Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 229 (2009) 107–118
www.elsevier.com/locate/entcs

Automating Soundness Proofs

Muck van Weerdenburg[1]

Department of Mathematics and Computer Science
Eindhoven University of Technology (TU/e)
Eindhoven, The Netherlands

Abstract

When developing a new language with semantics described by Structural Operational Semantics (SOS), one often wants an axiomatisation of this language (w.r.t. to some equivalence) as well. We describe a method for automating the straightforward soundness proofs for the axioms of such an axiomatisation.

Keywords: structural operational semantics, axiomatisation, soundness, theorem proving

1 Introduction

In the field of process theory new theories are constantly being constructed and each time similar activities occur. The development of a process language typically starts with the definition of the syntax and operational semantics, for instance by means of Structural Operation Semantics (SOS) [17]. Next an equivalence, such as bisimilarity [15,10], is defined on processes. Often an axiomatision for the language is also given. At the very least, such an axiomatisation must be proven sound.

Most of the axioms of an axiomatisation are usually very simple. Typical examples are associativity, commutativity and distribution axioms. The soundness proofs of these axioms are often also very straightforward but quite labour intensive. For this reason it is not uncommon to see claims that such properties hold without proofs. Another approach that is sometimes taken is to simple define operators to have properties such as associativity. In [12] it is shown that doing so can have unexpected and undesired consequences.

As the proofs are indeed very straightforward, an obvious question is whether we can automate such proofs. We are not aware of any attempts that try to do this, however. The closest techniques that come to mind are those of [1,19,20,3] where a

[1] E-Mail: M.J.van.Weerdenburg@tue.nl

doi:10.1016/j.entcs.2009.07.076

sound and complete axiomatisation is constructed from the operational semantics. The down side to these approaches is that they require certain operators and/or the semantics to be in a restricted format.

In this paper we describe a method that we think is capable of proving the most straightforward soundness claims. We translate the semantics and equivalence to axioms in a first-order deduction system and for each axiom that needs to be proven sound we show that it satisfies the equivalence by constructing a deductive proof. We give a proof of concept that (partially) implements our method and gives a more clear feeling of what is and what is not possible with this technique. We expect that our method can easily be adapted for proving symmetry, transitivity and congruence of equivalences w.r.t. to the semantics.

2 Soundness Proofs

We first look at the typical structure of a soundness proof. For this we consider CCS [10] without parallelism, which is defined by the following operational semantics.

$$\frac{}{a.p \xrightarrow{a} p} \qquad \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

Bisimilarity on this language is defined as follows. Let R be a symmetric relation on processes. We call R a *bisimulation relation* if, and only if, for all processes p and q with $p\,R\,q$ we have that if $p \xrightarrow{a} p'$ for some a and p', there is a q' such that $q \xrightarrow{a} q'$ and $p'\,R\,q'$. The latter requirement is referred to as the *transfer condition*. Two processes p and q are *bisimilar*, notation $p \leftrightarrows q$, if, and only if, there is a bisimulation relation R relating p and q.

With bisimilarity we consider the following (complete) axiomatisation.

$$\begin{aligned} x + y &= y + x \\ x + (y + z) &= (x + y) + z \\ x + x &= x \\ x + 0 &= x \end{aligned}$$

Let us prove the second axiom (i.e. $x + (y + z) = (x + y) + z$). This means we need to show that for all processes p, q and r there is a bisimulation relation R relating $p + (q + r)$ and $(p + q) + r$. The relation that is taken in typical soundness proofs is the smallest relation containing the pairs related by the axiom (both ways) and any other required pairs. Often this means that also all pairs (p, p) (for all p) are included. More complex axioms might also need other axioms (e.g. for commutativity of the parallel composition one needs commutativity of the communication merge).

Let $R = \{(p, p), (p + (q + r), (p + q) + r), ((p + q) + r, p + (q + r)) \ :$ "p, q and r are closed terms"$\}$. We know need to show that R is a bisimulation

relation. That is, for each pair in R we must prove the transfer condition. First for the pairs (p, p):

- Assume that $p \xrightarrow{a} p'$ for some a and p'. We then trivially have a q' (viz. p') such that $p \xrightarrow{a} p'$ and $(p', q') \in R$.

