Analele Ştiinţifice ale Universităţii "Alexandru Ioan Cuza" din Iaşi
Informatică, Tomul XII, 2002, 45–53

Scientific Annals of the "Alexandru Ioan Cuza" University of Iaşi
Computer Science Section, Tome XII, 2002, 45–53

Initial Logic and Induction in Algebraic Specification

Dorel LUCANU[1]

Abstract. The initial truth refers to those properties which are valid in initial models. In this paper we show how the initial truth can be organized as an institution and introduce a valid inference rule system with which we can develop proofs by induction in this logic.

AMS Subject Classification: 68Q65, 68Q60

Keywords and phrases: abstract data type, algebraic specification, induction, initial constraints, initial constraints deduction, initial many sorted algebra institution, initial order sorted algebra institution, initial membership algebra institution

1 Introduction

An abstract data type (ADT) defined by a many sorted specification (Σ, E) is the class of initial (Σ, E)-algebras. Therefore the sentences that are true about an ADT (Σ, E) are the same with those satisfied by the initial (Σ, E)-algebra. It is known that the equational deduction [6] is not enough to prove all sentences satisfied by the initial algebra. A powerful tool to prove initially satisfied sentences is induction. The general schema for proving a property $(\forall x)p(x)$ by induction is as follows:

[1]Universitatea "Al.I.Cuza" Iaşi

1. Define a subsignature whose functional symbols are called *constructors*. The general requirement for constructors is "to define the sort of x".
2. Use the equations to prove that the set of terms satisfying p is closed under constructors.

Usually the focus is on the second step. Very often the definition for constructors is given at the intuitive level (two exceptions are [4] and [2]) and the proof of the fact that a given subsignature is a signature of constructors is completely skipped. Consider e.g. the very well known example of the natural numbers (we use the OBJ3 syntax):

```
obj NAT is
  sort Nat .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
  op _+_ : Nat Nat -> Nat .
  vars M N : Nat .
  eq M + 0 = M .
  eq M + s N = s(M + N) .
endo
```

It is well known that the constructors for the natural numbers are the zero constant and the successor operation. A very simple property that can be proved by induction is $(\forall N)0 + N = N$. This consists of the following two steps:

1. $Eq(\texttt{NAT}) \models 0 + 0 = 0$.
2. $Eq(\texttt{NAT}) \cup \{0 + n\} \models 0 + s\, n = s\, n$, where n is a constant of sort `Nat`.

The verification of the two properties is straightforward. But how can we prove that the zero and successor operations are indeed constructors for `NAT`? To answer this question we need a formal definition for constructors. This can be given by using *constraints* [5].[2] We first include the definition of constructors in a separated specification:

[2]Initially we didn't use explicitly the constraints. Joseph Goguen suggested us that our approach uses a particular case of constraints.

```
obj NATC is
  sort Nat .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
endo
```

Usually, it is accepted that NATC is a constructor-specification for NAT iff the unique homomorphism $h : \mathbb{M}_{\mathtt{NATC}} \to \mathbb{M}_{\mathtt{NAT}}\restriction_{\mathtt{NATC}}$ is surjective, where $\mathbb{M}_{\mathtt{NATC}}$ is the initial NATC-algebra and $\mathbb{M}_{\mathtt{NAT}}\restriction_{\mathtt{NATC}}$ is the restriction of the initial NAT-algebra to the signature of NATC. This property is called *sufficient generation* in [2]. The theory becomes more general and flexible if we formalize this property in terms of constraints [5]: NATC is a *constructor-specification* for NAT iff the initial model $\mathbb{M}_{\mathtt{NAT}}$ satisfies the (initial) constraint $\mathfrak{ic}_{\mathtt{NATC},\mathtt{NAT}} = \langle(\emptyset,\emptyset) \hookrightarrow (Sig(\mathtt{NATC}), Eq(\mathtt{NATC})), Sig(\mathtt{NATC}) \hookrightarrow Sig(\mathtt{NAT})\rangle$. Now, proving that NATC is a constructor-specification for NAT is equivalent with showing that a given initial model satisfies a given constraint in the institution of constraints. But this institution is not very useful in practice because we have no yet a deductive calculus for constraints. Therefore we constructed a different institution, more appropriate to our aim.

