On Automation of OTS/CafeOBJ Method

Daniel Găină[1], Dorel Lucanu[2], Kazuhiro Ogata[1], and Kokichi Futatsugi[1]

[1] Japan Advanced Institute of Science and Technology (JAIST),
{daniel,ogata,futatsugi}@jaist.ac.jp
[2] Alexandru Ioan Cuza University
dlucanu@info.uaic.ro

Abstract. The proof scores method is an interactive verification method in algebraic specification that combines manual proof planning and reduction (automatic inference by rewriting). The proof score approach to software verification coordinates efficiently human intuition and machine automation. We are interested in applying these ideas to transition systems, more concretely, in developing the so-called OTS/CafeOBJ method, a modelling, specification, and verification method of observational transition systems. In this paper we propose a methodology that aims at developing automatically proof scores according to the rules of an entailment system. The proposed deduction rules include a set of generic rules, which can be found in other proof systems as well, together with a set of rules specific to our working context. The methodology is exhibited on the example of the alternating bit protocol, where the unreliability of channels is faithfully specified.

1 Introduction

This paper is focused on developing the OTS/CafeOBJ method, a modeling, specification and verification method of Observational Transition Systems (OTS), which has been previously explored in many case studies [22,8,21,7]. The logical framework used to develop the methodology is that of constructor-based order-sorted preorder algebra. The signatures are enhanced with a set of constructor operators, the sorts of constructors are called *constrained*, and the sorts that are not constrained are called *loose*. The models are those algebras that are reachable w.r.t. given constructors [1]. For example, given an algebraic signature (S, F), where S is the set of sorts and F is the family of function symbols, an (S, F)-algebra A is reachable w.r.t constructors $F^c \subseteq F$ if for any element $a \in A$ there exists a set Y of variables of loose sorts, an evaluation $f : Y \to A$, and a *constructor term* $t \in T_{(S,F^c)}(Y)$ such that $\overline{f}(t) = a$, where $\overline{f} : T_{(S,F^c)}(Y) \to A$ is the unique extension of $f : Y \to A$ to a (S, F^c)-morphism. The formulas consist of universally quantified conditional atoms, where the atomic sentences are of three types: equations, membership and preorder axioms. [3]

The advantages provided by the expressiveness of constructor-based logics have been previously explored in [2,1], where a method for proving coinductive

[3] In Maude literature preorder axioms are known as rewrite rules.

properties is presented. We propose a methodology for proving inductive properties of OTS specified with constructor-based logics. Only universally quantified conditional sentences are considered, a restriction that makes it possible to use term rewriting to verify system properties. We are interested in reaching a greater level of automation than in [21,8,7,22] by designing proof rules meant to be used for developing more complex proof tactics, which are implemented in Constructor-based Inductive Theorem Prover (CITP) [11]. In comparison with the previous approaches to verifying OTS, the proof scores consist of simple CITP commands. This has the advantage of making the proofs shorter and allowing automated reasoning. The present paper presents the verification methodology supported by CITP.

In [17] a subset of authors give a set of proof rules for constructor-based logics at the abstract level of institutions [13], and a quasi-completeness result is proved. [4] In [9], the proof rules are lifted up at the level of specifications such that quasi-completeness is preserved for the specifications with loose semantics denotation. [5] The entailment relation is constructed as follows: a basic entailment relation between specifications and atomic sentences is assumed which, in applications, is given by the system that assists the proof, for example CafeOBJ [5] or Maude [4]. This relation is then extended with proof rules for the quantification over variables of both constrained and loose sorts, logical implication, and case analysis.

In applications, the specifications are often declared with initial semantics (see [10] for details about the initial semantics in logics with constructors). Roughly speaking, less models means more properties to prove, which sometimes require new inference rules. In order to make the specification calculus defined in [9] effective in practice, it must be enriched with specialized proof rules for the initial data types that are often used in our methodology, and supported by proof tactics that can often be completed automatically. A first such enrichment is proposed in this paper.

In our approach, a goal $\mathtt{SP} \vdash E$ consists of a specification $\mathtt{SP}$ and a set of formulas E rather than a single formula. By applying a proof rule to a goal $\mathtt{SP} \vdash E$ we obtain a set of goals $\mathtt{SP}_1 \vdash E_1, \ldots, \mathtt{SP}_n \vdash E_n$ if some preconditions are satisfied. If it is not the case, the proof rule leaves the goal unchanged. This slightly general view captures a natural phenomenon related to verification: not only the formula of the initial goal is changing in the proof process but also the specification. It is crucial for automation to design proof tactics that preserve the confluence and termination properties in the proof process. Below we describe the contributions to the development of OTS/CafeOBJ method:

(1) The simplest tactics implement the proof rules of the specification calculus. We revise the entailment system defined in [9] to increase its efficiency in

[4] Some proof rules contain infinite premises which can only be checked with induction schemes. As a consequence, the resulting entailment system is not compact.

[5] Loose semantics is meant to capture all models that satisfy the axioms of a specification while initial semantics describes the initial (or standard) model of the axioms of a specification.

applications: we propose a more general *simultaneous induction* scheme, and we refine *case analysis* such that it can be applied automatically. The entailment system obtained is then enriched with *specialised proof rules* for the predefined data types declared with initial semantics.

(2) We define also *derived inference rules* which are built by combination of other tactics. For example, each of the proof rules is coupled with the reduction of the ground terms occurring in the formulas to prove to their normal forms. This has the advantage of preserving the confluence property of the specifications. We define a simple but efficient tactic to avoid non-termination processes during verification. The underlying assumption is the existence of a `BOOL` specification of boolean values which are protected since they are used to establish the truth. This tactic reduces a goal of the form $\mathtt{SP} \vdash \mathtt{t} = \mathtt{t}'$ if $\mathtt{C} \wedge \mathtt{b} = \mathtt{not\ b} \wedge \mathtt{C}'$ to the empty goal set, where `b` is a boolean ground term.

(3) We propose a *proof strategy* to build automatically a complete proof of a goal, which basically establishes an application order of the "basic" tactics. This proof procedure is closely linked to term rewriting and it preserves the confluence property of specifications during verification.

The strength of the proposed methodology is exhibited on a non-trivial case study, the alternating bit protocol (ABP). The unreliability of the communication channels is faithfully modelled by specifying dropping of elements in arbitrary positions of the communication channels. This technique of modelling non-determinism with underspecified operators and then exploiting that in theorem proving is possible due to the expressivity of constructor-based logics.

Structure of the paper. In Section 2, we define the proof rules of the method and the proof strategy. In Section 3.1, we specify the ABP with unreliable communication channels; the verification methodology is explained by proving a safety property for ABP. In Section 4, we summarise the conclusions and we give some future directions for research.

2 Proving Methodology

In this section we revise the entailment system in [9], and we enrich it with specialised proof rules for the initial data types that are often used in our methodology. We also propose tactics to construct proof scores automatically.

2.1 The Underlying Logic

We describe the logical framework underlying the verification methodology. The logic presented here is more expressive than the one in [9] as it includes membership and preorder axioms besides equations.

*Order-Sorted Algebra (***OSA***) [20].* An *order-sorted signature* is a triple $(S, \leq, F)$ with $(S, \leq)$ a preorder, i.e. reflexive and transitive, and (S, F) a many-sorted

signature. Let $\widehat{S} = S/_{\equiv_\leq}$ be the set of connected components of S under the equivalence relation $\equiv_\leq$ generated by $\leq$. The equivalence $\equiv_\leq$ can be extended to sequences in the usual way. An order-sorted signature is called *sensible* if for any two operators $\sigma : w \to s$ and $\sigma : w' \to s'$ such that $w \equiv_\leq w'$ we have $s \equiv_\leq s'$. Hereafter, we assume that all signatures are sensible.

An *order-sorted signature morphism* $\varphi : (S, \leq, F) \to (S', \leq', F')$ is a many-sorted signature morphism $\varphi : (S, F) \to (S', F')$ which

(1) preserves subsort overloading (i.e. for all $\sigma \in F_{w\to s} \cap F_{w'\to s'}$ with $w \equiv_\leq w'$ we have $\varphi^{op}_{(w,s)}(\sigma) = \varphi^{op}_{(w',s')}(\sigma)$, where $\varphi^{op}_{(w,s)} : F_{w\to s} \to F'_{\varphi(w)\to\varphi(s)}$), and such that
(2) $\varphi^{st} : (S, \leq) \to (S', \leq')$ is monotonic.

An *order-sorted Σ-algebra* M, where $\Sigma = (S, \leq, F)$, is a many-sorted (S, F)-algebra such that $s \leq s'$ implies $M_s \subseteq M_{s'}$, and for all $\sigma \in F_{w\to s} \cap F_{w'\to s'}$ with $w \equiv_\leq w'$ and $m \in M^w \cap M^{w'}$ we have $M_{\sigma:w\to s}(m) = M_{\sigma:w'\to s'}(m)$. For each connected component $[s] \in \widehat{S}$ we let $M_{[s]}$ denote the set $\bigcup_{s'\in[s]} M_{s'}$. An *order-sorted Σ-homomorphism* $h : M \to N$ is a many-sorted (S, F)-homomorphism such that for all $s \equiv_\leq s'$ and $m \in M_s \cap M_{s'}$ we have $h_s(m) = h_{s'}(m)$. This defines a category $\mathbb{M}od^{\mathbf{OSA}}(\Sigma)$.

Proposition 1. *[20] The category $\mathbb{M}od^{\mathbf{OSA}}(\Sigma)$ has an initial term algebra T_Σ defined as follows:*

- *if $(w, s) \in S^* \times S$, $\sigma \in F_{w\to s}$ and $t \in (T_\Sigma)^w$ then $\sigma(t) \in (T_\Sigma)_s$,*
- *if $s_0 \leq s$, $t \in (T_\Sigma)_{s_0}$ then $t \in (T_\Sigma)_s$.*

*Order-Sorted Preorder Algebra (***OSPA***) [6].* An *order-sorted preorder Σ-algebra* M, where $\Sigma = (S, \leq, F)$, is an order-sorted algebra with an additional preorder structure $(M_{[s]}, \leq_{[s]})$ for each connected component $[s] \in \widehat{S}$. An *order-sorted preorder Σ-homomorphism* $h : M \to N$ is an order-sorted homomorphism which preserves the preorder structure. This defines a category $\mathbb{M}od^{\mathbf{OSPA}}(\Sigma)$.

A signature morphism $\varphi : (S, \leq, F) \to (S', \leq', F')$ induces a forgetful functor $\mathbb{M}od^{\mathbf{OSPA}}(\varphi) : \mathbb{M}od^{\mathbf{OSPA}}(S', \leq', F') \to \mathbb{M}od^{\mathbf{OSPA}}(S, \leq, F)$ defined as follows:

- for each order-sorted preorder algebra $M' \in |\mathbb{M}od^{\mathbf{OSPA}}(S', \leq', F')|$,
  - $\mathbb{M}od^{\mathbf{OSPA}}(\varphi)(M')_s = M'_{\varphi(s)}$ for all $s \in S$,
  - $\mathbb{M}od^{\mathbf{OSPA}}(\varphi)(M')_\sigma = M'_{\varphi(\sigma)}$ for all $(w, s) \in S^* \times S$ and $\sigma \in F_{w\to s}$,
  - $m_1 \leq_{[s]} m_2$ whenever $m_1 \leq_{[\varphi(s)]} m_2$ for all $[s] \in \widehat{S}$ and $m_1, m_2 \in \mathbb{M}od^{\mathbf{OSPA}}(\varphi)(M)_{[s]}$.
- for each order-sorted preorder homomorphism $h' \in \mathbb{M}od^{\mathbf{OSPA}}(\Sigma')$,
  - $\mathbb{M}od^{\mathbf{OSPA}}(\varphi)(h')_s = h'_{\varphi(s)}$ for all $s \in S$.

We denote by $\_{\restriction_\varphi}$ the functor $\mathbb{M}od^{\mathbf{OSPA}}(\varphi)$. If $M'{\restriction_\varphi} = M$ then we say that M' is a φ-expansion of M, and M is the φ-reduct of M. If φ is a signature inclusion then we may write $M'{\restriction_{(S,\leq,F)}}$ instead of $M{\restriction_\varphi}$.

For each order-sorted signature $\Sigma = (S, \leq, F)$ there are three kinds of atomic sentences:

(1) equational atoms $t = t'$, where $t, t' \in (T_\Sigma)_{[s]}$ and $[s] \in \widehat{S}$,
(2) membership atoms $t : s$, where $t \in (T_\Sigma)_{[s]}$ and $[s] \in \widehat{S}$,
(3) preorder atoms $t \Rightarrow t'$, where $t, t' \in (T_\Sigma)_{[s]}$ and $[s] \in \widehat{S}$.

