangle}{=} \Phi]$, where the formula Φ contains free occurrences of the variables P and Q, strictly positive occurrences of the predicate r (namely, *sim* or *bisim*), and both positive and negative occurrences of the predicate *one*. With such a clause we associate a function ϕ from binary relations to binary relations whose definition corresponds to the formula Φ, except that atomic formulas of the form $one(p, a, p')$ are replaced with their denotation, namely, $p \xrightarrow{a} p'$. We are following here the classical approach to the fixed point semantics of stratified definitions: first, we determine the meaning of predicates of level lower than n; then their meaning is used in the definition of the one-step inference operator for predicates of level n. Since in the definition of the latter the predicate occurring negatively must have level lower than n, this techniques ensures that the one-step inference operator is monotone and thus has fixed points. In general, there will be more than one such fixed point.

Notice that both $def\mathcal{L}$ and $def\mathcal{R}$ are sound for all the relations which are fixed points of ϕ. To see this, assume that for any relation r there is only one such definitional clause (in case there are more, we group them in one clause which has as body the disjunction of the bodies). Then observe that the use of $def\mathcal{L}$ corresponds to replacing $\stackrel{\triangle}{=}$ by $\supset$ in the clause, that is, to assuming the formula $\forall P \forall Q[r(P, Q) \supset \Phi]$. The case for $def\mathcal{R}$ corresponds to the converse: that is, to replacing $\stackrel{\triangle}{=}$ with $\subset$.

Let Φ_s and Φ_b be the bodies of the clauses given in *sims* for *sim* and *bisim*, respectively, and consider the following corresponding functions ϕ_s and ϕ_b, on binary relations, associated with these formulas.

$$\begin{aligned}
\phi_s(\mathcal{R}) := \{(P,Q) \mid\ & \forall A \in \Lambda \forall P' \in S \text{ if } P \xrightarrow{A} P' \text{ then } \exists Q' \text{ such that } Q \xrightarrow{A} Q' \text{ and} \\
& (P',Q') \in \mathcal{R}\} \\
\phi_b(\mathcal{R}) := \{(P,Q) \mid\ & [\forall A \in \Lambda \forall P' \in S \text{ if } P \xrightarrow{A} P' \text{ then } \exists Q' \text{ such that } Q \xrightarrow{A} Q' \text{ and} \\
& (P',Q') \in \mathcal{R}] \text{ and } [\forall A \in \Lambda \forall Q' \in S \text{ if } Q \xrightarrow{A} Q' \text{ then } \exists P' \text{ such} \\
& \text{that } P \xrightarrow{A} P' \text{ and } (Q',P') \in \mathcal{R}]\}
\end{aligned}$$

We can see from their definitions that $\sqsubseteq$ and $\equiv$ are the greatest fixed points of ϕ_s and ϕ_b, respectively. Notice that in proofs of $\longrightarrow sim(p,q)$ and $\longrightarrow bisim(p,q)$ using definitions $ats_2(\delta)$ and *sims*, $def\mathcal{L}$ is used with $ats_2(\delta)$ but not with *sims*.

As the following example shows, when the transition system is not noetherian, the "if" parts of Theorem 16 may not hold.

Example 18 *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. Notice that an attempt to prove it would end up in a circularity.*

Notice that $ats_2(\delta), sims \Vdash sim(p,q)$ holds if and only if (p,q) is contained in every fixed point of ϕ_s and that $ats_2(\delta), sims \Vdash bisim(p,q)$ holds if and only if (p,q) is contained in every fixed point of ϕ_b. In a noetherian ats, ϕ_s and ϕ_b have unique fixed points, and it is for this reason that $\sqsubseteq$ and $\equiv$ can be completely characterized in a noetherian ats by provability (Theorem 16).