Next we consider the pairs $(p + (q + r), (p + q) + r)$, for all p, q and r.

- Assume that $p + (q + r) \xrightarrow{a} p'$ for some a and p'. For the rules of the operational semantics we know this means that either $p \xrightarrow{a} p'$ or $q + r \xrightarrow{a} p'$.
  - If $p \xrightarrow{a} p'$, then the operational semantics gives us that $p + q \xrightarrow{a} p'$ and $(p + q) + r \xrightarrow{a} p'$. As $(p', p') \in R$, we have satisfied the transfer condition for this case.
  - If $q + r \xrightarrow{a} p'$, then the same rules give us that $q \xrightarrow{a} p'$ or $r \xrightarrow{a} p'$.
    If $q \xrightarrow{a} p'$, then the operational semantics gives us that $p + q \xrightarrow{a} p'$ and $(p + q) + r \xrightarrow{a} p'$. As $(p', p') \in R$, we have satisfied the transfer condition for this case.
    If $r \xrightarrow{a} p'$, then the operational semantics gives us that $(p + q) + r \xrightarrow{a} p'$. As $(p', p') \in R$, we have satisfied the transfer condition for this case.

Finally we consider the pairs $((p + q) + r, p + (q + r))$, for all p, q and r.

- Assume that $(p + q) + r \xrightarrow{a} p'$ for some a and p'. For the rules of the operational semantics we know this means that either $p + q \xrightarrow{a} p'$ or $r \xrightarrow{a} p'$.
  - If $p + q \xrightarrow{a} p'$, then the same rules give us that $p \xrightarrow{a} p'$ or $q \xrightarrow{a} p'$.
    If $p \xrightarrow{a} p'$, then the operational semantics gives us that $p + (q + r) \xrightarrow{a} p'$. As $(p', p') \in R$, we have satisfied the transfer condition for this case.
    If $q \xrightarrow{a} p'$, then the operational semantics gives us that $q + r \xrightarrow{a} p'$ and $p + (q + r) \xrightarrow{a} p'$. As $(p', p') \in R$, we have satisfied the transfer condition for this case.
  - If $r \xrightarrow{a} p'$, then the operational semantics gives us that $q + r \xrightarrow{a} p'$ and $p + (q + r) \xrightarrow{a} p'$. As $(p', p') \in R$, we have satisfied the transfer condition for this case.

This concludes the soundness proof for the second axiom. It is clear that all the cases are very much alike and quite straightforward.

The way the above proof goes is very typical. First one takes the assumption (e.g. $p + (q + r) \xrightarrow{a} p'$) and uses the operational semantics to split this in "smaller" statements (i.e. $p \xrightarrow{a} p'$, $q \xrightarrow{a} p'$ and $r \xrightarrow{a} p'$). Once we cannot simplify any further (without case distinction on variables), the obtained statements are combined to get the desired result.

Do note that this last step is actually written down in the reverse order. One does not randomly combine the basic statements obtained from the assumptions in the hope the desired result will appear. What happens – consciously or not – is that one starts at the desired conclusion and simplifies this (much like the assumptions) to the various sets of basic statements that lead to this conclusion. After that it is merely checking whether one of these sets is included in the set of statements derived from the assumptions.

3 Automation

To automate soundness proofs we first must decide on a more formal framework in which we write our proofs. We assume sets of **variables** $\mathbb{V}$, **function symbols** $\mathbb{F}$ and **predicates** $\mathbb{P}$. Combining elements from $\mathbb{V}$ and $\mathbb{F}$ in the usual manner gives us the set of **terms** $\mathbb{T}$. Note that we assume that terms are typed, even though we do not explicitly use it here. A predicate from $\mathbb{P}$ applied to a number of terms is called an **atom**. We use a typical first-order logic. The base elements are atoms and we have the typical operators $\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\forall_x$ and $\exists_x$. We write $\mathrm{FV}(S)$ for the free (i.e. unbound) variables in the formulas in S.