The set $\mathbb{S}en^{\mathbf{OSPA}}(\Sigma)$ of sentences consists of universally quantified conditional atoms of the form $(\forall X)\mathtt{atm}$ if $\mathtt{atm}_1 \wedge \ldots \wedge \mathtt{atm}_n$, where X is a finite set of variables for Σ, and $\mathtt{atm}, \mathtt{atm}_i$ are atoms. Each order-sorted signature morphism $\varphi : \Sigma \to \Sigma'$ determines a function $\mathbb{S}en^{\mathbf{OSPA}}(\varphi) : \mathbb{S}en^{\mathbf{OSPA}}(\Sigma) \to \mathbb{S}en^{\mathbf{OSPA}}(\Sigma')$ which translates the sentences symbol-wise. When there is no danger of confusion we denote $\mathbb{S}en^{\mathbf{OSPA}}(\varphi)$ simply by φ.

The satisfaction of a sentence by a model $M \in \mathbb{M}od^{\mathbf{OSPA}}(\Sigma)$, where $\Sigma = (S, \leq, F)$, is defined by induction on the structure of the sentences:

- $M \models_\Sigma t = t'$ iff $M_t = M_{t'}$,
- $M \models_\Sigma t : s$ iff $M_t \in M_s$,
- $M \models_\Sigma t \Rightarrow t'$ iff $M_t \leq_{[s]} M_{t'}$,
- $M \models_\Sigma \mathtt{atm}$ if $\mathtt{atm}_1 \wedge \ldots \wedge \mathtt{atm}_n$ iff $M \models_\Sigma \mathtt{atm}_i$ for all $i \in \{1, \ldots, n\}$ implies $M \models_\Sigma \mathtt{atm}$,
- $M \models_\Sigma (\forall X)\rho$ iff for all ι_X-expansions M' of M we have $M' \models_{\Sigma[X]} \rho$.

where $t, t' \in (T_\Sigma)_{[s]}$ are terms, $s \in S$ is a sort, $\mathtt{atm}$ if $\mathtt{atm}_1 \wedge \ldots \wedge \mathtt{atm}_n \in \mathbb{S}en^{\mathbf{OSPA}}(\Sigma)$ is a quantifier-free sentence, $(\forall X)\rho \in \mathbb{S}en^{\mathbf{OSPA}}(\Sigma)$ is any sentence, $\iota_X : \Sigma \hookrightarrow \Sigma[X]$ is the extension of Σ with constants from X.

Proposition 2 (The satisfaction condition). *For all signature morphisms $\varphi : \Sigma \to \Sigma'$, models $M' \in \mathbb{M}od^{\mathbf{OSPA}}(\Sigma')$ and sentences $\rho \in \mathbb{S}en^{\mathbf{OSPA}}(\Sigma)$ we have $M'|_\varphi \models_\Sigma \rho$ iff $M' \models_{\Sigma'} \varphi(\rho)$.*

Proof. Straightforward, by induction on the structure of the sentences. □

*Constructor-based Order-Sorted Preorder Algebra (***COSPA***).* We apply the ideas from [17] to define the constructor-based version of **OSPA**. A *constructor-based order-sorted signature* $(S, \leq, F, F^c)$ consists of an order-sorted signature $(S, \leq, F)$ and a subfamily of sets of operation symbols $F^c \subseteq F$. A sort $s \in S$ is *constrained* if there exists $w \in S^*$ such that $F^c_{w \to s} \neq \emptyset$. Let S^c be the set of constrained sorts and $S^l = S - S^c$ the set of *loose* sorts. $M \in \mathbb{M}od^{\mathbf{OSPA}}(\Sigma)$ is *reachable* w.r.t. the constructors in F^c if there exists a function $f : Y \to M$, where Y is a set of variables of loose sorts, such that $f^\#_s : (T_{(S, \leq, F^c)}(Y))_s \to M_s$ is surjective for all $s \in S^c$, where $f^\# : T_{(S, \leq, F^c)}(Y) \to M|_{(S, \leq, F^c)}$ is the unique extension of f to a $(S, \leq, F^c)$-homomorphism. The notion of reachability generalises the one in [1] to the order-sorted case. Let $\mathbb{M}od^{\mathbf{COSPA}}(S, \leq, F, F^c) \subseteq \mathbb{M}od^{\mathbf{OSPA}}(S, \leq, F)$ be the full subcategory of reachable order-sorted preorder algebras.

A *constructor-based order-sorted signature morphism* $\varphi : (S, \leq, F, F^c) \to (S', \leq', F', F'^c)$ consists of an order-sorted signature morphism $\varphi : (S, \leq, F) \to (S', \leq', F')$ such that

(1) constructors are preserved along the signature morphisms, i.e. if $\sigma \in F^c$ then $\varphi(\sigma) \in F'^c$,

(2) no "new" constructors are introduced for "old" constrained sorts, i.e. if $s \in S^c$ and $\sigma' \in (F'^c)_{w' \to \varphi(s)}$ then there exists $\sigma \in F^c_{w \to s}$ such that $\varphi(\sigma) = \sigma'$, and
(3) if $s'_0 \in S'$ and $s \in S^c$ such that $s'_0 \leq' \varphi(s)$ then there exists $s_0 \in S$ such that $s_0 \leq s$ and $\varphi(s_0) = s'_0$.

Proposition 3. *For all constructor-based order-sorted signature morphisms* $\varphi : (S, \leq, F, F^c) \to (S', \leq', F', F'^c)$, $M' \in |\mathbb{M}od^{\mathbf{COSPA}}(S', \leq, F', F'^c)|$ *implies* $M'|_\varphi \in |\mathbb{M}od^{\mathbf{COSPA}}(S, \leq, F, F^c)|$.

Proof. Let $\varphi^c : (S, \leq, F^c) \to (S', \leq, F'^c)$ be the restriction of φ to constructors. It suffices to prove that for all sets Y' of variables for $(S', \leq', F'^c, F')$ of loose sorts there exists a set Y of variables for $(S, \leq, F, F^c)$ of loose sorts and an assignment $f : Y \to T_{(S', \leq', F'^c)}(Y')|_{\varphi^c}$ such that the unique extension $f^\# : T_{(S, \leq, F^c)}(Y) \to T_{(S', \leq', F'^c)}(Y')|_{\varphi^c}$ of f to a $(S, \leq, F^c)$-homomorphism is a surjection.

Let Y' be a set of loose variables for $(S', \leq', F', F'^c)$. We define $f : Y \to T_{(S', \leq', F'^c)}(Y')|_{\varphi^c}$ as follows:

- for all $s \in S^c$ let $Y_s = \emptyset$.
- for all $s \in S^l$ let Y_s be a set of fresh variables such that
  - if $\varphi(s) \in S'^l$ then there exists a bijection $f_s : Y_s \to Y'_{\varphi(s)}$, and
  - if $\varphi(s) \in S'^c$ then there exists a bijection $f_s : Y_s \to T_{(S', \leq', F'^c)}(Y')_{\varphi(s)}$.

We prove by induction on the structure of the terms $t' \in (T_{(S', \leq', F'^c)}(Y'))_{s'}$ that if $s' \in \varphi(S)$ then for all $s \in \varphi^{-1}(s')$ there exists $t \in (T_{(S, \leq, F)}(Y))_s$ such that $f^\#_s(t) = t'$.

(1) **For** $y' \in Y'_{s'}$**:** Assume that $s' \in \varphi(S)$, and fix $s \in \varphi^{-1}(s')$. Then take $t = f^{-1}(y')$.
(2) **For** $\sigma'(t') \in T_{(S', \leq', F'^c)}(Y')_{s'}$**:** Assume that $s' \in \varphi(S)$, and fix $s \in \varphi^{-1}(s')$. There exists $w \in S^*$ and $\sigma \in F_{w \to s}$ such that $\varphi(\sigma) = \sigma'$. By the induction hypothesis, there exists $t \in (T_{(S, \leq, F)}(Y))^w$ such that $f^\#(t) = t'$. It follows that $f^\#(\sigma(t)) = \sigma'(t')$.
(3) **For** $s'_0 \leq' s'$ **and** $t' \in T_{(S', \leq', F'^c)}(Y')_{s'_0}$**:** Assume that $s' \in \varphi(S)$, and fix $s \in \varphi^{-1}(s')$. There are two cases.
   (a) **For** $s \in S^c$**:** There exists $s_0 \in S$ such that $s_0 \leq s$ and $\varphi(s_0) = s'_0$. By the induction hypothesis, there exists $t \in T_{(S, \leq, F^c)}(Y)_{s_0}$ such that $f^\#(t) = t'$. It follows that $t \in T_{(S, \leq, F^c)}(Y)_s$ and we have $f^\#(t) = t'$.
   (b) **For** $s \in S^l$**:** it follows easily by the definition of f.

□

Hereafter, we work within the context of **COSPA**, which is the underlying logic of the methodology presented in this paper. For the sake of simplicity, we will make the following notations.

Notation 1 *For all constructor-based order-sorted signatures* $(S, \leq, F, F^c)$,

1. $\mathbb{S}en(S, \leq, F, F^c) = \mathbb{S}en^{\mathbf{OSPA}}(S, \leq, F)$ *and*

2. $\mathbb{M}od(S, \leq, F, F^c) = \mathbb{M}od^{\mathbf{COSPA}}(S, \leq, F, F^c)$.

For all constructor-based order-sorted signature morphisms $\varphi : (S, \leq, F, F^c) \to (S', \leq', F', F'^c)$,

1. $\mathbb{S}en(\varphi) = \mathbb{S}en^{\mathbf{OSPA}}(\varphi)$, *and*
2. $\mathbb{M}od(\varphi)$ *is the restriction of the functor*

$$\mathbb{M}od^{\mathbf{OSPA}}(\varphi) : \mathbb{M}od^{\mathbf{OSPA}}(S', \leq', F') \to \mathbb{M}od^{\mathbf{OSPA}}(S, \leq, F)$$

to the subcategory $\mathbb{M}od^{\mathbf{COSPA}}(S', \leq', F', F'^c)$.

For all $M \in |\mathbb{M}od(S, \leq, F, F^c)|$ *and* $\rho \in \mathbb{S}en(S, \leq, F, F^c)$ *we write* $M \models_{(S,\leq,F,F^c)} \rho$ *instead of* $M \models_{(S,\leq,F)} \rho$. *When there is no danger of confusion we drop the subscript* $(S, \leq, F, F^c)$ *from* $\models_{(S,\leq,F,F^c)}$, *and we write simply* $M \models \rho$.

The following corollary of Propositions 2 and 3 says that **COSPA** defined above is an institution [13].

Corollary 1. *For all signature morphisms* $\varphi : (S, \leq, F, F^c) \to (S', \leq', F', F'^c)$, *models* $M' \in \mathbb{M}od(S', \leq', F', F'^c)$ *and sentences* $\rho \in \mathbb{S}en(S, \leq, F, F^c)$ *we have* $M'|_\varphi \models_{(S,\leq,F,F^c)} \rho$ *iff* $M' \models_{(S',\leq',F',F'^c)} \varphi(\rho)$.

A *substitution* of Σ-terms with variables in Y for variables in X, where $\Sigma = (S, \leq, F, F^c)$, is a S-sorted function $\theta : X \to T_\Sigma(Y)$. Note that θ can be canonically extended to $\theta^{term} : T_\Sigma(X) \to T_\Sigma(Y)$ and $\theta^{sen} : \mathbb{S}en(\Sigma[X]) \to \mathbb{S}en(\Sigma[Y])$, where $\Sigma[X]$ and $\Sigma[Y]$ are the extensions of Σ with (non-constructor) constants from X and Y, respectively. When there is no danger of confusion we may drop the superscripts *term* and *sen* from notations. On the semantic side θ, determines a forgetful functor $\_|_\theta : \mathbb{M}od(\Sigma[Y]) \to \mathbb{M}od(\Sigma[X])$ such that for all $M \in \mathbb{M}od(\Sigma[Y])$, $M|_\theta$ interprets all symbols in Σ as M, and $(M|_\theta)_x = M_{\theta(x)}$ for all $x \in X$.

Proposition 4 (The satisfaction condition for substitutions). *For all substitutions* $\theta : X \to T_\Sigma(Y)$, *sentences* $\rho \in \mathbb{S}en(\Sigma[X])$ *and models* $M \in |\mathbb{M}od(\Sigma[Y])|$ *we have* $M|_\theta \models_{\Sigma[X]} \rho$ *iff* $M \models_{\Sigma[Y]} \theta(\rho)$.