One attempt to characterize the greatest fixed point of the relation transformer ϕ proof-theoretically is to introduce a notion of "proof with finite or infinite height". 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 $\vdash^\omega_L \Delta \longrightarrow C$ the existence of an ω-proof in L for $\Delta \longrightarrow C$. For example, $\vdash^\omega_{SIM_2} \Delta \longrightarrow C$ is true if $\Delta \longrightarrow C$ has an ω-proof using only SIM_2. If the set L of inference rules is determined by those in intuitionistic logic and instances of $def\mathcal{L}_{nc}$ and $def\mathcal{R}$ for some definition D, then we write $D \Vdash^\omega \Delta \longrightarrow C$. Notice that an ω-proof can have finite or infinite height, and that this is orthogonal to the proof being finitely or infinitely branching, which is related to the possibility of having infinitary rules.

We prove now that Lemma 15 still holds for ω-proofs.

Lemma 19 *Let (Λ, S, δ) be an ats and let $p, q \in S$. Then*

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

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 definition $ats_2(\delta)$. Since this sequent must be proved by one use of $def\mathcal{R}$, two uses of $\forall$R, and one use of $\supset$R, we have

$$ats_2(\delta) \Vdash^\omega one(p,A,P') \longrightarrow \exists q'[one(q,A,q') \wedge sim(P',q')],$$

where A and P' are variables. At this point, the proof can proceed by either $def\mathcal{L}$ or $\exists\mathcal{R}$. If the choice is $def\mathcal{L}$, 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 $\exists\mathcal{R}$ would give rise to a conjunction, 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 $def\mathcal{R}$ 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. ∎

It is important for this Lemma that we use $def\mathcal{L}_{nc}$ (see Section 5) and not $def\mathcal{L}$ since the latter can lead to infinite sequences of redundant occurrences of $def\mathcal{L}$.

We can now extend Theorem 16 to ω-proofs and drop the noetherian condition.

Theorem 20 *Let (Λ, S, δ) be an ats, and let p and q be members of S. Then*

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

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 Lemma 19, $\vdash^\omega_{SIM_2} sim(p,q)$. 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 (p,q).

Assume next that $p \sqsubseteq q$ holds. We construct a monotonic 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 $\longrightarrow sim(p',q')$ where $p' \sqsubseteq q'$ holds, and T_{k+1} is obtained from T_k by attaching to each leaf of T_k an appropriate instance of SIM_2. T_0 is the tree consisting solely of the node labeled $\longrightarrow sim(p,q)$. Given T_k, for every leaf labeled $\longrightarrow sim(p',q')$, we know that $p' \sqsubseteq q'$, so $\sqsubseteq$ contains pairs $(p'_1,q'_1),\ldots,(p'_m,q'_m),\ldots$, where $\{(a_1,p'_1),\ldots,(a_m,p'_m),\ldots\} = \langle\langle p'\rangle\rangle$ and $\{(a_1,q'_1),\ldots,(a_m,q'_m),\ldots\} \subseteq \langle\langle q\rangle\rangle$ for some actions $a_1,\ldots,a_m,\ldots$. Hence in T_{k+1} the sequent $p' \longrightarrow q'$ can be placed at the conclusion of a SIM_2 rule, whose premises are $p'_1 \longrightarrow q'_1,\ldots,p'_m \longrightarrow q'_m,\ldots$. The limit of this sequence $\{T_k\}_{k\in\omega}$ is an ω-proof for $sim(p,q)$ using occurrences of the SIM_2 rule. The application of Lemma 19 yields the conclusion $ats_2(\delta), sims \Vdash^\omega sim(p,q)$. ∎

For Lemma 19 and Theorem 20 to hold, it is important that the relation *one* be defined in a "noetherian" way itself. If we consider a recursive definition for *one*, like in $ccs(\mathcal{A})$ (with the clause for μ), then the reverse direction of these equivalences does not necessarily hold, as the following example illustrates.