When we are dealing with algebraic specification, we are using different logics: many sorted equational logic, order sorted equational logic, and/or membership equational logic (we referred here only those logics where the main focus is on the initial semantics). We associate with each one of these logics a new logic called initial logic. Following Goguen's slogan "semantics first", we define first the initial logics in terms of models and satisfaction relation. In other words, we organize each initial logic as an institution. Then we define a deductive calculus for proving by induction initial sentences. Recall that initial sentences include both equations and initial constraints. Initial sentences in initial membership equational logic includes also membership assertions. The advantage of such a deductive calculus is that we can develop then systems able to prove initial sentences by induction. A first step in this sense was already made by Jose Meseguer, Francisco Duran, and Manuel Clavel with the ITP (Inductive Theorem Prover) system written in Maude. ITP is based on membership equational logic and is

not completely automated because is not able to prove automatically initial constraints but it can do it into an assisted manner. The aim of this paper is to contribute to the theoretical foundation needed to increase the performance of systems like ITP.

The paper is organized as follows. Section 2 includes the references to several representative papers where the unfamiliar reader can find the preliminary definitions and notations used in the paper. Section 3 defines the initial constraints as a particular case of constraints. Section 4 shows how the initial logic can be organized as an institution. Section 5 presents in details a deductive calculus for proofs by induction in the initial many sorted equational logic. The difference from ITP is that we do not use the embedding of the many sorted equational logic into membership equational logic (MEL). Our calculus is a component of the initial many sorted equational logic itself. Moreover, our approach is more general because we use constructors defined via a specification morphism (ITP uses the particular case when this morphism is inclusion). Section 6 shows how the deductive system introduced in Section 5 can be adapted for the order sorted case. Section 7 includes some examples which shows the relationship between our approach and ITP for the case of the membership equational logic. We end the paper with some concluding remarks.

2 Preliminaries

We assume the reader familiar with the many sorted algebra theory [6], order sorted algebra [7], membership equational logic [11, 1], algebraic specification language like OBJ3 [8] and Maude [3], and institution theory [5, 10]. However, in order to make easier the reading of the paper, the appendix A includes the main definitions concerning institutions and constraints.

3 Initial constraints

In this section, $\mathcal{I} = (\underline{\mathbf{Sign}}, \mathbf{sen}, \underline{\mathbf{Mod}}, \models)$ denotes an arbitrary institution.

Notation 1 *Let Σ be a signature in* $\underline{\mathbf{Sign}}$ *and let E be a set of Σ-sentences. By $\varepsilon_{\Sigma,E}$ we denote the specification inclusion $\varepsilon_{\Sigma,E} : (\emptyset, \emptyset) \hookrightarrow (\Sigma, E)$.*

Definition 2 Initial constraints.
An initial Σ-constraint *is a pair of the form $\mathfrak{ic} = \langle \varepsilon_{\Sigma^c,E^c}, \Theta : \Sigma^c \to \Sigma \rangle$, where Σ^c and Σ are signatures in* $\underline{\mathbf{Sign}}$ *and E^c is a set Σ^c-equations.*

It is easy to see that the initial constraints are a particular case of constraints. Because Σ^c, E^c determines uniquely the morphism $\varepsilon_{\Sigma^c,E^c}$, we often use the shorter notation $\langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ in order to specify an initial constraint. The satisfaction of initial constraints is a special case of Definition 43:

Definition 3 Satisfaction of initial constraints.
Let $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ be an initial constraint and let M be a Σ-model.

1. *We say that M* satisfies $\mathfrak{ic}$ *iff:*
   (a) *$M{\restriction_\Theta} \models E^c$, and*
   (b) *the unique Σ^c-homomorphism $h : \mathbb{IM}_{\Sigma^c,\mathbf{E}^c} \to M{\restriction_\Theta}$ is surjective, where $\mathbb{IM}_{\Sigma^c,\mathbf{E}^c}$ denotes the initial (Σ^c, E^c)-model.*
2. *We say that M* completely satisfies $\mathfrak{ic}$ *iff:*
   (a) *$M{\restriction_\Theta} \models E^c$, and*
   (b) *$\mathbb{IM}_{\Sigma^c,\mathbf{E}^c}$ and $M{\restriction_\Theta}$ are isomorphic.*

The complete satisfaction is the *sufficient completeness* property [1, 2] stated in terms of initial constraints. In this paper we focus only on the first satisfaction relation.