We consider an **SOS rule** to be a tuple $\langle \varphi, a \rangle$, where a is an atom and φ is a conjunction of atoms and formulas of the form $\forall_x(\neg b)$ (with x a variable and b an atom). We assume that each predicate takes at least one argument and that the first argument of such a predicate in the conclusion of a SOS rule is of the form $f(t_1, \ldots, t_n)$. We use the three-valued-stable-model semantics of SOSs (conform [5]), but assume that the set of SOS rules defining the operation semantics has a two-valued model (i.e. the set of "unknowns" is empty). We know of no application where this is not desired and it is typically straightforward to prove using a stratification [5].

Note that for traditional Transition-System Specifications (TSSs) [9] the premises are defined as sets of atoms and negations of atoms. For practical cases these sets are easily represented by a formula using conjunctions (and universal quantifications over negations of atoms). When this is not the case one might desire the infinitary operators $\bigvee$ and $\bigwedge$, but we do not see any reasonable way of implementing this at this moment.

Our method of automation works by formulating soundness claims as formulas and proving these in a sequent-style calculus [8,18]. Such calculi have proven their worth in theorem provers as Coq [4], PVS [14] and Isabelle [16]. In order to be able to use this method we need to formulate the operational semantics, the (equivalence) relation and the (relations representing the) axioms as axioms in the set Ax. We do this as follows.

(i) If R is the set of SOS rules of the operational semantics, we add the following axioms to Ax for all P, f and t_i:

$$P(f(t_1, \ldots, t_n), t_{n+1}, \ldots, t_m) \doteq \bigvee_{\substack{\langle \varphi, a \rangle \in R, \{x_1, \ldots, x_k\} = \mathrm{FV}(\{\varphi, a\}), \\ a = P(f(t'_1, \ldots, t'_n), t'_{n+1}, \ldots, t'_m)}} (\exists_{x_1, \ldots, x_k}(\varphi \wedge \textstyle\bigwedge_{1 \leq i \leq m} t_i = t'_i))$$

(ii) We add a rule $\mathrm{is}_{\mathrm{rel}}(R) = \varphi$ to Ax for each R, where φ is the formula expressing that R satisfies relation rel. For example, $\mathrm{is}_{\leftrightarrow}(R)$ is true if, and only if, R is symmetric and satisfies the transfer condition.

(iii) For each relation R representing an axiom we add a rules $R(t, u) = \varphi$ where

φ expresses precisely when t and u are related by R. More on how to obtain such a relation later.

Note that the above typically results in an infinite set Ax. This is not a problem as an implementation does not explicitly need the set Ax; the elements of this set can be deduced from the finite representation of the semantics and axioms with the rules above.

To illustrate this strategy we again consider CCS without parallelism. For the operational semantics we get the following statements in Ax (per the transformation described above and some simplifications).

$$\begin{array}{lcll} 0 \xrightarrow{a} p & \doteq & \text{false} & \text{for all } a \text{ and } p \\ \text{act}(a) \xrightarrow{b} p & \doteq & a = b \ \wedge \ p = 0 & \text{for all } a, b \text{ and } p \\ p + q \xrightarrow{a} p' & \doteq & p \xrightarrow{a} p' \ \vee \ q \xrightarrow{a} p' & \text{for all } p, q, a \text{ and } p' \end{array}$$

We write the bisimulation relation characterisation as follows. We write $\underline{\leftrightarrow}$ for the predicate that indicates whether a set is a bisimulation relation.

$$\underline{\leftrightarrow}(R) \doteq \forall_{p,q}(R(p,q) \ \Rightarrow \ R(q,p) \ \wedge \ \forall_{a,p'}(p \xrightarrow{a} p' \ \Rightarrow \ \exists_{q'}(q \xrightarrow{a} q' \ \wedge \ R(p',q'))))$$

The bisimulation relation assoc_+ for axiom $x + (y + z) = (x + y) + z$ is defined by the following.

$$\begin{aligned} \text{assoc}_+(p,q) \doteq\ & p = q \vee \\ & \exists_{p',q',r'}(p = p' + (q' + r') \wedge q = (p' + q') + r') \vee \\ & \exists_{p',q',r'}(p = (p' + q') + r' \wedge q = p' + (q' + r')) \end{aligned}$$

Such a relation typically consists of the pairs related by the axiom itself but it also contains additional pairs in order to prove the soundness. For example, for most equivalences and axioms one needs the relation to be reflexive (see the proof in Sect. 2). In some occasions one needs additional non-trivial pairs in order to prove the soundness of an axioms. In that case it would be useful to have some extra logic in the implementation of our method to detect when the proof needs certain pairs in the relation that are not in it (similar to what is done in [6]). By either reporting this or even adding such pairs on the fly, the user can be relieved of the burden to supply these relations in most cases.