Example 21 *The CCS terms $\mu_x x$ and $\mu_x a.x$ are such that neither $\vdash^\omega_{SIM_2} sim(\mu_x a.x, \mu_x x)$ nor $\mu_x a.x \sqsubseteq \mu_x x$ hold. However, the judgment $ccs(\mathcal{A}), sims \Vdash^\omega sim(\mu_x a.x, \mu_x x)$ holds.*

In fact, notice that $\mu_x x \xrightarrow{a} \mu_x a.x$ does not hold, while $ccs(\mathcal{A}) \vdash^\omega one(\mu_x x, a, \mu_x a.x)$. The infinite proof of $one(\mu_x x, a, \mu_x a.x)$ is in a sense 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 variation of Theorem 20 where $ats_2(\delta)$ is replaced with $ccs(\mathcal{A})$ holds.

7 An inductive encoding of simulation and bisimulation

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. We will see that several properties cannot be proved within the logic because their proof requires inductive reasoning. We will then discuss one possible approach to enhance sequent calculus with induction.

Example 22 *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*

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

which is easy to verify.

Example 23 *The properties "bisimulation is symmetric" and "bisimulation is transitive" hold in any transition system. The corresponding encodings of these two properties are also provable in the sequent calculus; that is, we have*

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

and

$$ats_2(\delta), sims \Vdash \forall P \forall Q \forall R[bisim(P,Q) \wedge bisim(Q,R) \supset bisim(P,R)],$$

both of which are easy to verify.

As the following examples illustrate, there are plenty of properties of $\equiv$ and $\sqsubseteq$ that cannot be proved within the logic. In fact, as we already observed, 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 24 *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 $\forall P \forall Q[bisim(P,Q) \supset sim(P,Q)]$ but, in general, if δ is a non-noetherian transition relation, this formula cannot be proved using the definitions $ats_2(\delta)$ and sims. For example, if we take the transition system $(\{a\}, \{p\}, \{(p,a,p)\})$ it is immediate to see that $\{(p,p)\}$ 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 25 *The property "bisimulation equivalence is reflexive" holds in any transition system. The formula $\forall P[bisim(P,P)]$ cannot be proved using the definitions $ats_2(\delta)$ and sims. Consider for instance the same transition system as in Example 24: the empty set $\emptyset$ is a bisimulation (the least fixed point of ϕ_b), and it is, of course, not reflexive.*

Example 26 *The property "bisimulation equivalence is preserved by the + operator" is true in CCS. This property can be expressed as the formula $\forall P \forall Q \forall R[bisim(P,Q) \supset bisim(P+R, Q+R))]$. This sequent cannot be proved using $ccs(\mathcal{A})$ 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 $(a.0, a.0 + a.0)$ but not the pair $(a.0 + \mu_x a.x, a.0 + a.0 + \mu_x a.x)$.*

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 in Examples 24 and 25 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 $ccs(\mathcal{A}), sims \vdash^{\omega} bisim(P,P)$. Concerning the implication $bisim(P,Q) \supset sim(P,Q)$, observe that any infinite proof for $\longrightarrow bisim(P,Q)$ contains an infinite proof of $\longrightarrow sim(P,Q)$.

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 induction to capture the greatest fixed point.

In particular, define the binary relations $\sqsubseteq_i$ and $\equiv_i$ for each natural number i as follows. Both $\sqsubseteq_0$ and $\equiv_0$ are defined to be $S \times S$, and $\sqsubseteq_{i+1} := \phi_s(\sqsubseteq_i)$ and $\equiv_{i+1} := \phi_b(\equiv_i)$. Now set $\sqsubseteq_\omega := \bigcap_i \sqsubseteq_i$ and $\equiv_\omega := \bigcap_i \equiv_i$.

It is easy to show that for finitely-branching transition systems, ϕ_s and ϕ_b are downward-continuous and, hence, $\sqsubseteq$ equals $\sqsubseteq_\omega$ and $\equiv$ equals $\equiv_\omega$. A simple induction shows that for $i \geq 0$, $\sqsubseteq \subseteq \sqsubseteq_i$ and $\equiv \subseteq \equiv_i$ and thus $\sqsubseteq \subseteq \sqsubseteq_\omega$ and $\equiv \subseteq \equiv_\omega$. The converse needs the finitely branching assumption and follows from Theorem 5.6 of [25]. In CCS, finite branching is guaranteed whenever all the recursion variables in μ-expressions are prefixed.