Proof. Straightforward, by induction on the structure of the sentences. □

Notation 2 *Given* $t \in T_\Sigma(X)$ *and* $\theta : X \to T_\Sigma(Y)$ *such that* $X = \{x_1, \ldots, x_n\}$ *and* $\theta(x_i) = t_i$ *for all* $i \in \{1, \ldots, n\}$ *then we may write the term* $\theta(t)$ *in the form* $t[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$.

2.2 General Proof Rules

The entailment system in [9] is generalised to **COSPA**. In addition, we propose a *simultaneous induction* scheme which is more general than the structural induction [9], and *case analysis* is refined such that it can be applied automatically.

A specification $\mathtt{SP}$ consists of a signature $\mathbb{S}ig(\mathtt{SP})$, a set of sentences $\mathbb{A}x(\mathtt{SP}) \subseteq \mathbb{S}en(\mathbb{S}ig(\mathtt{SP}))$, and a class of models $\mathbb{M}od(\mathtt{SP}) \subseteq \mathbb{M}od(\mathbb{S}ig(\mathtt{SP}))$ such that $M \models \mathbb{A}x(\mathtt{SP})$ for all $M \in |\mathbb{M}od(\mathtt{SP})|$. A specification morphism $\varphi : \mathtt{SP}_1 \to \mathtt{SP}_2$ consists of a signature morphism $\varphi : \mathbb{S}ig(\mathtt{SP}_1) \to \mathbb{S}ig(\mathtt{SP}_2)$ such that

(a) $\mathbb{A}x(\mathtt{SP}_2) \models \varphi(\mathbb{A}x(\mathtt{SP}_1))$, and
(b) for all $M_2 \in |\mathbb{M}od(\mathtt{SP}_2)|$ we have $M_2{\restriction_\varphi} \in |\mathbb{M}od(\mathtt{SP}_1)|$.

This defines a category of specifications SPEC. Below we recall some of the specification-building operators introduced in [23].

Basic Specification. Any pair (Σ, E) of signature Σ and set of sentences E is a basic specification with the signature $\mathbb{S}ig(\Sigma, E) = \Sigma$, set of sentences $\mathbb{A}x(\Sigma, E) = E$, and class of models $\mathbb{M}od(\Sigma, E)$ consisting of all Σ-models which satisfy E.

Sum. For any specifications $\mathtt{SP}_1$ and $\mathtt{SP}_2$ such that $\mathbb{S}ig(\mathtt{SP}_1) = \mathbb{S}ig(\mathtt{SP}_2)$, $\mathtt{SP}_1 \cup \mathtt{SP}_2$ is a specification such that $\mathbb{S}ig(\mathtt{SP}_1 \cup \mathtt{SP}_2) = \mathbb{S}ig(\mathtt{SP}_1)$, $\mathbb{A}x(\mathtt{SP}_1 \cup \mathtt{SP}_2) = \mathbb{A}x(\mathtt{SP}_1) \cup \mathbb{A}x(\mathtt{SP}_2)$, and $\mathbb{M}od(\mathtt{SP}_1 \cup \mathtt{SP}_2) = \mathbb{M}od(\mathtt{SP}_1) \cap \mathbb{M}od(\mathtt{SP}_2)$.

Translation. Let SP be a specification and $\varphi : \mathbb{S}ig(\mathtt{SP}) \hookrightarrow \Sigma$ a signature morphism. $\mathtt{SP} * \varphi$ is a specification such that $\mathbb{S}ig(\mathtt{SP} * \varphi) = \Sigma$, $\mathbb{A}x(\mathtt{SP} * \varphi) = \varphi(\mathbb{A}x(\mathtt{SP}))$, and $\mathbb{M}od(\mathtt{SP} * \varphi)$ consisting of all Σ-models M' such that the reduct of M' along φ is a SP-model, in symbols, $M'{\restriction_\varphi} \in \mathbb{M}od(\mathtt{SP})$.

Initiality. Given a class $\mathcal{H}$ of model morphisms, for any two specifications $\mathtt{SP}_0$ and SP and any signature morphism $\varphi : \mathbb{S}ig(\mathtt{SP}_0) \to \mathbb{S}ig(\mathtt{SP})$, the *free restriction* of SP to $\mathtt{SP}_0$ through φ, denoted $\mathtt{SP}!_{\mathcal{H}}(\varphi, \mathtt{SP}_0)$, is a specification such that $\mathbb{S}ig(\mathtt{SP}!_{\mathcal{H}}(\varphi, \mathtt{SP}_0)) = \mathbb{S}ig(\mathtt{SP})$, $\mathbb{A}x(\mathtt{SP}!_{\mathcal{H}}(\varphi, \mathtt{SP}_0)) = \mathbb{A}x(\mathtt{SP})$, and
$\mathbb{M}od(\mathtt{SP}!_{\mathcal{H}}(\varphi, \mathtt{SP}_0)) = \{M \in \mathbb{M}od(\mathtt{SP}) \mid$ there exists $M_0 \in \mathbb{M}od(\mathtt{SP}_0)$ and
$\eta : M_0 \to M{\restriction_\varphi} \in \mathcal{H}$ such that for all
$h_0 : M_0 \to N{\restriction_\varphi}$ with $N \in \mathbb{M}od(\mathtt{SP})$
there exists a unique arrow
$h : M \to N$ that satisfies $\eta; h{\restriction_\varphi} = h_0\}$

If $\mathcal{H}$ consists of identity morphisms then no "junk" and no "confusion" is added to the models of $\mathtt{SP}_0$. In this case we say that $\mathtt{SP}_0$ is imported by SP in *protecting mode*. Importations in protecting mode can be realized also by the operator *Translation*.

Definition 3. *[3] An entailment system for deducing the logical consequences of specifications is a family of predicates* $\vdash \stackrel{def}{=} \{\mathtt{SP} \vdash \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ *on the sets of sentences with the following properties:*

$$[lemma]\frac{\mathtt{SP} \cup (\mathbb{S}ig(\mathtt{SP}), E_0) \vdash E \;\; \mathtt{SP} \vdash E_0}{\mathtt{SP} \vdash E}$$

$$[union]\frac{\mathtt{SP} \vdash E_0 \;\; \mathtt{SP} \vdash E}{\mathtt{SP} \vdash E_0 \cup E} \qquad [trans]\frac{\mathtt{SP} \vdash E \text{ and } \varphi : \mathbb{S}ig(\mathtt{SP}) \to \Sigma}{\mathtt{SP} * \varphi \vdash \varphi(E)}$$

$$[sum1]\frac{\mathtt{SP}_1 \vdash \Gamma}{\mathtt{SP}_1 \cup \mathtt{SP}_2 \vdash \Gamma} \qquad [sum2]\frac{\mathtt{SP}_2 \vdash \Gamma}{\mathtt{SP}_1 \cup \mathtt{SP}_2 \vdash \Gamma}$$

where $\mathtt{SP}, \mathtt{SP}_1, \mathtt{SP}_2 \in |\mathtt{SPEC}|$, $\Gamma \subseteq \mathbb{S}en(\mathbb{S}ig(\mathtt{SP}_1))$, *and* $E_0, E \subseteq \mathbb{S}en(\mathbb{S}ig(\mathtt{SP}))$.

Definition 4. *[3] An entailment system* $\vdash = \{\mathtt{SP} \vdash \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ *is* sound *if* $\mathtt{SP} \vdash E$ *implies* $\mathtt{SP} \models E$ *for all specifications* SP *and sets of sentences* E.

Below we define the proof rules that support our verification methodology.

Simultaneous Induction [SI]. This proof rule is a generalization of the structural induction, and it can be applied to a goal of the form $\mathtt{SP} \vdash \{(\forall X)\rho_i \mid i = \overline{1,m}\}$, where X is a set of constrained variables for Σ , where $\Sigma = \mathbb{S}ig(\mathtt{SP})$, and for all $i \in \{1,\ldots,m\}$, ρ_i is a $\Sigma[X]$-sentence [6]. Let F^c be the constructors of the signature Σ. We let $\mathtt{CON}$ range over all sort-preserving mappings $X \to F^c$, i.e. for every $x \in X$, the sort of $\mathtt{CON}(x)$ is less or equal than the sort of x. For each mapping $\mathtt{CON} : X \to F^c$ and variable $x \in X$ let $Z_{x,\mathtt{CON}} = z^1_{x,\mathtt{CON}} \ldots z^n_{x,\mathtt{CON}}$ be a string of arguments for the constructor $\mathtt{CON}(x)$. By an abuse of notation, we let $Z_{x,\mathtt{CON}}$ denote both the string $z^1_{x,\mathtt{CON}} \ldots z^n_{x,\mathtt{CON}}$ and the set $\{z^1_{x,\mathtt{CON}}, \ldots, z^n_{x,\mathtt{CON}}\}$. We define the set of variables $Z_{\mathtt{CON}} = \bigcup_{x \in X} Z_{x,\mathtt{CON}}$ and the substitution $\mathtt{VAR}_{\mathtt{CON}}{}^{\#} : X \to T_\Sigma(Z_{\mathtt{CON}})$ by $\mathtt{VAR}_{\mathtt{CON}}{}^{\#}(x) = \mathtt{CON}(x)(Z_{x,\mathtt{CON}})$. Let $\mathtt{VAR}_{\mathtt{CON}}$ range over all substitutions $X \to T_\Sigma(Z_{\mathtt{CON}})$ with the following properties:

(a) $\mathtt{VAR}_{\mathtt{CON}}(x) \in Z_{x,\mathtt{CON}}$ or $\mathtt{VAR}_{\mathtt{CON}}(x) = \mathtt{CON}(x)(Z_{x,\mathtt{CON}})$ for all $x \in X$, and
(b) $\mathtt{VAR}_{\mathtt{CON}}(x) \in Z_{x,\mathtt{CON}}$ for some $x \in X$.

Since any substitution is sort-decreasing, the sort of $\mathtt{VAR}_{\mathtt{CON}}(x)$ is less or equal than the sort of x. The function $\mathtt{CON}$ gives the induction cases, while the function $\mathtt{VAR}_{\mathtt{CON}}$ is used to define the induction hypothesis for each case. For all sort-preserving mappings $\mathtt{CON} : X \to F^c$ we define the following specification:

$$\mathtt{SP}_{\mathtt{CON}} = \mathtt{SP} * \iota_{Z_{\mathtt{CON}}} \cup (\Sigma[Z_{\mathtt{CON}}], \{\mathtt{VAR}_{\mathtt{CON}}(\rho_i) \mid \mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}}) \text{ and } i = \overline{1,m}\})$$

where $\iota_{Z_{\mathtt{CON}}} : \Sigma \hookrightarrow \Sigma[Z_{\mathtt{CON}}]$. *Simultaneous Induction* is defined as follows:

$$[SI]\frac{\mathtt{SP}_{\mathtt{CON}} \vdash \mathtt{VAR}_{\mathtt{CON}}{}^{\#}(\rho_i) \text{ for all } \mathtt{CON} : X \to F^c \text{ and } i = \overline{1,m}}{\mathtt{SP} \vdash \{(\forall X)\rho_i \mid i = \overline{1,m}\}}$$

Note that $\mathtt{SP}_{\mathtt{CON}}$ consists of the specification $\mathtt{SP}$ refined by the induction hypothesis $\{\mathtt{VAR}_{\mathtt{CON}}(\rho_i) \mid \mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}}) \text{ and } i = \overline{1,m}\}$.

Lemma 1. *[9] Consider a specification* $\mathtt{SP}$ *and a sentence* $(\forall X)\rho \in \mathbb{S}en(\mathbb{S}ig(\mathtt{SP}))$. *Let* $\Sigma = \mathbb{S}ig(\mathtt{SP})$ *and* Σ^c *be the sub-signature of constructors of* Σ. *We have*

(1) $\mathtt{SP} \models (\forall X)\rho$ *iff* $\mathtt{SP} * \iota_X \models \rho$, *where* $\iota_X : \Sigma \hookrightarrow \Sigma[X]$, *and*
(2) if X *is a set of variables of constrained sorts then* $\mathtt{SP} \models (\forall X)\rho$ *iff* $\mathtt{SP} \models (\forall Y)\theta(\rho)$ *for all substitutions* $\theta : X \to T_\Sigma(Y)$ *such that*
  (a) Y *is a finite set of variables of loose sorts, and*
  (b) $\theta(x) \in T_{\Sigma^c}(Y)$ *for all* $x \in X$.

The above lemma is used to prove the soundness of $[SI]$.