The deduction rules of the sequent calculus that we use here are as follows. Note that we only use rules that in some way reduce the "complexity" of statements. This allows us to apply any of the rules without risking infinite behaviour. The rules ($\exists$E), (axI) and (axE) are discussed below. We write Γ, φ and φ, Δ for $\Gamma \cup \{\varphi\}$ and $\{\varphi\} \cup \Delta$, respectively.

$$\frac{}{\Gamma \vdash \text{true}, \Delta}(\text{trueI}) \quad \frac{}{\Gamma, \text{false} \vdash \Delta}(\text{falseE})$$

$$\frac{\Gamma, \varphi \vdash \Delta}{\Gamma \vdash \neg\varphi, \Delta}(\neg\text{I}) \quad \frac{\Gamma \vdash \varphi, \Delta}{\Gamma, \neg\varphi \vdash \Delta}(\neg\text{E})$$

$$\frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \varphi \vee \psi, \Delta}(\vee\text{I}) \quad \frac{\Gamma, \varphi \vdash \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \varphi \vee \psi \vdash \Delta}(\vee\text{E})$$

$$\frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \wedge \psi, \Delta}(\wedge\text{I}) \quad \frac{\Gamma, \varphi, \psi \vdash \Delta}{\Gamma, \varphi \wedge \psi \vdash \Delta}(\wedge\text{E})$$

$$\frac{\Gamma, \varphi \vdash \psi, \Delta}{\Gamma \vdash \varphi \Rightarrow \psi, \Delta}(\Rightarrow\text{I}) \quad \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \psi \vdash \Delta}{\Gamma, \varphi \Rightarrow \psi \vdash \Delta}(\Rightarrow\text{E})$$

$$\frac{\Gamma \vdash \varphi[X/x], \Delta \qquad X \notin \text{FV}(\Gamma \cup \Delta)}{\Gamma \vdash \exists_x(\varphi), \Delta}(\exists\text{I}) \quad \frac{\Gamma, \varphi \vdash \Delta \qquad x \notin \text{FV}(\Gamma \cup \Delta)}{\Gamma, \exists_x(\varphi) \vdash \Delta}(\exists\text{E})$$

$$\frac{\Gamma \vdash \varphi, \Delta \qquad x \notin \text{FV}(\Gamma \cup \Delta)}{\Gamma \vdash \forall_x(\varphi), \Delta}(\forall\text{I})$$

$$\frac{\Gamma \vdash C[u], \Delta \quad t \doteq u \in \text{Ax}}{\Gamma \vdash C[t], \Delta}(\text{axI}) \quad \frac{\Gamma, C[u] \vdash \Delta \quad t \doteq u \in \text{Ax}}{\Gamma, C[t] \vdash \Delta}(\text{axE})$$

$$\frac{\Gamma \vdash t_1 = u_1 \wedge \ldots \wedge t_n = u_n, \Delta}{\Gamma \vdash f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n), \Delta}(\text{eqI}) \quad \frac{\Gamma, t_1 = u_1, \ldots, t_n = u_n \vdash \Delta}{\Gamma, f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n) \vdash \Delta}(\text{eqE}_1)$$

$$\frac{}{\Gamma \vdash t = t, \Delta}(\text{reflexivity}) \quad \frac{f \neq g}{\Gamma, f(t_1, \ldots, t_n) = g(u_1, \ldots, u_m) \vdash \Delta}(\text{eqE}_2)$$

$$\frac{\Gamma[u/x] \vdash \Delta[u/x]}{\Gamma, x = u \vdash \Delta}(\text{substL}) \quad \frac{\Gamma[u/x] \vdash \Delta[u/x]}{\Gamma, u = x \vdash \Delta}(\text{substR})$$

$$\frac{}{\Gamma, \varphi \vdash \varphi, \Delta}(\text{assumption})$$

The following rules deserve special mention:

- Rule ($\exists$E) is special because instead of having the premise $\varphi[t/x]$ as is common in such logics we introduce a *meta variable* X. The reason for this is that we do not wish to pick a specific t at the moment we apply the rule as we do not know

which t will be useful later in the proof. Instead we postpone this decision until we find a value that is suited. For example, from $P(t) \vDash \exists_x(\varphi \Rightarrow P(x))$ we get $P(t) \vDash \varphi \Rightarrow P(X)$ with rule ($\exists$E) and $P(t), \varphi \vdash P(X)$. Only at this point we find that we can only apply (assumption) if $X = t$. Note that this is still no guarantee that the right choice is made, but at least we can base our choice in a more direct way. Also note that there is not explicit rule to instantiate meta-variables as this is done on the meta level.

- Rules (axI) and (axE) assume a set of axioms Ax of the form $\varphi \doteq \psi$. In this set we store the facts about our semantics. More on this later. Note that we typically do not accept axioms that specify recursive relations (e.g. $P(t) \doteq Q(t) \wedge P(f(t))$) as this can result in infinite application of these rules.

Note that the only rule that is really missing here is the elimination rule for the $\forall$. The reason for this is that, similar to ($\exists$E), one does not know beforehand which values to instantiate the bound variable with. The solution of introducing a meta-variable does not suffice here as one might need multiple instantiations for one proof.

The strategy we use for proving is to apply any applicable rule with the restriction that the rule ($\exists$I) can only be applied when no other rule can. The reason for the restriction is that the variables that can be used when replacing a meta-variable with a (more) concrete value are limited to the free variables in the sequent where the meta-variable was introduced. To maximise the freedom in choosing such a value we wish to eliminate as much of existential quantifications from the assumptions and universal quantifications from the targets as possible.

Now that we have formulated the semantics and axioms in our framework, given our derivation rules and defined our strategy, we can prove the soundness of axiom $x + (y + z) = (x + y) + z$ by deriving $\vdash \underline{\leftrightarrow}(\mathrm{assoc}_+)$. This is done in the following fashion.

$$
\begin{array}{l}
\vdash \underline{\leftrightarrow}(\mathrm{assoc}_+) \\
\quad \{\ (\mathrm{axI})\ \} \\
\vdash \forall_{p,q}(\mathrm{assoc}_+(p,q) \ \Rightarrow\ \forall_{a,p'}(p \xrightarrow{a} p' \ \Rightarrow\ \exists_{q'}(q \xrightarrow{a} q' \ \wedge\ \mathrm{assoc}_+(p',q')))) \\
\quad \{\ 2\times(\forall\mathrm{I})\ \} \\
\vdash \mathrm{assoc}_+(p,q) \ \Rightarrow\ \forall_{a,p'}(p \xrightarrow{a} p' \ \Rightarrow\ \exists_{q'}(q \xrightarrow{a} q' \ \wedge\ \mathrm{assoc}_+(p',q'))) \\
\quad \{\ (\Rightarrow\mathrm{I}\ \} \\
\mathrm{assoc}_+(p,q) \vdash \forall_{a,p'}(p \xrightarrow{a} p' \ \Rightarrow\ \exists_{q'}(q \xrightarrow{a} q' \ \wedge\ \mathrm{assoc}_+(p',q'))) \\
\quad \{\ (\mathrm{axE})\ \} \\
p = q \vee \exists_{p',q',r'}(p = p' + (q' + r') \wedge q = (p' + q') + r') \vee \\
\quad \exists_{p',q',r'}(p = (p' + q') + r' \wedge q = p' + (q' + r')) \vdash \\
\qquad \forall_{a,p'}(p \xrightarrow{a} p' \ \Rightarrow\ \exists_{q'}(q \xrightarrow{a} q' \ \wedge\ \mathrm{assoc}_+(p',q')))
\end{array}
$$

Applying ($\vee$E) at this point gives three proof obligations. We only consider the second.