Thus, one approach to showing that two states are bisimilar is to show that for all natural numbers i, those two states are related by $\equiv_i$. Such statements can often be proved by induction on natural numbers. We can incorporate induction into our proof systems by introducing natural numbers using z for zero and s for successor and using the predicate nat and the following "introduction" rules for this new predicate.

$$\frac{}{\Delta \longrightarrow nat(z)}\ nat\mathcal{R} \qquad \frac{\Delta \longrightarrow nat(x)}{\Delta \longrightarrow nat(s(x))}\ nat\mathcal{R}$$

$$\frac{\longrightarrow Q(z) \quad Q(y) \longrightarrow Q(s(y)) \quad \Delta, Q(x) \longrightarrow P}{nat(x), \Delta \longrightarrow P}\ nat\mathcal{L}$$

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 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. The paper [22] contains a proof that cut-elimination holds for intuitionistic logic extended with both these rules for natural numbers and with stratified definitions. Notice that with this formulation of induction, cut-free proofs will not have the subformula property. We use $\Vdash_{\mathbb{N}}$ to denote provability using both left and right introduction rules for definitions as well as the above mentioned left and right rules for natural numbers. If n is a natural number, we write $\bar{n}$ to denote the corresponding "numeral" for n: that is, $\bar{n}$ is the term containing n occurrences of s and one occurrence of z.

We can now encode $\sqsubseteq_i$ and $\equiv_i$ by using the indexed versions of sim and $bisim$ given by the relations $ssim$ and $sbisim$, respectively, defined in Figure 7. We will denote this definition as $ssims$. The following proposition shows how a proof using induction can yield proofs that do not involve induction.

Proposition 27 *Let (Λ, S, δ) be an ats, and let p and q be members of S. If $ats_2(\delta), ssims \Vdash_{\mathbb{N}} sim(p,q)$ then for every natural number n, $ats_2(\delta), ssims \Vdash ssim(\bar{n},p,q)$. If $ats_2(\delta), ssims \Vdash_{\mathbb{N}} bisim(p,q)$ then for every natural number n, $ats_2(\delta), ssims \Vdash sbisim(\bar{n},p,q)$.*

Proof We prove the first result about simulation: the result about bisimulation is analogous. A cut-free proof of the sequent $\longrightarrow sim(p,q)$ must end in a $def\mathcal{R}$ rule, which (using $\forall\mathcal{R}$ and $\supset\mathcal{R}$ also) means that the sequent $nat(k) \longrightarrow ssim(k,p,q)$ is provable, where k is a variable. Call this

$$
\begin{aligned}
sim(P,Q) &\triangleq \forall K.\ nat(K) \supset ssim(K,P,Q).\\
ssim(z,P,Q) &\triangleq \top\\
ssim(s(K),P,Q) &\triangleq [\forall A \forall P'.\ one(P,A,P') \supset\\
&\qquad\qquad \exists Q'.\ one(Q,A,Q') \wedge ssim(K,P',Q')].\\
bisim(P,Q) &\triangleq \forall K.\ nat(K) \supset sbisim(K,P,Q).\\
sbisim(z,P,Q) &\triangleq \top.\\
sbisim(s(K),P,Q) &\triangleq [\forall A \forall P'.\ one(P,A,P') \supset\\
&\qquad\qquad \exists Q'.\ one(Q,A,Q') \wedge sbisim(K,P',Q')] \wedge\\
&\quad\ [\forall A \forall Q'.\ one(Q,A,Q') \supset\\
&\qquad\qquad \exists P'.\ one(P,A,P') \wedge sbisim(K,Q',P')].
\end{aligned}
$$