4 Initial Logic

In this section we define the ingredients of the institution $\mathcal{IT}(\mathcal{I})$ corresponding to the initial truth from a given institution $\mathcal{I} = (\underline{\mathbf{Sign}}, \mathbf{sen}, \underline{\mathbf{Mod}}, \models)$.

Definition 4 Initial sentences.
Let Σ be a signature in $\underline{\mathbf{Sign}}$. *An* initial Σ-sentence *is either a Σ-sentence or an initial Σ-constraint.*

Definition 5 Initial satisfaction.
Let Σ be a signature, let E be a set of initial Σ-sentences, and let α be an initial Σ-sentence. We say that (Σ, E) initially satisfies α, *we write $(\Sigma, E) \mathrel{|\!\cong} \alpha$, iff the initial model $\mathbb{IM}_{\Sigma,\mathbf{E}}$ satisfies α, i.e., $\mathbb{IM}_{\Sigma,\mathbf{E}} \models \alpha$.*

Theorem 6 Institution $\mathcal{IT}(\mathcal{I})$.
Consider the following notations:

1. *the functor* **isen** *maps a signature Σ into the set of all initial Σ-sentences;*
2. *the functor* $\underline{\mathbf{IMod}} : \underline{\mathbf{Sign}} \to \underline{\mathbf{Cat}}$ *maps a signature Σ into the subcategory of the specifications of the form (Σ, E);*
3. *if $\Phi : \Sigma \to \Sigma'$ is a morphism of signatures then $\Phi(\alpha)$, the translation by Φ of the initial Σ-sentence α, is defined as usual and $(\Sigma', E')\restriction_\Phi$, the Φ-reduct of the specification (Σ', E'), is the specification $(\Sigma, (\mathbb{IM}_{\Sigma',\mathbf{E}'}\restriction_\Phi)^*)$, where M^* denotes the set of initial sentences satisfied by the model M.*

Then $IMSA = (\underline{\mathbf{MSSign}}, \mathbf{isen}, \underline{\mathbf{IMod}}, \mathrel{|\!\cong})$ is an institution.

Proof: We have to prove the satisfaction condition. Consider $\Phi : \Sigma \to \Sigma'$ a signature morphism, (Σ', E') a specification, and α an initial Σ-sentence. Then:

$(\Sigma', E') \mathrel{|\!\cong} \Phi(\alpha)$	iff	(by definition)
$\mathbb{IM}_{\Sigma',\mathbf{E}'} \models \Phi(\alpha)$	iff	($\mathcal{IC}(MSA)$ is an institution)
$\mathbb{IM}_{\Sigma',\mathbf{E}'}\restriction_\Phi \models \alpha$	iff	(M is an initial (Σ, M^*)-model)
$(\Sigma, (\mathbb{IM}_{\Sigma',\mathbf{E}'}\restriction_\Phi)^*) \mathrel{|\!\cong} \alpha$	iff	(by definition)
$(\Sigma', E')\restriction_\Phi \mathrel{|\!\cong} \alpha$.		

□

Definition 7 Constructors for a specification.
Let $\Phi : (\Sigma^c, E^c) \to (\Sigma, E)$ *be a morphism of specifications.*

1. (Σ^c, E^c) *is a* constructor-specification *for* (Σ, E) *iff the initial constraint* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Phi \restriction_{\Sigma^c} \colon \Sigma^c \to \Sigma \rangle$ *is a semantical consequence of* E *w.r.t. satisfaction relation.*
2. (Σ^c, E^c) *is a* complete constructor-specification *for* (Σ, E) *iff the initial constraint* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Phi \restriction_{\Sigma^c} \colon \Sigma^c \to \Sigma \rangle$ *is a semantical consequence of* E *w.r.t. complete satisfaction relation.*

5 Induction in Initial Many Sorted Logic

Let MSA denote the institution of many sorted algebra and let $IMSA$ denote the institution $\mathcal{IT}(MSA)$. In this section we present induction as a proof subcalculus in $IMSA$ as it was defined in [9].

In the next definition we use the following notation: if e is a Σ-equation modulo E of the form $(\forall x) t_1 = t_2$ and t is a ground Σ-term, then $e(x \leftarrow t)$ denotes the equation $(\forall \emptyset) t_1(x \leftarrow t) = t_2(x \leftarrow t)$.