$$\exists_{p',q',r'}(p = p' + (q' + r') \wedge q = (p' + q') + r') \vdash$$
$$\quad \forall_{a,p'}(p \xrightarrow{a} p' \;\Rightarrow\; \exists_{q'}(q \xrightarrow{a} q' \;\wedge\; \mathrm{assoc}_+(p', q')))$$
$$\quad \{\ 3\times(\exists\mathrm{E})\ \}$$
$$p = p' + (q' + r') \wedge q = (p' + q') + r' \vdash$$
$$\quad \forall_{a,p'}(p \xrightarrow{a} p' \;\Rightarrow\; \exists_{q'}(q \xrightarrow{a} q' \;\wedge\; \mathrm{assoc}_+(p', q')))$$
$$\quad \{\ (\wedge\mathrm{E})\ \}$$
$$p = p' + (q' + r'), q = (p' + q') + r' \vdash$$
$$\quad \forall_{a,p'}(p \xrightarrow{a} p' \;\Rightarrow\; \exists_{q'}(q \xrightarrow{a} q' \;\wedge\; \mathrm{assoc}_+(p', q')))$$
$$\quad \{\ (\mathrm{subst})\ \}$$
$$q = (p' + q') + r' \vdash$$
$$\quad \forall_{a,p''}(p' + (q' + r') \xrightarrow{a} p' \;\Rightarrow\; \exists_{q''}(q \xrightarrow{a} q' \;\wedge\; \mathrm{assoc}_+(p'', q'')))$$
$$\quad \{\ (\mathrm{subst})\ \}$$
$$\vdash \forall_{a,p''}(p' + (q' + r') \xrightarrow{a} p' \;\Rightarrow\; \exists_{q''}((p' + q') + r' \xrightarrow{a} q' \;\wedge\; \mathrm{assoc}_+(p'', q'')))$$
$$\quad \{\ 2\times(\forall\mathrm{I})\ \}$$
$$\vdash p' + (q' + r') \xrightarrow{a} p' \;\Rightarrow\; \exists_{q''}((p' + q') + r' \xrightarrow{a} q' \;\wedge\; \mathrm{assoc}_+(p'', q''))$$
$$\quad \{\ (\Rightarrow\mathrm{I})\ \}$$
$$p' + (q' + r') \xrightarrow{a} p' \vdash \exists_{q''}((p' + q') + r' \xrightarrow{a} q' \;\wedge\; \mathrm{assoc}_+(p'', q''))$$
$$\quad \{\ (\mathrm{axE})\ \}$$
$$p' \xrightarrow{a} p' \;\vee\; q' + r' \xrightarrow{a} p' \vdash \exists_{q''}((p' + q') + r' \xrightarrow{a} q'' \;\wedge\; \mathrm{assoc}_+(p'', q''))$$

Applying ($\vee$E) at this point gives two proof obligations. We only consider the first.

$$p' \xrightarrow{a} p' \vdash \exists_{q''}((p' + q') + r' \xrightarrow{a} q'' \;\wedge\; \mathrm{assoc}_+(p'', q''))$$
$$\quad \{\ (\exists\mathrm{I})\ \}$$
$$p' \xrightarrow{a} p' \vdash (p' + q') + r' \xrightarrow{a} X \;\wedge\; \mathrm{assoc}_+(p'', X)$$

Applying ($\wedge$I) at this point gives another two proof obligations. Again, we only consider the first.

$$p' \xrightarrow{a} p' \vdash (p' + q') + r' \xrightarrow{a} X$$
$$\quad \{\ (\mathrm{axI})\ \}$$
$$p' \xrightarrow{a} p' \vdash (p' + q') \xrightarrow{a} X \;\vee\; r' \xrightarrow{a} X$$
$$\quad \{\ (\vee\mathrm{I})\ \}$$
$$p' \xrightarrow{a} p' \vdash (p' + q') \xrightarrow{a} X, r' \xrightarrow{a} X$$
$$\quad \{\ (\mathrm{axI})\ \}$$
$$p' \xrightarrow{a} p' \vdash p' \xrightarrow{a} X \;\vee\; q' \xrightarrow{a} X, r' \xrightarrow{a} X$$
$$\quad \{\ (\vee\mathrm{I})\ \}$$
$$p' \xrightarrow{a} p' \vdash p' \xrightarrow{a} X, q' \xrightarrow{a} X, r' \xrightarrow{a} X$$
$$\quad \{\ \text{Prepare for application of (assumption)}\ \}$$
$$p' \xrightarrow{a} p' \vdash p' \xrightarrow{a} p', q' \xrightarrow{a} p', r' \xrightarrow{a} p'$$

Finally we apply (assumption) and we are done.