Figure 7: The *ssims* definition for indexed simulation and bisimulation. Free variables are assumed to be universally quantified at the top level of clauses.

proof Ξ. Now let n be a natural number. It is possible to substitute $\bar{n}$ for the variable k into the proof Ξ to obtain the proof $\Xi[\bar{n}/k]$ of the sequent $nat(\bar{n}) \longrightarrow ssim(\bar{n},p,q)$. (Such substitution into proofs is not completely trivial: for example, when substituting into an occurrence of $def\mathcal{L}$, some premises many no longer appear in the resulting proof. For details, see [22].) Given that n is a natural number, it is easy to construct a cut-free proof of $\longrightarrow nat(\bar{n})$, one using only the right rules for *nat*. Now placing these two proofs together with a cut rule yields

$$\dfrac{\longrightarrow nat(\bar{n}) \qquad nat(\bar{n}) \longrightarrow ssim(\bar{n},p,q)}{\longrightarrow ssim(\bar{n},p,q)}\ cut$$

Given the cut-elimination result for this logic involving definitions and induction [22], we can conclude that $\longrightarrow ssim(\bar{n},p,q)$ has a cut-free proof. Since the predicate *nat* does not appear in the definition of *ssim* or in any definitional clause on which it relies, the resulting proof does not contain any occurrences of induction. ∎

Proposition 28 *Let (Λ, S, δ) be an ats, p and q be members of S, and n be a natural number. If $ats_2(\delta), ssims \Vdash ssim(\bar{n},p,q)$ then $p \sqsubseteq_n q$. If $ats_2(\delta), ssims \Vdash sbisim(\bar{n},p,q)$ then $p \equiv_n q$.*

Proof We prove the first result about simulation: the result about bisimulation is analogous. Assume that n is 0. Then $p \sqsubseteq_0 q$ holds immediately. Otherwise, let n be $m+1$. Assume that $ats_2(\delta), ssims \Vdash ssim(s(\bar{m}),p,q)$. An analysis of the inference rules in cut-free proofs using the permutation of inference rules described in Lemma 11 shows that for some premise set P, there is a subproof of the sequent $\longrightarrow ssim(\bar{m},p',q')$ for every $(p',q') \in P$. Using the inductive assumption, $p' \sqsubseteq_m q'$ for all $(p',q') \in P$. Hence, by the definition of ϕ_s, we have $p \sqsubseteq_{m+1} q$. ∎

Putting these results together with the one mentioned earlier regarding when ϕ_s and ϕ_b are downward continuous, we can prove the following.

Corollary 29 *If $ats_2(\delta), ssims \Vdash_{\mathbb{N}} sim(p,q)$ then $p \sqsubseteq_\omega q$. If $ats_2(\delta), ssims \Vdash_{\mathbb{N}} bisim(p,q)$ then $p \equiv_\omega q$. If the abstract transition system is finitely branching, then we can conclude the stronger fact that $p \sqsubseteq q$ or $p \equiv q$.*

In this full proof system, it is possible to prove bisimilarity also for non-noetherian ats's. In particular, it is possible to prove that *bisim* is reflexive (and together with Example 23, that *bisim* is an equivalence relation). Below we list some properties that can only be proved by using induction (along with the definitions $ccs(\mathcal{A})$ and *ssims*).

$bisim(\mu_x\ a.x, \mu_x\ (a.x + a.x))$	Example of bisimilar infinite processes
$\forall P\ bisim(P + 0, P)$	0 is neutral element for +
$\forall P\ bisim(P + P, P)$	+ is idempotent
$\forall P, Q\ (bisim(P, Q)) \supset sim(P, Q))$	cf. Example 24
$\forall P\ bisim(P, P)$	Reflexivity, cf. Example 25
$\forall P, Q, R\ (bisim(P, Q) \supset bisim(P + R, Q + R))$	+ preserves *bisim*, cf. Example 26

We leave the construction of the proofs of these theorems to the reader.