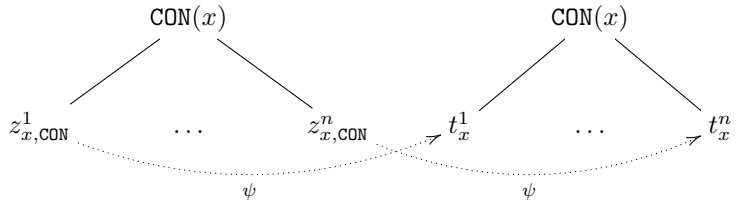
Proposition 5. Simultaneous Induction *is sound.*

Proof. Assume that $\mathtt{SP}_{\mathtt{CON}} \models \mathtt{VAR}_{\mathtt{CON}}{}^{\#}(\rho_i)$ for all $\mathtt{CON} : X \to F^c$ and $i = \overline{1,m}$. Let $\Sigma^c = (S, \leq, F^c)$ be the sub-signature of constructors, where $\Sigma = (S, \leq, F, F^c)$. By Lemma 1(2) it suffices to show that $\mathtt{SP} \models (\forall Y)\theta(\rho_i)$ for all $i = \overline{1,m}$ and $\theta : X \to T_\Sigma(Y)$ such that

[6] Note that $(\forall X)(\forall Y)\rho = (\forall X \cup Y)\rho$.

(1) Y is a finite set of variables of loose sorts, and
(2) $\theta(x) \in T_{\Sigma^c}(Y)$ for all $x \in X$.

Let $\theta : X \to T_\Sigma(Y)$ be such a substitution. We proceed by induction on the sum of depths of terms in $\{\theta(x) \mid x \in X\}$, which exists as a consequence of X being finite. Let $\mathtt{CON} : X \to F^c$ be the sort-preserving mapping such that for all $x \in X$ the topmost constructor of $\theta(x)$ is $\mathtt{CON}(x)$. For all $x \in X$ let $T_x = t_x^1 \dots t_x^n$ be the string of the intermediate subterms of $\theta(x)$. We define the substitution $\psi : Z_{\mathtt{CON}} \to T_{\Sigma^c}(Y)$ by $\psi(z^j_{x,\mathtt{CON}}) = t^j_x$.



By our assumptions we have

$$\mathtt{SP} * \iota_{Z_{\mathtt{CON}}} \cup (\Sigma[Z_{\mathtt{CON}}], \{\mathtt{VAR}_{\mathtt{CON}}(\rho_j) \mid \mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}}), j = \overline{1,m}\}) \models \mathtt{VAR}^{\#}_{\mathtt{CON}}(\rho_i)$$

for all $i = \overline{1,m}$. Since substitutions preserve satisfaction, we obtain

$$\mathtt{SP} * \iota_Y \cup (\Sigma[Y], \{\psi(\mathtt{VAR}_{\mathtt{CON}}(\rho_j)) | \mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}}), j = \overline{1,m}\}) \models \psi(\mathtt{VAR}^{\#}_{\mathtt{CON}}(\rho_i))$$

for all $i = \overline{1,m}$. Since $\mathtt{VAR}^{\#}_{\mathtt{CON}}; \psi = \theta$ we have

$$\mathtt{SP} * \iota_Y \cup (\Sigma[Y], \{\psi(\mathtt{VAR}_{\mathtt{CON}}(\rho_j)) \mid \mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}}), j = \overline{1,m}\}) \models \theta(\rho_i)$$

for all $i = \overline{1,m}$.

For all substitutions $\mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}})$, the sum of depths of terms in $\{\psi(\mathtt{VAR}_{\mathtt{CON}}(x)) \mid x \in X\}$ is strictly less than the sum of depths of terms in $\{\theta(x) \mid x \in X\}$. By the induction hypothesis, $\mathtt{SP} \models (\forall Y)\psi(\mathtt{VAR}_{\mathtt{CON}}(\rho_j))$ for all $\mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}})$ and $j = \overline{1,m}$. By Lemma 1(1), $\mathtt{SP} * \iota_Y \models \psi(\mathtt{VAR}_{\mathtt{CON}}(\rho_j))$ for all $\mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}})$ and $j = \overline{1,m}$. Since

$$\mathtt{SP} * \iota_Y \models \{\psi(\mathtt{VAR}_{\mathtt{CON}}(\rho_j)) \mid \mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}}), j = \overline{1,m}\}$$

and

$$\mathtt{SP} * \iota_Y \cup (\Sigma[Y], \{\psi(\mathtt{VAR}_{\mathtt{CON}}(\rho_j)) \mid \mathtt{VAR}_{\mathtt{CON}} : X \to T_\Sigma(Z_{\mathtt{CON}}), j = \overline{1,m}\}) \models \theta(\rho_i)$$

where $i = \overline{1,m}$. We obtain $\mathtt{SP} * \iota_Y \models \theta(\rho_i)$, where $i = \overline{1,m}$. By Lemma 1(1), $\mathtt{SP} \models (\forall Y)\theta(\rho_i)$, where $i = \overline{1,m}$. □

Case Analysis [CA]. This proof rule divides a goal into a sufficient number of separate cases. Analysing each such case individually may be enough to prove the initial goal. We say that $E = \{(\forall X)(\forall Y^i)u = v^i \text{ if } \mathtt{Cond}^i \mid i = \overline{1,n}\} \subseteq \mathbb{A}x(SP)$ is a $[CA]$-*set of sentences*, where $\mathtt{SP}$ is a specification, if

- X is the set of all variables occurring in u, and
- $\mathtt{SP} \models \bigvee_{i=1}^{i\leq n} (\exists Y^i)\psi(\mathtt{Cond}^i)$ for all substitutions $\psi : X \to T_\Sigma$.

Consider a goal $\mathtt{SP} \vdash \rho$, and a $[CA]$-set of sentences $E \subseteq \mathbb{A}x(\mathtt{SP})$ as above. Let t be a ground term occurring in ρ, and t_1 a subterm of t which is matched by u, i.e. there exists a substitution $\theta : X \to T_{\mathbb{S}ig(\mathtt{SP})}$ such that $\theta(u) = t_1$. We define *Case Analysis* as follows:

$$[CA]\frac{\mathtt{SP} * \iota_{Y^i} \cup (\Sigma[Y^i], \theta(\mathtt{Cond}^i)) \vdash \rho \text{ for all } i = \overline{1,n}}{\mathtt{SP} \vdash \rho}$$

where $\iota_{Y^j} : \Sigma \hookrightarrow \Sigma[Y^j]$ is the signature inclusion.

In our case study, X consists of a single variable of the sort representing the state space of the transitional system, and Y^j consists of variables added by the matching equations. The specification $\mathtt{SP}$ obviously satisfies the disjunction $\bigvee_{i=1}^{i\leq n} (\exists Y^i)\psi(\mathtt{Cond}^i)$ as the conditions $\mathtt{Cond}^i$ describe all possible patterns of a sequence.

Proposition 6. Case Analysis *is sound.*

Proof. Assume that $\mathtt{SP} * \iota_{Y^i} \cup (\Sigma[Y^i], \theta(\mathtt{Cond}^i)) \models \rho$ for all $i \in \{1, \ldots, n\}$ and let $M \in \mathbb{M}od(\mathtt{SP})$. We prove that $M \models_\Sigma \rho$. By our assumptions, there exists $j \in \{1, \ldots, n\}$ and a ι_{Y^j}-expansion M' of M, where $\iota_{Y^j} : \Sigma \hookrightarrow \Sigma[Y^j]$, such that $M' \models_{\Sigma[Y^j]} \theta(\mathtt{Cond}^j)$. Notice that $M' \in \mathbb{M}od(\mathtt{SP} * \iota_{Y^j} \cup (\Sigma[Y^j], \theta(\mathtt{Cond}^j)))$, and since $\mathtt{SP} * \iota_{Y^j} \cup (\Sigma[Y^j], \theta(\mathtt{Cond}^j)) \models \rho$, we have $M' \models_{\Sigma[Y^j]} \rho$. Hence $M \models_\Sigma \rho$. □

Substitutivity [ST]. We can infer new sentences by substituting terms for variables. This proof rule is used during the verification process to instantiate sentences that are not executable by rewriting.

$$[ST]\frac{(\forall Y)\rho \in \mathbb{A}x(\mathtt{SP}) \quad \theta : Y \to T_{\mathbb{S}ig(\mathtt{SP})}(Z)}{\mathtt{SP} \vdash (\forall Z)\theta(\rho)}$$

Proposition 7. Substitutivity *is sound.*

Proof. The proof is a direct consequence of the satisfaction condition for substitutions (see Proposition 4). □

Subterm Replacement [SR]. A specialized rule of inference using subterm replacement is the basis for *term rewriting*. Let $\mathtt{SP}$ be a specification and $(\forall X)\rho$ a Σ-sentence, where $\Sigma = \mathbb{S}ig(\mathtt{SP})$. Suppose that $X = Y \cup \{z\}$, where $z \notin Y$, and $\theta, \psi : X \to T_\Sigma(Y)$ are two substitutions such that $\theta(y) = \psi(y) = y$ for all $y \in Y$.

$$\frac{\mathtt{SP} \vdash (\forall Y)\psi(\rho) \quad \mathtt{SP} \vdash (\forall Y)\theta(z) = \psi(z)}{\mathtt{SP} \vdash (\forall Y)\theta(\rho)}$$

The following result is a generalisation of soundness of *Subterm Replacement* for many-sorted algebra [12].

Proposition 8. Subterm Replacement *is sound.*

Theorem of Constants [TC]. We define the following proof rule for the quantification over variables of loose sorts:

$$[TC]\frac{\mathtt{SP} * \iota_Y \vdash \rho}{\mathtt{SP} \vdash (\forall Y)\rho}$$

where $\mathtt{SP}$ is a specification, $(\forall Y)\rho$ is a sentence, and $\iota_Y : \mathbb{S}ig(\mathtt{SP}) \hookrightarrow \mathbb{S}ig(\mathtt{SP})[Y]$. Note that even if not all of the variables in Y are of loose sorts if we have proved $\mathtt{SP} * \iota_Y \vdash \rho$ then we conclude $\mathtt{SP} \vdash (\forall Y)\rho$. This means that there are cases in practice when we apply $[TC]$ to goals with sentences quantified over variables of constrained sorts. The following proposition is a corollary of Lemma 1(1).

Proposition 9. Theorem of Constants *is sound.*

Remark 1. In the verification process we separate variables that are handled with $[SI]$ from the variables that are dealt with $[TC]$, rather than distinguishing variables of constrained sorts from variables of loose sorts. In general, this choice cannot be automated and is entirely up to the user to make.

Implication. The following proof rule is defined for the logical implication:

$$[IP]\frac{\mathtt{SP} \cup (\mathbb{S}ig(\mathtt{SP}), \mathtt{Cond}) \vdash \mathtt{atm}}{\mathtt{SP} \vdash \mathtt{atm} \text{ if } \mathtt{Cond}}$$

where $\mathtt{SP}$ is a specification, $\mathtt{atm}$ is an atom, and $\mathtt{Cond}$ is a conjunction of atoms.

Proposition 10. *[9]* Implication *is sound.*

The following theorem is a corollary of Propositions 5-10.

Theorem 1. *Consider a sound entailment relation* $\vdash^b = \{\mathtt{SP} \vdash^b \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ *for proving quantifier-free unconditional atoms. The least entailment relation* $\vdash = \{\mathtt{SP} \vdash \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ *over* $\vdash^b$ *closed to* $[SI]$, $[CA]$, $[ST]$, $[SR]$, $[TC]$ *and* $[IP]$ *is sound.*

Proof. By Proposition 5-10, the semantic entailment relation $\{\mathtt{SP} \models \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ is closed to $[SI]$, $[CA]$, $[ST]$, $[SR]$, $[TC]$ and $[IP]$. Then the least entailment relation $\{\mathtt{SP} \vdash \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ closed to $[SI]$, $[CA]$, $[ST]$, $[SR]$, $[TC]$ and $[IP]$ is included in $\{\mathtt{SP} \models \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$. □

2.3 Specialised Proof Rules

When initial semantics is used the above rules are not enough to prove the desired properties of the specifications. Specialised proof rules (that cannot be derived from the general inference rules defined in Section 2.2) are needed to complete the verification process. Since algebraic specification languages have libraries defining the initial data types often used in applications, it is natural for the theorem prover supporting the verification to be equipped with deduction rules for the predefined data types declared with initial semantics.