Definition 8 Equation deduction by induction.
Let $\mathfrak{ic}$ *be an initial constraint of the form* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *and let* E *be a set of* Σ*-equations. The relation* $\models_\Sigma$ *is defined by the following deduction rules:*

Equational deduction. *For each* Σ*-equation* e*,*

$$(DE)\ \frac{E \vdash_\Sigma e}{E \cup \{\mathfrak{ic}\} \models_\Sigma e},$$

Induction. *For each* Σ*-equation* e *of the form* $(\forall x) t_1 = t_2$ *with* x *a variable of sort* $\Theta(s)$ *and* $s \in \mathrm{sort}(\Sigma^c)$*,*

$$(IND)\ \frac{\begin{array}{l}(\forall f \in \Sigma^c_{[],s}) E \cup \{\mathfrak{ic}\} \models_\Sigma e(x \leftarrow \Theta(f)) \\ \text{and} \\ (\forall f \in \Sigma^c_{s_1 \ldots s_n, s}, t_i \in T_{\Sigma, s_i}\ i = 1, \ldots, n) \\ E \cup \{e(x \leftarrow t_i) \mid s_i = s, i \in \{1, \ldots, n\}\} \cup \{\mathfrak{ic}\} \models_\Sigma \\ e(x \leftarrow \Theta(f)(t_1 \ldots t_n))\end{array}}{E \cup \{\mathfrak{ic}\} \models_\Sigma e}$$

Theorem 9 Soundness of equation deduction by induction.
Consider the notations from Definition 8. If $(\Sigma, E) \models \mathfrak{ic}$ *and* $E \cup \{\mathfrak{ic}\} \models_\Sigma e$, *then* $(\Sigma, E) \models_\Sigma e$.

Proof: We have only to show the soundness of the rule (IND). Consider e a Σ-equation of the form $(\forall x)t_1 = t_2$ where x is a variable of sort $\Theta(s)$. The rule (IND) implies:

$$\{[t] \in (\mathbb{M}_{\Sigma^c,\mathbf{E}^c})_s \mid [t_1(x \leftarrow \Theta(t))]_E = [t_2(x \leftarrow \Theta(t))]_E\} = (\mathbb{M}_{\Sigma^c,\mathbf{E}^c})_s. \quad (1)$$

Because $(\Sigma, E) \models \mathfrak{ic}$, it follows that there is a surjective homomorphism $h : \mathbb{M}_{\Sigma^c,\mathbf{E}^c} \to \mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta$. We show now that if t is a Σ^c-term, then $[\Theta(t)]_E = h([t]_{E^c})$. We proceed by structural induction on t. If $t = f \in \Sigma^c_{[],s}$ then:[3]

$$[\Theta(t)]_E = [\Theta(f)]_E \in (\mathbb{M}_{\Sigma,\mathbf{E}})_{\Theta(s)} = (\mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta)_s \qquad \text{and}$$
$$h([t]_{E^c}) = h([f]_{E^c}) = h([\![f]\!]_{\mathbb{M}_{\Sigma^c,\mathbf{E}^c}}) = [\![f]\!]_{\mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta} = [\Theta(f)]_E.$$

If $t = f(t'_1 \dots t'_n)$, then:

$$\begin{aligned}
&h([f(t'_1 \dots t'_n)]_{E^c}) = \\
&[\![f]\!]_{\mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta}(h([t'_1]_{E^c}), \dots, h([t'_n]_{E^c})) = \qquad \text{(by induction hypothesis)} \\
&[\![f]\!]_{\mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta}([\Theta(t'_1)]_E), \dots, [\Theta(t'_n)]_E)) = \qquad ((\mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta)_s = (\mathbb{M}_{\Sigma,\mathbf{E}})_{\Theta(s)}) \\
&[\Theta(f)(\Theta(t'_1), \dots, \Theta(t'_n))]_E = \\
&[\Theta(f(t'_1, \dots, t'_n)]_E.
\end{aligned}$$