8 Conclusion

It has been observed before that intuitionistic and linear logics can be used to specify transition systems. In this paper, we have shown that if 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 relation.

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, $def\mathcal{R}$ is used to establish B and, in the second case, $def\mathcal{L}$ 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, both left and right introduction of definitions are used together. From a computational or concurrency point-of-view, proofs using just $def\mathcal{R}$ 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 $def\mathcal{L}$ inference rule allows capturing certain forms of *must* behavior.

Acknowledgments. We would like to thank Robert Stärk for several helpful discussions and an anonymous referee for many helpful corrections and suggestions. 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. The work of Palamidessi has also been partially supported by the HCM project EXPRESS.

References

[1] K. R. Apt and R. Bol. Logic programming and negation: a survey. *Journal of Logic Programming*, 19-20:9–71, 1994.

[2] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *Journal of the ACM*, 29(3):841–862, 1982.

[3] 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.

[4] R. Burstall and Furio Honsell. A natural deduction treatment of operational semantics. In *Proceedings of the 8th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume LNCS, Vol. 338, pages 250–269. Springer-Verlag, 1988.

[5] Iliano Cervesato and Frank Pfenning. A linear logic framework. In *Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science*, pages 264–275, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press. An extended version of this paper will appear in *Information and Computation*.

[6] Jawahar Chirimar. *Proof Theoretic Approach to Specification Languages*. PhD thesis, University of Pennsylvania, February 1995.

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

[8] Lars-Henrik Eriksson. A finitary version of the calculus of partial inductive definitions. In L.-H. Eriksson, L. Hallnäs, and P. Schroeder-Heister, editors, *Proceedings of the Second International Workshop on Extensions to Logic Programming*, volume 596 of *Lecture Notes in Artificial Intelligence*, pages 89–134. Springer-Verlag, 1991.

[9] 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.

[10] Gerhard Gentzen. Investigations into logical deductions. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland Publishing Co., Amsterdam, 1969.

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

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

[13] Lars Hallnäs. Partial inductive definitions. *Theoretical Computer Science*, 87:115–142, 1991.

[14] 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.

[15] John Hannan. Extended natural semantics. *J. of Functional Programming*, 3(2):123–152, April 1993.

[16] John Hannan and Dale Miller. From operational semantics to abstract machines. *Mathematical Structures in Computer Science*, 2(4):415–459, 1992.

[17] Matthew Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[18] Gérard Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[19] Gilles Kahn. Natural semantics. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, volume 247 of *LNCS*, pages 22–39. Springer-Verlag, March 1987.

[20] Raymond McDowell. *Reasoning in a Logic with Definitions and Induction.* PhD thesis, University of Pennsylvania, December 1997.

[21] Raymond McDowell and Dale Miller. A logic for reasoning with higher-order abstract syntax. In Glynn Winskel, editor, *Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 434–445, Warsaw, Poland, July 1997. IEEE Computer Society Press. An extended version of this paper will appear in the *ACM Transactions on Computational Logic.*

[22] Raymond McDowell and Dale Miller. Cut-elimination for a logic with definitions and induction. *Theoretical Computer Science*, 232:91–119, 2000.

[23] 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.

[24] Dale Miller. Forum: A multiple-conclusion specification language. *Theoretical Computer Science*, 165(1):201–232, September 1996.

[25] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, New York, NY, 1980.

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

[27] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML.* MIT Press, 1990.

[28] Gopalan Nadathur and Dale Miller. An Overview of λProlog. In *Fifth International Logic Programming Conference*, pages 810–827, Seattle, Washington, August 1988. MIT Press.

[29] Lawrence C. Paulson. Isabelle: The next 700 theorem provers. In Piergiorgio Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[30] G. Plotkin. A structural approach to operational semantics. DAIMI FN-19, Aarhus University, Aarhus, Denmark, September 1981.

[31] 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.

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

[33] R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 278–297, Amsterdam, 1990. Springer-Verlag.