Contradiction [CT]. In algebraic specification the boolean data type is often used to establish the truth. The boolean specification `BOOL` is defined with initial semantics and it is imported by any other specification in protecting mode. The following proof rule is valid for any specification that protects the boolean values. For any specification morphisms $\iota : \texttt{BOOL} \hookrightarrow \texttt{SP}$ such that $\iota : \mathbb{S}ig(\texttt{BOOL}) \hookrightarrow \mathbb{S}ig(\texttt{SP})$ is an inclusion, we define the following proof rule:

$$[CT]\frac{\texttt{SP} \vdash \texttt{true => false}}{\texttt{SP} \vdash \rho}$$

Proposition 11. Contradiction *is sound.*

Proof. For any specification morphisms $\iota : \texttt{BOOL} \hookrightarrow \texttt{SP}$ and $M \in |\mathbb{M}od(\texttt{SP})|$ we have $M \not\models \texttt{true => false}$. If $\texttt{SP} \models \texttt{true => false}$ then SP has no models, which implies $\texttt{SP} \models \rho$ for any sentence $\rho \in \mathbb{S}en(\mathbb{S}ig(\texttt{SP}))$. □

It follows that negation has been somehow introduced, in a weaker form. For example, the following sentence $(\forall \texttt{Y})\texttt{true} = \texttt{false}$ if $\texttt{t}_1 = \texttt{t}_2$ says that the term $\texttt{t}_1$ is different from $\texttt{t}_2$.

Less-Equal [LE]. Let `NAT` denote the specification defined with initial semantics which includes a sort `Nat` with two constructors $\texttt{0} :\rightarrow \texttt{Nat}$ and $\texttt{s\_} : \texttt{Nat} \rightarrow \texttt{Nat}$, and an ordinary operators `<= : Nat Nat -> Bool` defined by the following sets of equations:

$$\begin{array}{ll}
(\forall \texttt{M}) & \texttt{M <= M} = \texttt{true}, \\
(\forall \texttt{M}) & \texttt{0 <= s M} = \texttt{true}, \\
(\forall \texttt{M}) & \texttt{s M <= 0} = \texttt{false}, \\
(\forall \texttt{M}, \texttt{N}) & \texttt{s M <= s N} = \texttt{M <= N}.
\end{array}$$

For any specification morphism $\texttt{NAT} \hookrightarrow \texttt{SP}$ such that we have $\{(\texttt{s}^m\texttt{0 <= t} = \texttt{true}), (\texttt{t <= s}^n\texttt{0} = \texttt{true})\} \subseteq \mathbb{A}\texttt{x}(\texttt{SP})$ and $n < m$, where $t \in T_{\mathbb{S}ig(\texttt{SP})}$ and $n, m \in \mathbb{N}$, we define the following proof rule

$$[LE]\frac{}{\texttt{SP} \vdash \rho}$$

where ρ is any sentence.[7]

[7] Note that for all natural numbers $n, m \in \mathbb{N}$ we have $n \leq m$ iff $\texttt{s}^n\texttt{0 <= s}^m\texttt{0}$, where $\texttt{s}^n\texttt{0} = \underbrace{s \ldots s}_{\text{n times}}\ \texttt{0}$ and $\texttt{s}^n\texttt{0 <= s}^m\texttt{0}$ stands for $\texttt{s}^n\texttt{0 <= s}^m\texttt{0} = \texttt{true}$.

Proposition 12. Less-Equal *rule is sound.*

Proof. For any specification morphism $\texttt{NAT} \hookrightarrow \texttt{SP}$ such that $\{\texttt{s}^m\texttt{0} <= \texttt{t}, \texttt{t} <= \texttt{s}^n\texttt{0}\} \subseteq \mathbb{A}x(\texttt{SP})$, we have $\texttt{SP} \models \texttt{s}^m\texttt{0} <= \texttt{s}^n\texttt{0}$. For any $M \in \mathbb{M}od(\texttt{SP})$ if $n < m$ then $M \models \texttt{s}^n\texttt{0} <= \texttt{s}^m\texttt{0}$ and $M \not\models \texttt{s}^n\texttt{0} = \texttt{s}^m\texttt{0}$, which is a contradiction with $\texttt{SP} \models \texttt{s}^m\texttt{0} <= \texttt{s}^n\texttt{0}$. It follows that SP has no models. Hence, for any sentence ρ we have $\texttt{SP} \models \rho$. □

Sequence Case Analisys [SC]. Our approach to support the automation of case analysis is to consider specialized proof rules for various patterns over the data types. Here we consider the example of sequences. Let SEQUENCE denote the specification defined with initial semantics which includes a sort Sequence together with the following constructors: empty denoting the empty sequence, and an associative operation _,_ denoting the concatenation. The elements of the sequences are of sort Elt. Let $\iota : \texttt{SEQUENCE} \hookrightarrow \texttt{SP}$ be a specification morphism, where $\iota : \mathbb{S}ig(\texttt{SEQUENCE}) \hookrightarrow \mathbb{S}ig(\texttt{SP})$ is an inclusion of signatures. Suppose we want

$$\texttt{SP} \vdash (\forall \texttt{Y})\texttt{atm} \text{ if } \texttt{Cond} \wedge \texttt{L1}, \texttt{E1}, \texttt{L2}, \texttt{E2}, \texttt{L3} = \texttt{t1}, \texttt{t2} \wedge \texttt{Cond}'$$

where Li are variables of sort Sequence, Ei are variables of sort Elt, $\texttt{Y} \supseteq \{\texttt{L1}, \texttt{L2}, \texttt{L3}, \texttt{E1}, \texttt{E2}\}$ is a set of variables, atm is an atom, Cond and Cond′ are conjunctions of atoms, and $\texttt{t}_1, \texttt{t}_2 \in T_{\mathbb{S}ig(\texttt{SP})}(\texttt{Y} - \{\texttt{L1}, \texttt{E1}, \texttt{L2}, \texttt{E2}, \texttt{L3}\})_{\texttt{Sequence}}$ are terms that do not contain the variables L1, E1, L2, E2, L3. $[SC]$ divides the above goal into the following cases:

(1) $\texttt{SP} \vdash (\forall \texttt{Y})\texttt{atm}[\texttt{L3} \leftarrow \texttt{L3}, \texttt{t2}]$ if $\texttt{Cond}[\texttt{L3} \leftarrow \texttt{L3}, \texttt{t2}] \wedge \texttt{L1}, \texttt{E1}, \texttt{L2}, \texttt{E2}, \texttt{L3} = \texttt{t1} \wedge \texttt{Cond}'[\texttt{L3} \leftarrow \texttt{L3}, \texttt{t2}]$,
(2) $\texttt{SP} \vdash (\forall \texttt{Y}')\texttt{atm}[\texttt{L2} \leftarrow \texttt{L2}, \texttt{L2}']$ if $\texttt{Cond}[\texttt{L2} \leftarrow \texttt{L2}, \texttt{L2}'] \wedge \texttt{L1}, \texttt{E1}, \texttt{L2} = \texttt{t1} \wedge \texttt{L2}', \texttt{E2}, \texttt{L3} = \texttt{t2} \wedge \texttt{Cond}'[\texttt{L2} \leftarrow \texttt{L2}, \texttt{L2}']$, where $\texttt{Y}' = \texttt{Y} \cup \{\texttt{L2}'\}$,
(3) $\texttt{SP} \vdash (\forall \texttt{Y})\texttt{atm}[\texttt{L1} \leftarrow \texttt{t1}, \texttt{L1}]$ if $\texttt{Cond}[\texttt{L1} \leftarrow \texttt{t1}, \texttt{L1}] \wedge \texttt{L1}, \texttt{E1}, \texttt{L2}, \texttt{E2}, \texttt{L3} = \texttt{t2} \wedge \texttt{Cond}'[\texttt{L1} \leftarrow \texttt{t1}, \texttt{L1}]$

The above three subgoals describe the following three possibilities: either both E1 and E2 are in t1, or E1 is in t1 and E2 is in t2, or E1 and E2 are in t2. The rule $[SC]$ is specific to SEQUENCE, which is declared with initial semantics, and it cannot be derived from the general proof rules.

Proposition 13. Sequence Case Analysis *is sound.*

Proof. Assume that

(1) $\texttt{SP} \models (\forall \texttt{Y})\texttt{atm}[\texttt{L3} \leftarrow \texttt{L3}, \texttt{t2}]$ if $\texttt{Cond}[\texttt{L3} \leftarrow \texttt{L3}, \texttt{t2}] \wedge \texttt{L1}, \texttt{E1}, \texttt{L2}, \texttt{E2}, \texttt{L3} = \texttt{t1} \wedge \texttt{Cond}'[\texttt{L3} \leftarrow \texttt{L3}, \texttt{t2}]$,
(2) $\texttt{SP} \models (\forall \texttt{Y}')\texttt{atm}[\texttt{L2} \leftarrow \texttt{L2}, \texttt{L2}']$ if $\texttt{Cond}[\texttt{L2} \leftarrow \texttt{L2}, \texttt{L2}'] \wedge \texttt{L1}, \texttt{E1}, \texttt{L2} = \texttt{t1} \wedge \texttt{L2}', \texttt{E2}, \texttt{L3} = \texttt{t2} \wedge \texttt{Cond}'[\texttt{L2} \leftarrow \texttt{L2}, \texttt{L2}']$, where $\texttt{Y}' = \texttt{Y} \cup \{\texttt{L2}'\}$,
(3) $\texttt{SP} \models (\forall \texttt{Y})\texttt{atm}[\texttt{L1} \leftarrow \texttt{t1}, \texttt{L1}]$ if $\texttt{Cond}[\texttt{L1} \leftarrow \texttt{t1}, \texttt{L1}] \wedge \texttt{L1}, \texttt{E1}, \texttt{L2}, \texttt{E2}, \texttt{L3} = \texttt{t2} \wedge \texttt{Cond}'[\texttt{L1} \leftarrow \texttt{t1}, \texttt{L1}]$.

We show $\mathtt{SP} \models (\forall \mathtt{Y})\mathtt{t} = \mathtt{t}'$ if $\mathtt{Cond} \wedge \mathtt{L1}, \mathtt{E1}, \mathtt{L2}, \mathtt{E2}, \mathtt{L3} = \mathtt{t1}, \mathtt{t2} \wedge \mathtt{Cond}'$. Let $M \in \mathbb{M}od(\mathtt{SP})$.

We denote by Σ the signature $\mathbb{S}ig(\mathtt{SP})$. Let N be a ι_y-expansion of M, where $\iota_y : \Sigma \hookrightarrow \Sigma[Y]$, such that $N \models_{\Sigma[Y]} \mathtt{Cond}$, $N \models_{\Sigma[Y]} \mathtt{L1}, \mathtt{E1}, \mathtt{L2}, \mathtt{E2}, \mathtt{L3} = \mathtt{t1}, \mathtt{t2}$ and $N \models_{\Sigma[Y]} \mathtt{Cond}'$. Since the sequences are protected, there are three possibilities:

(a) $N_{\mathtt{E1}}$ and $N_{\mathtt{E2}}$ are in $N_{\mathtt{t1}}$,
(b) $N_{\mathtt{E1}}$ is in $N_{\mathtt{t1}}$, and $N_{\mathtt{E2}}$ is in $N_{\mathtt{t2}}$, and
(c) $N_{\mathtt{E1}}$ and $N_{\mathtt{E2}}$ are in $N_{\mathtt{t2}}$.