Note that our method does has not have many restrictions on the input. Of course, the result of our method does highly depend on the complexity of the input. Mainly the use of universal quantification in premises (e.g. to implement negative transitions) is problematic. This might be solved by introducing special logic to the implementation that, for example, uses pattern matching to determine which instantiations of the variables of such a quantifier might help the proof. Specifically recognising the simple form of $\forall_x(\neg a)$, with a and atom, would extend the applicability of our method to the widely used TSSs with negative premises (in the sense of [5]).

Besides this the most significant restriction is probably the manner in which the (equality) relation has to be represented. Weaker notions than bisimulation, such as trace equivalence, are not easily characterised in this way. However, it is often the case that only a few axioms really need this weaker notion to be proven sound; many axioms hold also with respect to bisimulation. This way we can still prove most axioms in settings with equivalences that are difficult to formalise in our framework. The same holds for weak versions of bisimilarity that need a transitive closure of silent steps (which typically breaks the finite boundary on required calculations).

At the moment all semantics have to be represented as SOS rules. This means that side-conditions on rules also have to be *completely* formalised in SOS rules. For this reason it would be useful to make a separation between SOS rules and auxiliary facts. This is also required for many timed languages where one typically needs the fact that $p \longmapsto q$ and $p \longmapsto r$ means that $q = r$ (where $\longmapsto$ is a time transition).

4 Proof Of Concept

As a proof of concept we have created a program [2] that reads a simple specification file (consisting of the operational semantics, equality definition and axioms) and translates it to a Isabelle/HOL [13] theorem file (including a "tactic" conform Sect. 3 for proving). The latter can be loaded and "run" in Isabelle/HOL which will indicate that the axioms are sound (or not).

The reason we have chosen Isabelle/HOL is that it very closely matches our way of reasoning as described in Sect. 3. Especially the support for using meta-variables (or *unknowns*) is crucial. The main difference is that it has a single formula as right-hand side of $\vdash$ instead of a set of formulas Δ. This means that rule ($\vee$I) requires one to make a choice. Fortunately Isabelle/HOL allows one to make both choices, resulting in multiple derivations.

For simplicity we have chosen to combine all axioms in to one equivalence relation. This avoids the situation where one has to figure out which axioms have to be proven together. Also, we have assumed that there is only one sort. This means, for example, that labels of transitions are the same as processes. As long as the

[2] Available on request.

process semantics is non-trivial (i.e. can be used as a model for the labels as well), this should not be a problem.

To illustrate our proof of concept we show how our running example can be proven sound automatically. First we have a specification of the semantics and axioms. Here `=>` represents the meta implication (i.e. the horizontal line in an SOS rule) and `->` and `&&` are the logical implication and conjunction, respectively. All unbound variables are considered universally quantified.

```
semantics

  # prefix
  true => trans(act(a),a,0());

  # alternative composition
  trans(p,a,p2) => trans(alt(p,q),a,p2);
  trans(q,a,q2) => trans(alt(p,q),a,q2);


relation(p,q)

  # symmetry
  relation(q,p);

  # transfer condition
  trans(p,a,p2) -> <exists q2: trans(q,a,q2) && relation(p2,q2)>;


axioms

  alt(x,y) = alt(y,x);
  alt(x,alt(y,z)) = alt(alt(x,y),z);
  alt(x,x) = x;
  alt(x,0()) = x;

  # auxiliary
  x = x;  # reflexivity is needed for the soundness proofs
```

Running the Isabelle/HOL file typically gives one of three results. If everything is fine and soundness has been proven no errors are given. When there is a problem Isabelle/HOL will give errors indicating the proof could not be completed *or* it will keep running (possibly forever).

For the cases that we have tested so far, a successful result is given in less than a minute. Note that it takes this long because we combine all axioms in on relation. This results in many branches in the proof that are fully explored but do not lead to a positive result (e.g. trying to relate $x+x$ with $(x+y)+z$). With several smaller relations the performance increases significantly (i.e. to at most a few seconds).

Table 1
Number of SOS rules, conjuncts in relation, axioms, auxiliary axioms and proven axioms per language.