The proof by structural induction is finished now. It follows that for any $[t'] \in \mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta$ there exists $[t] \in \mathbb{M}_{\Sigma^c,\mathbf{E}^c}$ such that $[t']_E = [\Theta(t)]_E$. If t' is of sort $\Theta(s)$, then $[t_1(x \leftarrow t')]_E = [t_1(x \leftarrow \Theta(t))]_E = [t_2(x \leftarrow \Theta(t))_E = [t_2(x \leftarrow t')]_E$. Using this relation together with the equality $(\mathbb{M}_{\Sigma,\mathbf{E}})_{\Theta(s)} = (\mathbb{M}_{\Sigma,\mathbf{E}}\restriction_\Theta)_s$ in (1) we obtain:

$$\{[t'] \in (\mathbb{M}_{\Sigma,\mathbf{E}})_{\Theta(s)} \mid [t_1(x \leftarrow t')]_E = [t_2(x \leftarrow t')]_E\} = (\mathbb{M}_{\Sigma,\mathbf{E}})_{\Theta(s)} \quad (2)$$

that is equivalent to say that $\mathbb{M}_{\Sigma,\mathbf{E}} \models e$. □

[3] $[\![f]\!]_M$ denotes the interpretation of the symbol f in algebra M.

Corollary 10 *Consider the notations from Definition 8. If* (Σ^c, E^c) *is a constructor-specification for* (Σ, E) *and* $E \cup \{\mathfrak{ic}\} \models_\Sigma e$, *then* $E \models\!\!\models_\Sigma e$.

Definition 8 and Theorem 9 formalizes the equation deduction by induction in the terms of our institution IMSA. In what follows we focus on deduction of initial constraints. We give first a preliminary definition.

Definition 11 *Membership extension of a initial constraint.*
Let $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *be a initial constraint. Then:*

1. $MXSig(\mathfrak{ic}) = \Sigma \cup Sig(\texttt{BOOL}) \cup \{s : \Theta(s) \to \texttt{Bool} \mid s \in sort(\Sigma^c)\}$;
2. $MXEq(\mathfrak{ic}) = \{s(\Theta(f)) = \texttt{true} \mid f \in \Sigma^c_{[],s}\} \bigcup$
   $\{(\forall x_1, \ldots, x_n) s(\Theta(f)(x_1 \ldots x_n)) = \bigwedge_{i=1}^n s_i(x_i) \mid f \in \Sigma^c_{s_1 \ldots s_n, s}\}$.

Theorem 12 *Let* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *be a initial constraint. If* $(\Sigma, E) \models\!\!\models \mathfrak{ic}$ *then* $[\![s]\!]_{\mathbb{M}_{\texttt{MXSig(ic)},\texttt{MXEq(ic)} \cup \texttt{E}}}([t]) = true$ *for all* $[t] \in (\mathbb{M}_{\Sigma,\texttt{E}})_s$, $s \in \mathrm{sort}(\Sigma^c)$.

Proof: Because the unique homomorphism $h : \mathbb{M}_{\Sigma^c,\texttt{E}^c} \to \mathbb{M}_{\Sigma,\texttt{E}} \restriction_\Theta$ is surjective, it follows that for any $[t] \in \mathbb{M}_{\Sigma,\texttt{E}}$ there exists $[t^c] \in \mathbb{M}_{\Sigma^c,\texttt{E}^c}$ such that $[t] = [\Theta(t^c)]$. It is easy to check by structural induction that $[\![s]\!]_{\mathbb{M}_{\texttt{MXSig(ic)},\texttt{MXEq(ic)} \cup \texttt{E}}}([t^c]) = true$. □

Corollary 13 *Let* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *be a initial constraint. If* $(\Sigma, E) \models\!\!\models \mathfrak{ic}$ *then* $\mathbb{M}_{\texttt{MXSig(ic)},\texttt{MXEq(ic)} \cup \texttt{E}} \cong \mathbb{M}_{\Sigma,\texttt{E}}$.