We will focus on the first case as the rest of the cases are similar. Without danger of confusion the interpretation of $\_, \_$ into the model N will be denoted also by $\_, \_$. Since $N_{\mathtt{L1}}, N_{\mathtt{E1}}, N_{\mathtt{L2}}, N_{\mathtt{E2}}, N_{\mathtt{L3}} = N_{\mathtt{t1}}, N_{\mathtt{t2}}$ and both $N_{\mathtt{E1}}$ and $N_{\mathtt{E2}}$ are in $N_{\mathtt{t1}}$, there exists $n \in N_{\mathtt{Sequence}}$ such that $N_{\mathtt{L3}} = n, N_{\mathtt{t2}}$. Let $\theta : \mathtt{Y} \to T_\Sigma(\mathtt{Y})$ be the substitution which is the identity on $\mathtt{Y} - \{\mathtt{L3}\}$ and $\theta(\mathtt{L3}) = \mathtt{L3}, \mathtt{t2}$. Let N' be a $\iota_{\mathtt{Y}}$-expansion of M such that $N'_{\mathtt{L3}} = n$ and $N'_y = N_y$ for all $y \in \mathtt{Y} - \{\mathtt{L3}\}$. Since $\mathtt{t}_2$ does not contain $\mathtt{L3}$ we have $N'_{\mathtt{t2}} = N_{\mathtt{t2}}$. Note that $(N'|_\theta)_{\mathtt{L3}} = N'_{\theta(\mathtt{L3})} = N'_{\mathtt{L3},\mathtt{t2}} = n, N'_{\mathtt{t2}} = n, N_{\mathtt{t2}} = N_{\mathtt{L3}}$. We obtain $N'|_\theta = N$, and since $N \models_{\Sigma[\mathtt{Y}]} \mathtt{Cond}$ and $N \models_{\Sigma[\mathtt{Y}]} \mathtt{Cond}'$, by the satisfaction condition for substitutions, $N' \models_{\Sigma[\mathtt{Y}]} \mathtt{Cond}[\mathtt{L3} \leftarrow \mathtt{L3}, \mathtt{t2}]$ and $N' \models_{\Sigma[\mathtt{Y}]} \mathtt{Cond}'[\mathtt{L3} \leftarrow \mathtt{L3}, \mathtt{t2}]$. We have

$$N'_{\mathtt{L1}}, N'_{\mathtt{E1}}, N'_{\mathtt{L2}}, N'_{\mathtt{E2}}, N'_{\mathtt{L3}}, N'_{\mathtt{t2}} = N_{\mathtt{L1}}, N_{\mathtt{E1}}, N_{\mathtt{L2}}, N_{\mathtt{E2}}, n, N_{\mathtt{t2}} =$$
$$N_{\mathtt{L1}}, N_{\mathtt{E1}}, N_{\mathtt{L2}}, N_{\mathtt{E2}}, N_{\mathtt{L3}} = N_{\mathtt{t1}}, N_{\mathtt{t2}} = N'_{\mathtt{t1}}, N'_{\mathtt{t2}}$$

Since the sequences are protected, $N'_{\mathtt{L1}}, N'_{\mathtt{E1}}, N'_{\mathtt{L2}}, N'_{\mathtt{E2}}, N'_{\mathtt{L3}} = N'_{\mathtt{t1}}$. We obtain $N' \models_{\Sigma[\mathtt{Y}]} \mathtt{L1}, \mathtt{E1}, \mathtt{L2}, \mathtt{E2}, \mathtt{L3} = \mathtt{t1}$. By assumption (1), $N' \models_{\Sigma[\mathtt{Y}]} \mathtt{atm}[\mathtt{L3} \leftarrow \mathtt{L3}, \mathtt{t2}]$, and by the satisfaction condition for substitutions, $N \models_{\Sigma[\mathtt{Y}]} \mathtt{atm}$. □

$[SC]$ simplifies the formula to prove while $[CA]$ refines the specification of the goal by adding new equations. The following theorem is a corollary of Theorem 1 and Propositions 11, 12 and 13.

Theorem 2. *Consider a sound entailment relation $\vdash^b = \{\mathtt{SP} \vdash^b \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ for proving quantifier-free unconditional atoms. Then the least entailment relation $\vdash = \{\mathtt{SP} \vdash \_\}_{\mathtt{SP} \in |\mathtt{SPEC}|}$ over $\vdash^b$ closed to $[SI]$, $[CA]$, $[ST]$, $[SR]$, $[TC]$, $[IP]$, $[CT]$, $[LE]$ and $[SC]$ is sound.*

2.4 Tactics

This methodology is designed for algebraic specification languages which are executable by rewriting. Given a goal $\mathtt{SP} \vdash E$, the underlying assumption is that $\mathbb{A}x(\mathtt{SP})$ forms a term rewriting system which is terminating and possibly confluent. By applying a tactic to a goal it is desirable to preserve termination and confluence. Note that the proof rules of the specification calculus can be regarded, upside down, as tactics for decomposing problems.

Reduction $[RD]$. The basic entailment relation $\vdash^b$ from Theorem 2 is provided by the system which supports the description of constructor-based specifications. The present methodology is implemented in CITP [11] which is built on top of Maude. Given a specification $\mathtt{SP}$, any goal of the form (1) $\mathtt{SP} \vdash t = t'$, (2) $\mathtt{SP} \vdash t : s$ and (3) $\mathtt{SP} \vdash t \Rightarrow t'$, where t, t' are ground terms and s is a sort, is reduced to the empty goal set if (1) t and t' can be rewritten to the same normal form by the system using the equations and membership axioms of $\mathbb{A}x(\mathtt{SP})$, (2) the sort of t is a subsort of s, (3) t can be rewritten to t' by applying the preorder axioms of $\mathbb{A}x(\mathtt{SP})$, respectively. Maude search engine is invoked in the third case.

Inconsistency $[IC]$. Let $\mathtt{BOOL} \hookrightarrow \mathtt{SP}$ be a specification morphism. One can easily prove by induction that $\mathtt{SP} \vdash (\forall B)t = t'$ if $\mathtt{Cond} \wedge B = \mathtt{not}\ B \wedge \mathtt{Cond}'$, where B is a variable of sort $\mathtt{Bool}$. By $[ST]$, $\mathtt{SP} \vdash t = t'$ if $\mathtt{Cond} \wedge b = \mathtt{not}\ b \wedge \mathtt{Cond}'$ for all ground terms b of sort $\mathtt{Bool}$. It follows that the entailment system defined above is closed to the following rule of inference

$$[IC]\frac{}{\mathtt{SP} \vdash t = t' \text{ if } \mathtt{Cond} \wedge b = \mathtt{not}\ b \wedge \mathtt{Cond}'}$$

where $\mathtt{BOOL} \hookrightarrow \mathtt{SP}$ is a specification morphism and $(t = t' \text{ if } \mathtt{Cond} \wedge b = \mathtt{not}\ b \wedge \mathtt{Cond}')$ is a quantifier-free sentence such that b is a ground term of sort $\mathtt{Bool}$. $[IC]$ is crucial for the automation since the equation $b = \mathtt{not}\ b$ might cause a non-terminating process when added to $\mathtt{SP}$ by the $[IP]$ rule.

Normal Forms $[NF]$. Algebraic specification languages executable by rewriting are equipped with a partial function $nf_{\mathtt{SP}} : T_{\mathbb{S}ig(\mathtt{SP})} \to T_{\mathbb{S}ig(\mathtt{SP})}$ for all specifications $\mathtt{SP}$ such that for all $t \in T_{\mathbb{S}ig(\mathtt{SP})}$, $nf_{\mathtt{SP}}(t)$ is a normal form of t w.r.t. $\mathbb{A}x(\mathtt{SP})$ if one exists, and $nf_{\mathtt{SP}}(t)$ is undefined, otherwise. Note that $\mathtt{SP} \vdash^b t = t'$ iff $nf_{\mathtt{SP}}(t)$ and $nf_{\mathtt{SP}}(t')$ are defined and equal.

Let $nf_{\mathtt{SP}} : \mathbb{S}en(\mathtt{SP}) \to \mathbb{S}en(\mathtt{SP})$ be the canonical extension of $nf : T_{\mathbb{S}ig(\mathtt{SP})} \to T_{\mathbb{S}ig(\mathtt{SP})}$. The following rule of inference can be obtained by a successive application of $[SR]$:

$$[NF]\frac{\mathtt{SP} \vdash nf_{\mathtt{SP}}(\rho)}{\mathtt{SP} \vdash \rho}$$

where ρ is any sentence.

Proposition 14. *The entailment system defined in Theorem 1 is closed to* $[NF]$.

Proof. Let t_1 be a ground term occurring in ρ. Since $\mathtt{SP} \vdash^b t_1 = nf(t_1)$, by $[SR]$ the entailment system of Theorem 1 is closed to the following rule of inference: $\dfrac{\mathtt{SP} \vdash \rho[t_1 \leftarrow nf(t_1)]}{\mathtt{SP} \vdash \rho}$. Let $\{t_1, \ldots, t_n\}$ be all ground terms occurring in ρ. Note that $nf(\rho) = \rho[t_1 \leftarrow nf(t_1), \ldots, t_n \leftarrow nf(t_n)]$. By applying n times the rule $[SR]$ we get that the entailment system defined in Theorem 1 is closed to $[NF]$. □

In order to preserve the confluence property of specifications in the proof process, each application of any of the proof rules above is preceded by a reduction of the ground terms occurring in the formulas to prove to their normal

forms. In practice, the proof of a goal $\mathtt{SP} \vdash \rho$ stops if $nf_{\mathtt{SP}}(\rho)$ is undefined. The following simple example illustrates the benefit of this tactic.

Example 1. Consider a specification SP with three constant symbols `a b c` $: \to$ `s` and one equation `a = b`. If we apply $[IP]$ to $\mathtt{SP} \vdash \mathtt{b} = \mathtt{c}$ if $\mathtt{a} = \mathtt{c}$ we get $\mathtt{SP} \cup (\mathbb{S}ig(\mathtt{SP}), \{\mathtt{a} = \mathtt{c}\}) \vdash \mathtt{b} = \mathtt{c}$, and $\mathtt{SP} \cup (\mathbb{S}ig(\mathtt{SP}), \{\mathtt{a} = \mathtt{c}\})$ is not confluent because the critical pair $\mathtt{c} \leftarrow \mathtt{a} \rightarrow \mathtt{b}$ is not joinable. On the other hand, if a reduction to the normal forms is performed then the new goal is $\mathtt{SP} \vdash \mathtt{b} = \mathtt{c}$ if $\mathtt{b} = \mathtt{c}$, which is reduced to $\mathtt{SP} \cup (\mathbb{S}ig(\mathtt{SP}), \{\mathtt{b} = \mathtt{c}\}) \vdash \mathtt{b} = \mathtt{c}$ by $[IP]$. By applying $[NF]$ we get $\mathtt{SP} \cup (\mathbb{S}ig(\mathtt{SP}), \{\mathtt{b} = \mathtt{c}\}) \vdash \mathtt{c} = \mathtt{c}$; finally $\mathtt{SP} \cup (\mathbb{S}ig(\mathtt{SP}), \{\mathtt{b} = \mathtt{c}\}) \vdash \mathtt{c} = \mathtt{c}$ is discharged by $[RD]$.

Since $[NF]$ is coupled with any other proof rule, it will be omitted from the discussions.

Case Analysis (revisited). In practice, we give labels to conditional equations used for case analysis. It follows that the $[CA]$-sets of sentences are part of the specification. Given a specification SP let $CA(\mathtt{SP}) = (\{(\forall X_j)(\forall Y_j^i)u_j = v_j^i \text{ if } \mathtt{Cond}^i \mid i = \overline{1, n_j}\})_{j \in J} \subseteq \mathcal{P}(\mathbb{A}x(\mathtt{SP}))$ be the family of all $[CA]$-sets of sentences of SP. Consider a ground term $t \in T_{\mathbb{S}ig(\mathtt{SP})}$. We say that t_1 is a $[CA]$-*subterm* of t if there exists $j_{t_1} \in J$ such that

(1) $u_{j_{t_1}}$ matches t_1, i.e. there exists a substitution $\theta_{t_1} : X \to T_{\mathbb{S}ig(\mathtt{SP})}$ such that $\theta(u_{j_{t_1}}) = t_1$, and
(2) there is no $j \in J$ such that u_j matches a proper subterm of t_1.

Assume a goal $\mathtt{SP} \vdash \rho$ and a ground term t occurring in ρ. Consider the tactic which consists of successive applications of $[CA]$, one for each $[CA]$-subterm of t. This tactic will replace $[CA]$ defined above. In applications, t is selected automatically from the list of all ground terms which occur in ρ.

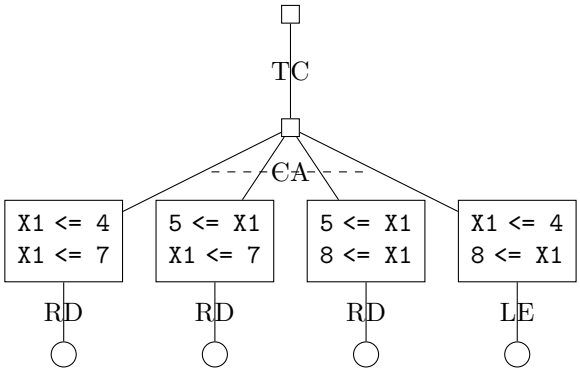
Example 2. Consider the specification morphism $\mathtt{NAT} \hookrightarrow \mathtt{FUN}$, where FUN is a specification with two functions over the natural numbers:

(1) F is defined by the following $[CA]$-set: $\begin{cases} \mathtt{F(X)} = \mathtt{5} \text{ if } \mathtt{X <= 7}, \\ \mathtt{F(X)} = \mathtt{1} \text{ if } \mathtt{8 <= X}. \end{cases}$
(2) G is defined by the following $[CA]$-set: $\begin{cases} \mathtt{G(Y)} = \mathtt{2} \text{ if } \mathtt{Y <= 4}, \\ \mathtt{G(Y)} = \mathtt{7} \text{ if } \mathtt{5 <= Y}. \end{cases}$

Suppose we want to prove $\mathtt{FUN} \vdash (\forall \mathtt{X})\mathtt{9 <= G(F(X)) + G(X) = true}$. By $[TC]$ the new goal is $\mathtt{FUN} * \iota_{\mathtt{X1}} \vdash \mathtt{9 <= G(F(X1)) + G(X1) = true}$, where $\iota_{\mathtt{X1}} : \mathbb{S}ig(\mathtt{FUN}) \hookrightarrow \mathbb{S}ig(\mathtt{FUN})[\mathtt{X1}]$ is the inclusion of signatures. Case analysis is performed with respect to the $[CA]$-subterms `F(X1)` and `G(X1)` of the term `9 <= G(F(X1)) + G(X1)`. There are four cases:

$$(a)\ \mathtt{X1 <= 4}, \qquad (c)\ \mathtt{8 <= X1 <= 4},$$
$$(b)\ \mathtt{5 <= X1 <= 7}, \quad (d)\ \mathtt{8 <= X1}.$$

Note that the cases (a), (b) and (d) are discharged by $[RD]$ while the case (c) is discharged by $[LE]$. The corresponding proof tree is depicted in the figure below, where the circles represent the empty goal set.