Language	#SOS	#rel.	#axs.	#aux.	#proven
CCS	5	2	5	1	6
$\mathrm{BPA}_{\delta\epsilon}$	9	4	9	1	10
BPA^*	13	3	8	3	11
ACP	22	4	20	1	19

In Table 1 we give the results of the specifications we have tested. Here, CCS is from [10], $\mathrm{BPA}_{\delta\epsilon}$ is from [2], BPA^* is from [7] and ACP is from [2]. For CCS we have taken one instantiation of the scheme for the parallel operator axiom. The real axiomatisation has an infinite set of axioms and cannot be represented completely in a finite manner [11]. For BPA^* we have added to additional axioms that are needed to prove axioms BKS2 and BKS3. All specifications include the axiom $x = x$. These additional axioms should not be needed if the implementation itself can detect the additional requirements it needs.

The axioms of ACP that could not be proven sound are the axioms CM1 and CM6. This is because they rely on the commutativity of the communication function γ, which we cannot specify. Note that a minor reformulation of the relevant SOS rules can avoid this problem.

5 Conclusion

We have described a method to automatically prove the soundness of axioms given an operational semantics and order. We have implemented a proof of concept to show that this method can indeed be used to prove soundness for many axioms.

It would be interesting to see if this method can also be used to solve problems other than soundness. We believe that proofs for commutativity, transitivity and congruence proofs should also be possible. Perhaps also settings outside of process theory can benifit from this method.

Also interesting would be a characterisation of the semantics/axioms for which this method is guaranteed to work.

References

[1] L. Aceto, B. Bloom, and F. W. Vaandrager. Turning SOS rules into equations. In *Logic in Computer Science*, pages 113–124, 1992.

[2] J. Baeten and W. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambrdige University Press, 1990.

[3] J. C. M. Baeten and E. P. de Vink. Axiomatizing gsos with termination. In *STACS '02: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 583–595. Springer-Verlag, 2002.

[4] B. Barras, S. Boutin, C. Cornes, J. Courant, J. Filliatre, E. Giménez, H. Herbelin, G. Huet, C. Muñoz, C. Murthy, C. P. C. Parent, A. Saïbi, and B. Werner. The Coq proof assistant reference manual – version v6.1. Technical Report 0203, INRIA, 1997.

[5] R. N. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, 1996.

[6] L. A. Dennis, A. Bundy, and I. Green. Using a generalisation critic to find bisimulations for coinductive proofs. In W. McCune, editor, *Automated Deduction - CADE-14, 14th International Conference on Automated Deduction, Townsville, North Queensland, Australia, July 13-17, 1997, Proceedings*, volume 1249 of *Lecture Notes in Computer Science*, pages 276–290. Springer, 1997.

[7] W. Fokkink. *Clocks, Trees and Stars in Process Theory*. PhD thesis, University of Amsterdam, 1994.

[8] G. Gentzen. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39:176–210,405–431, 1935.

[9] J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[10] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., 1982.

[11] F. Moller. The nonexistence of finite axiomatisations for ccs congruences. In *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 142–153. IEEE Computer Society, 1990.

[12] M. Mousavi and M. Reniers. Congruence for structural congruences. In V. Sassone, editor, *Proceedings of the Eighth International Conference on Foundations of Software Science and Computation Structures (FOSSACS'05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 47–62. Springer-Verlag.

[13] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[14] S. Owre, J. M. Rushby, and N. Shankar. Pvs: A prototype verification system. In *CADE-11: Proceedings of the 11th International Conference on Automated Deduction*, pages 748–752. Springer-Verlag, 1992.

[15] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of the fifth GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer Verlag, 1981.

[16] L. C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[17] G. D. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming*, 60–61:3–15, 2004.

[18] M. E. Szabo. *The Collected Papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North-Holland Publishing Company, 1969.

[19] I. Ulidowski. Axiomatisations of weak equivalences for de simone languages. In *CONCUR '95: Proceedings of the 6th International Conference on Concurrency Theory*, pages 219–233. Springer-Verlag, 1995.

[20] I. Ulidowski. Finite axiom systems for testing preorder and de simone process languages. *Theoretical Computer Science*, 239(1):97–139, 2000.