Definition 14 *Initial constraint deduction.*
The relation $\models$ *is extended by the following deduction rules:*

Reflexivity. *For each many sorted specification* (Σ, E),

$$(IC_R)\ \frac{}{E \models_\Sigma \langle \Sigma, E, \mathrm{id}_\Sigma \rangle}.$$

Derived operations. *For each initial constraint* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *with* $\Sigma = \Theta(\Sigma^c) \cup \{f : \Theta(s_1) \cdots \Theta(s_n) \to \Theta(s)\}$ *and* E *a set of* Σ*-equations,*

$$(IC_f)\ \frac{\begin{array}{l}E \models_\Sigma \Theta(E^c),\\ E \cup MXEq(\mathfrak{ic}) \cup \{\mathfrak{ic}\} \models_{MXSig(\mathfrak{ic})}\\ (\forall x_1,\ldots,x_n) s(f\ (x_1 \ldots x_n))=\texttt{true if} \bigwedge_{i=1}^n s_i(x_i)=\texttt{true}\end{array}}{E \models_\Sigma \mathfrak{ic}}.$$

Transitivity. *For all initial constraints* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma\rangle$ *and* $\mathfrak{ic}' = \langle \Sigma, E, \Theta' : \Sigma \to \Sigma'\rangle$,

$$(IC_T)\ \frac{E \models_\Sigma \mathfrak{ic},\ \ E' \models_{\Sigma'} \mathfrak{ic}'}{E \cup E' \models_\Sigma \langle \Sigma^c, E^c, \Theta; \Theta' : \Sigma^c \to \Sigma'\rangle}.$$

Equation addition. *For each initial constraint* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma\rangle$ *and* E, E' *two sets of* Σ*-equations,*

$$(IC_{eq})\ \frac{E \models_\Sigma \mathfrak{ic},\ \ E \subseteq E'}{E' \models_\Sigma \mathfrak{ic}}$$

Disjoint signature addition. *For each initial constraint* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma\rangle$, Σ' *a signature, and* E *a set of* Σ*-equations,*

$$(IC_{op})\ \frac{E \models_\Sigma \mathfrak{ic},\ \ sort(\Sigma) \cap sort(\Sigma') = \emptyset}{E \models_{\Sigma'} \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \cup \Sigma'\rangle}$$

Sum. *For all initial constraints* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma\rangle$ *and* $\mathfrak{ic}' = \langle \Sigma'^c, E'^c, \Theta' : \Sigma'^c \to \Sigma'\rangle$,

$$(IC_S)\ \frac{E \models_\Sigma \mathfrak{ic},\ \ E' \models_{\Sigma'} \mathfrak{ic}', E + E' \models_{\Sigma+\Sigma'} \{\mathfrak{ic}_1, \mathfrak{ic}_2\}}{E+E' \models_{\Sigma+\Sigma'} \langle (\Sigma^c, E^c)+(\Sigma'^c, E'^c), \Theta+\Theta' : \Sigma^c+\Sigma'^c \to \Sigma+\Sigma'\rangle}$$

where:

$$\mathfrak{ic}_1 = \langle (\Sigma, E), \Sigma \hookrightarrow \Sigma + \Sigma'\rangle$$
$$\mathfrak{ic}_2 = \langle (\Sigma', E'), \Sigma' \hookrightarrow \Sigma + \Sigma'\rangle.$$

Theorem 15 *Soundness of the initial constraint deduction.*
Let $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma\rangle$ *be an initial constraint. If* $E \models_\Sigma \mathfrak{ic}$ *then* $(\Sigma, E) \cong_\Sigma \mathfrak{ic}$.