Remark 2. The proof of `FUN` $\vdash$ (∀`X`)`9 <= G(F(X))+G(X) = true` can be performed automatically by CITP by giving the command `(apply TC CA RD .)`.

Note that $[LE]$ is applied automatically by CITP without giving an explicit command. The idea is that the interaction with the user is not needed to discharge goals with inconsistent specifications. Hence, $[IC]$ and $[CT]$ are applied also automatically by CITP.

Proof Strategy. Except $[SI]$, all basic tactics are designed for goals consisting of a specification and a single formula. We make the following convention: if a tactic different from $[SI]$ is applied to a goal of the form `SP` $\vdash \{\rho_1, \ldots, \rho_n\}$, the goal is decomposed into a set of subgoals $\{$`SP` $\vdash \rho_1, \ldots,$ `SP` $\vdash \rho_n\}$, and then the tactic is applied to each `SP` $\vdash \rho_i$.

The application order of the basic tactics is crucial for automating the proof process. $[SI]$ is applied first. $[TC]$ is performed before $[IC]$ as $[IC]$ can be applied only to goals with quantifier-free formulas. $[IC]$ discharges any goal of the form `SP` $\vdash$ `atm` if `Cond` $\wedge$ `b` $=$ `not b` $\wedge$ `Cond'`, otherwise $[IC]$ leaves the goal unchanged. An application of $[IP]$ is preceded by an application of $[IC]$. $[RD]$ attempts to discharge any goal with an atomic formula. If $[RD]$ fails to complete the proof then $[CT]$ and $[LE]$ tactics are used.

The order of tactic applications described above is well-established for all goals. However, the application of case analysis depends on the problem to solve. Based on the case studies, we have two proof strategies:

(1) $[SI]$, $[CA]$, $[SC]$, $[TC]$, $[IC]$, $[IP]$, $[RD]$, $[CT]$, $[LE]$, and
(2) $[SI]$, $[TC]$, $[CA]$, $[IC]$, $[IP]$, $[RD]$, $[CT]$, $[LE]$.

A tactic can be applied to a goal if it has a certain pattern, otherwise the tactic leaves the goal unchanged. Note that not all OTS are modelled using sequences. In such a case the application of $[SC]$ leaves the goal unchanged. The advantages of these proof strategies can be noticed clearly in concrete examples which can be found at `http://www.jaist.ac.jp/~danielmg/citp.html`.

The present methodology does not include automatic lemma discovery. It is for the user to find the appropriate induction scheme for the problem to solve, and the constrained variables for induction.

3 Methodology at Work: ABP Case Study

Alternating Bit Protocol is a communication protocol that enables to send reliable messages on unreliable channels. It is often used as a case study, either for some algebraic formalisms or for tools dedicated to analysis or verification of concurrent systems. Even if the protocol seems to be simple, its complete algebraic specification is quite complex and its formal correctness proof is very large. We show that using our methodology, most of the proving process can be automated.

Since this section involves the operational semantics of the system that assists the proofs, we write the formulas in Maude-like notation, meaning that we omit the universal quantifier and we distinguish ordinary equations from matching equations.

3.1 The ABP Protocol

We describe briefly the protocol. Two processes, *Sender* and *Receiver*, that do not share any common memory use two channels to communicate with each other. Sender sends repeatedly a pair `< bit1,pac >` of a bit and a packet to the receiver over one of the channels, let's say `channel1`. When Sender gets `bit1` from Receiver over the other channel, let's say `channel2`, it is a confirmation from Receiver that the packet sent was received. In this case Sender alternates `bit1` and selects the next packet for sending. Receiver puts `bit2` into `channel2` repeatedly. When Receiver gets a pair `< b,p >` such that `b` is different from `bit2` it stores `p` into a `list` and alternates `bit2`. Initially both channels are empty and the Sender's bit is different from the Receiver's bit. We assume that the channels are unreliable, meaning that the data in the channels may be lost, but not exchanged or damaged.

The packets sent by Sender to Receiver through `channel1` are indexed by the natural numbers and are of the form $\texttt{pac}(\texttt{0}), \texttt{pac}(\texttt{s0}), \ldots, \texttt{pac}(\texttt{s}^n\texttt{0})$, where `op pac : Nat -> Packet` is the constructor for packets. The bits sent by both Sender and Receiver into the communication channels are modelled by the boolean values `true` and `false`. The communication channels and the packets received by the Receiver are modelled by sequences.

(a) `channel1` consists of sequences of pairs of bits and packets of the form $< \texttt{b}_1, \texttt{p}_1 >, \ldots, < \texttt{b}_n, \texttt{p}_n >$.
(b) `channel2` consists of sequences of bits of the form $\texttt{b}_1, \ldots, \texttt{b}_n$.
(c) The `list` of packets received by Receiver consists of sequences of packets of the form $\texttt{p}_1, \ldots, \texttt{p}_n$.

In the OTS/CafeOBJ method the transitions between the states of the system are modelled with constructor operators. For the ABP specification, the constructors are the following ones:

Constructor	Meaning
`init : -> Sys`	Initial state
`rec1 : Sys -> Sys`	Sender receives bits
`rec2 : Sys -> Sys`	Receiver receives pairs of bits & packets
`send1 : Sys -> Sys`	Sender sends pairs of bits & packets
`send2 : Sys -> Sys`	Receiver sends bits
`drop1 : Sys -> Sys`	Dropping one element of channel1
`drop2 : Sys -> Sys`	Dropping one element of channel2

The structure of a state is abstracted by the following observers, each one returning an observable information about the state:

Observer	Meaning
`channel1 : Sys -> Channel1`	Sender-to-Receiver channel
`channel2 : Sys -> Channel2`	Receiver-to-Sender channel
`bit1 : Sys -> Bit`	Sender's bit
`bit2 : Sys -> Bit`	Receiver's bit
`next : Sys -> Nat`	Number of packet sent by Sender
`list : Sys -> List`	Lists of packets received by Receiver

The meaning of an observer is formally described by means of (conditional) equations. For instance, the value of `bit1` after the application of `rec1` is described by the following $[CA]$-set of conditional equations:

```
bit1(rec1(S))= bit1(S)     if channel2(S)= empty
bit1(rec1(S))= bit1(S)     if B,C2 := channel2(S)⋀ B = not bit1(S)
bit1(rec1(S))= not bit1(S) if B,C2 := channel2(S)⋀ B = bit1(S)
```

More tricky is the specification of loosing data from the channels, because the dropped elements are arbitrarily chosen. We use "underspecified" operations to model the dropping actions. For instance, the value of `channel1` after the application of `drop1` is specified by two operations `p1 r1 : Sys -> Channel1` and the following $[CA]$-set of conditional equations:

```
channel1(drop1(S))= p1(S),r1(S) if p1(S),< B,P >,r1(S) := channel1(S)
channel1(drop1(S))= channel1(S) if match(channel1(S),p1(S),r1(S))= false
```

Roughly speaking, the arguments of `p1` and `r1` are constructor terms of the form $\sigma_n(\ldots\sigma_1(\texttt{init}))$, where $\sigma_i \in \{\texttt{rec1},\ldots,\texttt{drop2}\}$, and the values returned are sequences because the communication channels are protected. There are no equations to define `p1` and `r1`, meaning that each model has its own interpretation of `p1` and `r1`. If `channel1(S)` is matched by `p1(S),< B,P >,r1(S)` then the element `< B,P >` is dropped, otherwise, `drop1` does not affect the system state. Each model of the specification is deterministic but the non-deterministic behaviour consists of the different interpretations of the functions `p1` and `r1` into the models. An application of `drop1` to a given state does not change the values of `bit1`, `bit2`, `next` and `list`.

3.2 The Correctness Proof

We will explain our methodology by proving a safety property for ABP. The protocol is specified using only conditional equations. In order to avoid non-termination in the proof process, we use preorder axioms with the semantic of equality. The proof begins with the following four lemmas:

(1) If channel1 contains bit1 then all bits before bit1 are equal to bit1.

```
inv1 =def= B' => bit1(S) if
           C1,< B,P >,C2,< B',P' >,C3 := channel1(S) /\ B = bit1(S)
```

(2) All bits of channel1 are equal to bit1 when bit2 is equal to bit1.

```
inv2 =def= B => bit1(S) if
           C1,< B,P >,C2 := channel1(S) /\ bit2(S) = bit1(S)
```

(3) bit2 is equal to bit1 when channel2 contains bit1.

```
inv3 =def= bit2(S) = bit1(S) if
           D1,B,D2 := channel2(S) /\ B = bit1(S)
```

(4) If channel2 contains bit1 then all the bits before bit1 are equal to bit1.

```
inv4 =def= B' => bit1(S) if
           D1,B,D2,B',D3 := channel2(S) /\ B = bit1(S)
```

Remark 3. The invariants inv1, inv2 and inv4 are specified as preorder axioms. As equations the above invariants would cause non-termination: when $[SI]$ is applied to the goal $\texttt{ABP} \vdash \{\texttt{inv1}, \texttt{inv2}, \texttt{inv3}, \texttt{inv4}\}$, invi are added as hypotheses to the specification ABP; then an application of inv1, for example, to reduce a term implies the evaluation of the condition C1,< B,P >,C2,< B',P' >,C3 := channel1(S) that requires another application of inv1, which produces a non-termination process. Since these hypotheses are needed in the verification process, i.e., they must be executable, we choose to formalise them as preorder axioms. In this way, the termination property is preserved and the new (equational and preorder) axioms are executable.

The invariants cannot be proved independently (without a simultaneous induction scheme) since the proof of each invariant depends on the rest.

Secondly, we prove the following invariant by (ordinary) structural induction using the invariants above.

```
inv5 =def= P => pac(next(S)) if C1,< B,P >,C2 := channel1(S) /\ B = bit1(S)
```

The invariant inv5 says that if channel1 contains a pair < B,P > and B is equal to bit1 then P is equal to pac(next).

When Receiver gets the nth packet it has received $\texttt{pac}(0), \ldots, \texttt{pac}(n)$, in this order. Each $\texttt{pac}(i)$ for $i = 0, \ldots, n$ has been received only once, and no other packet have been received. This property is formalised below.

$\texttt{goal1} \stackrel{def}{=} \texttt{mk(next)(S) = pac(next(S)),list(S) if bit1(S) = not bit2(S)}$

$\texttt{goal2} \stackrel{def}{=} \texttt{mk(next)(S) = list(S) if bit1(S) = bit2(S)}$

where `mk : Nat -> List` is defined by $\begin{cases} \texttt{mk}(0) = \texttt{pac}(0), \\ \texttt{mk}(\texttt{s n}) = \texttt{pac}(\texttt{s n})\texttt{mk}(\texttt{n}). \end{cases}$ The formulas `goal1` and `goal2` are proved by simultaneous induction using `inv2`, `inv3` and `inv5`. We present the proof of the four invariants as the rest of the proof is similar.