Proof: The soundness of the reflexivity follows directly from definitions.
In order to prove the soundness of the rule (IC_f), we show first that if t is a ground Σ-term of sort s' such that $E \cup MXEq(\mathfrak{ic}) \cup \{\mathfrak{ic}\} \models_{MXSig(\mathfrak{ic})} (\forall \emptyset) s'(t) = \mathtt{true}$, then there exists a ground Σ^c-term t^c such that $E \models_{\Sigma} (\forall \emptyset) t = \Theta(t^c)$. This can be easily checked by structural induction on t. If $t \in \Sigma_{[],\Theta(s')}$, then there is $g \in \Sigma^c_{[],s'}$ such that $t = \Theta(g)$. If $t = \Theta(g)(t_1, \ldots, t_n)$ then $s_i(t_i) = \mathtt{true}$ for $i = 1, \ldots, n$ and, by inductive hypothesis, there exist $t^c_1, \ldots, t^c_n$ such that $E \models_{\Sigma} (\forall \emptyset) t_i = \Theta(t^c_i)$ for $i = 1, \ldots, n$. In that case we have $t^c = g(t^c_1, \ldots, t^c_n)$. Suppose that $t = f(t_1, \ldots, t_n)$ and $s' = s$. Because $MXEq(\mathfrak{ic})$ does not include any equation of the form $(\forall \ldots) s(f(\ldots)) = \ldots$, it follows that there exists $t' = \Theta(g)(\ldots)$ such that $E \models (\forall \emptyset) t = t'$ and we proceed as above.
Then, the surjective homomorphism $h : \mathbb{M}_{\Sigma^c, \mathtt{E}^c} \to \mathbb{M}_{\Sigma, \mathtt{E}} \restriction_{\Theta}$ is defined by $h([t]_{E^c}) = [\Theta(t)]_E$. The fact that $\mathbb{M}_{\Sigma, \mathtt{E}} \restriction_{\Theta} \models E^c$ follows from the satisfaction condition.
We show now the soundness of the transitivity. Note that the composition of two surjective homomorphisms produces a surjective homomorphism. Applying the satisfaction condition we obtain $\mathbb{M}_{\Sigma, \mathtt{E}} \models \Theta(E^c)$. Because there exists a surjective homomorphism $h : \mathbb{M}_{\Sigma, \mathtt{E}} \to \mathbb{M}_{\Sigma', \mathtt{E}'} \restriction_{\Theta'}$ it follows that $\mathbb{M}_{\Sigma', \mathtt{E}'} \restriction_{\Theta'} \models \Theta(E^c)$. We apply again the satisfaction condition and we obtain $\mathbb{M}_{\Sigma', \mathtt{E}'} \restriction_{\Theta'} \restriction_{\Theta} \models E^c$.
The soundness of the equation/signature addition and sum rules follows in a similar way. □

Theorem 16 *The following deduction rule is sound:*

Argument sort deduction. *If* $f \in \Sigma^c_{s_1 \ldots s_n, s}$, *then*

$$(ASD)\ \frac{E \cup MXEq(\mathfrak{ic}) \cup \{\mathfrak{ic}\} \models s(f(\ldots t_i \ldots)) = \mathtt{true}}{E \cup MXEq(\mathfrak{ic}) \cup \{\mathfrak{ic}\} \models s_i(t_i) = \mathtt{true}}.$$

Proof: The conclusion follows from Theorem 12. □

Proposition 17 *Natural Numbers I.*
The specification:

```
obj NATC is
  sort Nat .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
endo
```

is a constructor-specification for:

```
obj NAT+ is
  pr NATC .
  op _+_ : Nat Nat -> Nat .
  vars M N : Nat .
  eq M + 0 = M .
  eq M + s N = s(M + N) .
endo
```

Proof: The membership extension of $\mathfrak{ic}_{\texttt{NATC},\texttt{NAT+}}$=$\langle Sig(\texttt{NATC}), Eq(\texttt{NATC}), Sig(\texttt{NATC}) \hookrightarrow Sig(\texttt{NAT+})\rangle$ is:

```
obj NAT+MX is
  pr BOOL .
  pr NAT+ .
  op Nat : Nat -> Bool .
  vars M N : Nat .
  eq Nat(0) = true .
  eq Nat(s N) = Nat(N) .
endo
```

The proof of $Eq(\texttt{NAT+MX}) \models (\forall M, N)\texttt{Nat}(M+N)=\texttt{Nat}(M)$ `and` $\texttt{Nat}(N)$ can be checked using OBJ3 system:

```
obj NAT+PROOF is
  pr NAT+MX .
  ops m n : -> Nat .
endo
open .
red Nat(m + 0) == Nat(0) and Nat(m) .
```

```
eq Nat(m + n) = Nat(m) and Nat(n) .  *** ind. hyp.
red  Nat(m + s n) == Nat(m) and Nat(s n) .
close
```

We apply IC_f and obtain that $Eq(\texttt{NAT+}) \models\!\simeq \mathfrak{ic}_{\texttt{NATC,NAT+}}$ that is equivalent with the conclusion of the proposition. □

Proposition 18 *Associativity and commutativity of addition of naturals.*
1. `NAT+` $\models\!\simeq$ `(∀ M N P) (M + N) + P = M + (N + P) .`
2. `NAT+` $\models\!\simeq$ `(∀ M N) M + N = N + M .`

Proof: We denote by e_1 and e_2 the two equations. First we prove that $Eq(\texttt{NAT+}) \cup \{\mathfrak{ic}_{\texttt{NATC,NAT+}}\} \models\!\simeq \{e_1, e_2\}$. A proof of this can be found e.g. in [8]. Next we apply Corollary 10 and Proposition 17. □