Let $\texttt{INV} = \texttt{ABP} \cup (\mathbb{S}ig(\texttt{ABP}), \{\texttt{lemma-inc}\})$, where

$$\texttt{lemma-inc} \stackrel{def}{=} \texttt{true => false if not bit1(S) => bit1(S)}$$

The above axiom says that `not bit1(S)` and `bit1(S)` are different for all system states `S`. Note that `lemma-inc` is not executable since the left-hand side term is ground and the condition contains the variable `S`. The proof of $\texttt{INV} \vdash \{\texttt{inv1}, \texttt{inv2}, \texttt{inv3}, \texttt{inv4}\}$ is performed with CITP using the proof strategy $[SI]$, $[CA]$, $[SC]$, $[TC]$, $[IC]$, $[IP]$, $[RD]$, $[CT]$. The goal is generated by the following command:

```
(goal INV |-
crl [inv1]: B1:Bit => bit1(S:Sys) if
 C1:Channel1,< B:Bit,P:Packet >,C2:Channel1,< B1:Bit,P1:Packet >,C3:Channel1
 := channel1(S:Sys) ∧ B:Bit = bit1(S:Sys) ;
crl [inv2]: B:Bit => bit1(S:Sys) if
 C1:Channel1,< B:Bit,P:Packet >,C2:Channel1 := channel1(S:Sys) ∧
 bit2(S:Sys) = bit1(S:Sys) ;
ceq [inv3]: bit2(S:Sys) = bit1(S:Sys) if
 C2:Channel2,B:Bit,C3:Channel2 := channel2(S:Sys) ∧ B:Bit = bit1(S:Sys)
 [metadata "enhanced"];
crl [inv4]: B':Bit => bit1(S:Sys) if
 D1:Channel2,B:Bit,D2:Channel2,B':Bit,D3:Channel2 := channel2(S:Sys) ∧
 B:Bit = bit1(S:Sys) ; )
```

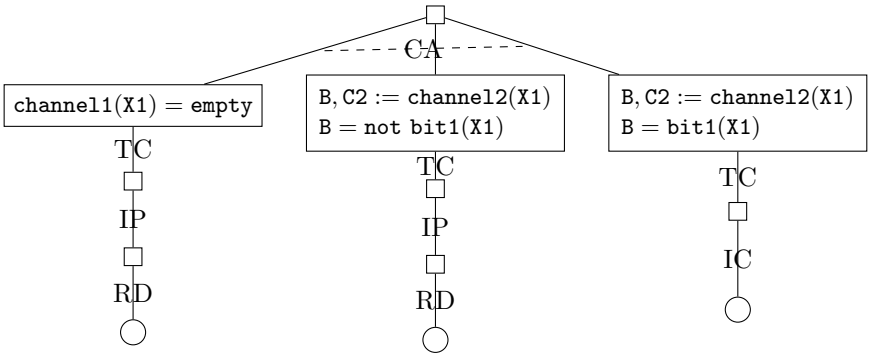
CITP discharges an equation $t_1 = t_2$ with the attribute `"enhanced"` by proving t_1 `=>` t_2. The proof consists of the sequence of commands described below.

```
(set ind on S:Sys .)          --- rec1 ---
(apply SI .)                  (init lemma-inc by S:Sys <- X1 .) (auto .)

--- init ---                  --- rec2 ---
(auto .)                      (init lemma-inc by S:Sys <- X1 .) (auto .)

--- drop1 ---                 --- send1 ---
(auto .)                      (auto .)

--- drop2 ---                 --- send2 ---
(auto .)                      (auto .)
```

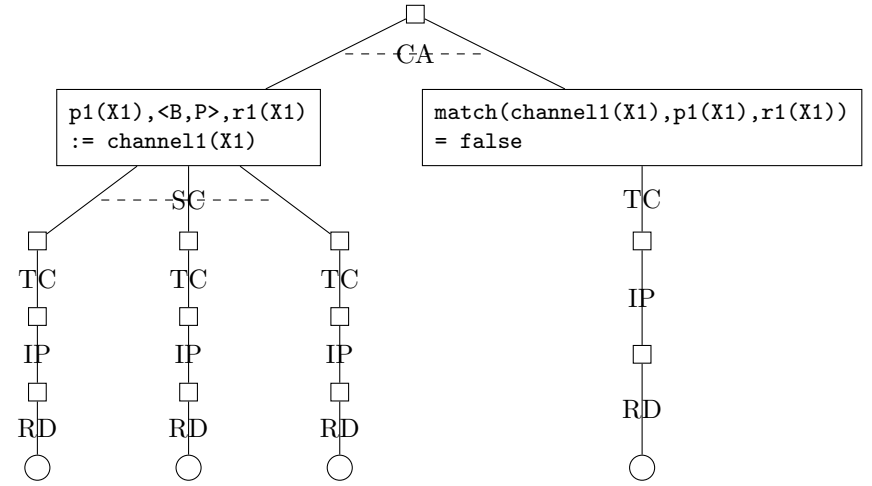
The variable `S` is selected for induction. By applying $[SI]$ we obtain seven subgoals corresponding to each of the seven constructors:

$$\texttt{INV} * \iota_{\texttt{X1}} \cup (\mathbb{S}ig(\texttt{INV})[\texttt{X1}], \Gamma[S \leftarrow \texttt{X1}]) \vdash \Gamma[S \leftarrow \sigma(\texttt{X1})]$$

where $\iota_{\mathtt{X1}} : \mathbb{S}ig(\mathtt{INV}) \hookrightarrow \mathbb{S}ig(\mathtt{INV})[\mathtt{X1}]$, X1 is a constant representing an arbitrary state of the system, $\sigma \in \{\mathtt{init}, \mathtt{rec1}, \mathtt{rec2}, \mathtt{send1}, \mathtt{send2}, \mathtt{drop1}, \mathtt{drop2}\}$, and $\Gamma = \{\mathtt{inv1}, \mathtt{inv2}, \mathtt{inv3}, \mathtt{inv4}\}$. For the cases $\sigma = \mathtt{rec1}$ and $\sigma = \mathtt{rec2}$, `lemma-inc` is instantiated by substituting X1 for S. Then each of the remaining goals is split into four subgoals corresponding to each invariant. We obtain 28 subgoals that are discharged automatically by the command `(auto .)` which uses the proof strategy $[SI]$, $[CA]$, $[SC]$, $[TC]$, $[IC]$, $[IP]$, $[RD]$, $[CT]$. Note that the application of $[SI]$ at this point leaves the goal unchanged. The proof tree corresponding to $\mathtt{INV} * \iota_{\mathtt{X1}} \cup (\mathbb{S}ig(\mathtt{INV})[\mathtt{X1}], \Gamma[S \leftarrow \mathtt{X1}], \mathtt{lemma\text{-}inc}[S \leftarrow \mathtt{X1}]) \vdash \mathtt{inv2}[S \leftarrow \mathtt{rec1}(\mathtt{X1})]$ is depicted in the figure below:



Note that $[CA]$ is performed w.r.t. the $[CA]$-subterm `bit1(rec1(X1))`. There are three cases given by the conditions of the $[CA]$-set of sentences which defines the value of `bit1` after the execution of `rec1` at a given state. The proof tree corresponding to $\mathtt{INV} * \iota_{\mathtt{X1}} \cup (\mathbb{S}ig(\mathtt{INV})[\mathtt{X1}], \Gamma[S \leftarrow \mathtt{X1}]) \vdash \mathtt{inv2}[S \leftarrow \mathtt{drop1}(\mathtt{X1})]$ is depicted in the following figure.



In this case, CA is performed w.r.t. the $[CA]$-subterm `channel1(drop1(X1))`. The interested reader can look into `http://www.ldl.jaist.ac.jp/citp/` for tool demonstration.

4 Conclusions

We presented a methodology for proving inductive properties of OTS. The proposed method aims at automating the proof score approach to verification. We revise the entailment system in [9] to increase its efficiency in applications and we enrich it with a set of specific proof rules for initial data types that are often used in our methodology. We define proof tactics that preserve confluence and termination of the specifications in the proof process, and we propose a proof strategy to apply the tactics automatically. The viability of the methodology is demonstrated by CITP, a prototype tool implementing the methodology. We used the ABP example in order to exhibit the main strong points of the proposed methodology. Algebraic specification languages have standard libraries with predefined modules. In order to perform verification of complex software systems it is crucial to have tactics for the initial data types that are often used in practice such as booleans, sequences or natural numbers. The challenge is how to integrate these tactics with the ones for loose semantics and to push the boundaries of automation.

The logical frameworks underlying tools like Circ [19,16] or Maude ITP [18] do not include preorder axioms, and are not based on constructors. Circ implements a similar tactic with the *Case Analysis* proposed here. *Case analysis* is interactive in Maude ITP. Another paper that uses the ABP as a benchmark example is [14]. The proof in [14] is coinductive and it is based on the circular coinductive rewriting algorithm [15] implemented in the BOBJ system. In [14], the unreliability of the communication channels is modelled by "fair streams", while here it is modelled by a special dropping operation that is closer to a real description. With the OTS approach, the ABP specification is closer to a faithful representation, and since all data types are specified in detail, the proof becomes more complex.

Future work includes testing of the methodology on other case studies, and increasing the automation level.

Acknowledgements

The work presented in this paper has been partially supported by the Japanese Contract Kakenhi 23220002, and by the Romanian Contract 161/15.06.2010, SMISCSNR 602-12516 (DAK).

References

1. M. Bidoit and R. Hennicker. Constructor-based observational logic. *J. Log. Algebr. Program.*, 67(1-2):3–51, 2006.
2. M. Bidoit, R. Hennicker, and A. Kurz. Observational logic, constructor-based logic, and their duality. *Theor. Comput. Sci.*, 3(298):471–510, 2003.
3. T. Borzyszkowski. Logical systems for structured specifications. *Theor. Comput. Sci.*, 286(2):197–245, 2002.

4. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.
5. R. Diaconescu and K. Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, volume 6 of *AMAST Series in Computing*. World Scientific, 1998.
6. R. Diaconescu and K. Futatsugi. Logical foundations of CafeOBJ. *Theor. Comput. Sci.*, 285(2):289–318, 2002.
7. K. Futatsugi. Verifying Specifications with Proof Scores in CafeOBJ. In *ASE*, pages 3–10. IEEE Computer Society, 2006.
8. K. Futatsugi, J. A. Goguen, and K. Ogata. Verifying Design with Proof Scores. In B. Meyer and J. Woodcock, editors, *VSTTE*, volume 4171 of *Lecture Notes in Computer Science*, pages 277–290. Springer, 2005.
9. K. Futatsugi, D. Găină, and K. Ogata. Principles of proof scores in CafeOBJ. *Theor. Comput. Sci.*, 464:90–112, 2012.
10. D. Găină and K. Futatsugi. Initial Semnatics in Logics with Constructors. *J. Log. Comput.*, 2013. http://dx.doi.org/10.1093/logcom/exs044.
11. D. Găină, M. Zhang, Y. Chiba, and Y. Arimoto. Constructor-based Inductive Theorem Prover. In S. Milius and R. Heckel, editors, *CALCO*, volume 8089 of *Lecture Notes in Computer Science*. Springer, 2013.
12. J. Goguen. *Theorem Proving and Algebra*. 1994.
13. J. A. Goguen and R. Burstall. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.
14. J. A. Goguen and K. Lin. Behavioral Verification of Distributed Concurrent Systems with BOBJ. In *3rd International Conference on Quality Software (QSIC)*, pages 216–, 2003.
15. J. A. Goguen, K. Lin, and G. Rosu. Circular Coinductive Rewriting. In *ASE*, pages 123–132, 2000.
16. E. Goriac, D. Lucanu, and G. Rosu. Automating Coinduction with Case Analysis. In *ICFEM 2010*, volume 6447 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2010.
17. D. Găină, K. Futatsugi, and K. Ogata. Constructor-based Logics. *J. UCS*, 18(16):2204–2233, 2012.
18. J. D. Hendrix. *Decision Procedures for Equationally Based Reasoning*. Technical Report, UIUC, 2008.
19. D. Lucanu, E.-I. Goriac, G. Caltais, and G. Roşu. CIRC : A Behavioral Verification Tool based on Circular Coinduction. In *CALCO 2009*, volume 5728 of *LNCS*, pages 433–442. Springer, 2009.
20. J. Meseguer. Order-Sorted Parameterization and Induction. In J. Palsberg, editor, *Semantics and Algebraic Specification*, volume 5700 of *Lecture Notes in Computer Science*, pages 43–80. Springer, 2009.
21. K. Ogata and K. Futatsugi. Flaw and modification of the iKP electronic payment protocols. *Inf. Process. Lett.*, 86(2):57–62, 2003.
22. K. Ogata and K. Futatsugi. Simulation-based Verification for Invariant Properties in the OTS/CafeOBJ Method. *Electr. Notes Theor. Comput. Sci.*, 201:127–154, 2008.
23. D. Sannella and A. Tarlecki. Specifications in an Arbitrary Institution. *Inf. Comput.*, 76(2/3):165–210, 1988.