Now we can redefine `NAT+` as follows:

```
obj NAT+ is
  pr NATC .
  op _+_ : Nat Nat -> Nat [assoc comm] .
  vars M N : Nat .
  eq M + 0 = M .
  eq M + s N = s(M + N) .
endo
```

Proposition 19 *Natural Numbers II.*
`NATC` *is a constructor specification for:*

```
obj NAT is
  pr NAT+ .
  op _*_ : Nat Nat -> Nat .
  vars M N : Nat .
  eq M * 0 = 0 .
  eq M * s N = (M * N) + M .
endo
```

Proof: We consider the initial constraint

$$\mathfrak{ic}_{\mathtt{NAT+},\mathtt{NAT}} = \langle Sig(\mathtt{NAT+}), Eq(\mathtt{NAT+}), Sig(\mathtt{NAT+}) \hookrightarrow Sig(\mathtt{NAT}) \rangle.$$

The membership extension of $\mathfrak{ic}_{\mathtt{NAT+},\mathtt{NAT}}$ is:

```
obj NATMX is
  pr BOOL .
  pr NAT .
  op Nat : Nat -> Bool .
  vars M N : Nat .
  eq Nat(0) = true .
  eq Nat(s N) = Nat(N) .
  eq Nat(M + N) = Nat(M) and Nat(N) .
endo
```

In order to apply (IC_f), we have to show that:

$$Eq(\mathtt{NATMX}) \models\!\!\sim (\forall M, N)\mathtt{Nat}(M * N) = \mathtt{true\ if\ Nat}(M)\ \mathtt{and\ Nat}(N).$$

Again, this can be easily checked using OBJ3 system:

```
obj NATMX-PROOF is
  pr NATMX .
  ops m n : -> Nat .
endo

open .
red Nat(m * 0) .

eq Nat(m * n) = Nat(m) and Nat(n) .  *** ind. hyp.
red  Nat(m * s n) == Nat(m) and Nat(s n) .
```

The conclusion of the proposition follows now by transitivity. □

The next result exhibits a proof by induction in module `Nat`.

Proposition 20 $Eq(\mathtt{NAT}) \cup \{\mathfrak{ic}_{\mathtt{NAT+},\mathtt{NAT}}\} \models\!\!\sim (\forall \mathtt{M}, \mathtt{N})\ \mathtt{s(M) * N = M * N + N}$.

Proof: We prove first that $Eq(\mathtt{NAT}) \cup \{\mathfrak{ic}_{\mathtt{NATC,NAT}}\} \models\!\!\simeq e$ where e is the equation from the proposition.

```
    obj NAT-PROOF is
      pr NAT .
      ops m n : -> Nat .
    endo

open .
*** s M * N = M * N + N
red s(m) * 0 == (s(m) * 0) + 0 .

eq s(m) * n = (m * n) + n .
red s(m) * s(n) == (m * s(n)) + s(n) .
close
```

The above proof uses the fact that the addition is associative and commutative. Next we apply Corollary 10 and Proposition 19. □

The next example exhibits the use of the sum rule. Consider the following specifications:

```
    obj BITC is
      sort Bits .
      ops 0 1 : -> Bits .
      op _ _ : Bits Bits -> Bits [assoc] .
    endo

    obj NAT2 is
      pr NATC .
      op _%2 : Nat -> Nat .
      op _/2 : Nat -> Nat .
      var N : Nat .
      eq 0 %2 = 0 .
      eq s 0 %2 = s 0 .
      eq s s N %2 = N %2 .
      eq 0 /2 = 0 .
```

```
  eq s 0 /2 = 0 .
  eq s s N /2 = s( N /2) .
endo

obj BITS is
  pr (BITC + NAT2) .
  op #_ : Bits -> Nat .
  op [_] : Nat -> Bits .
  var B : Bits .  var N : Nat .
  eq # 0 = 0 .
  eq # 1 = s 0 .
  eq #(B 0) = s(# B) .
  eq #(B 1) = s(# B) .
  eq [0] = 0 .
  eq [s 0] = 1 .
  eq [s s N] = [s(N /2)][N %2] .
endo
```

`BITC` specifies the finite sequences of bits. `NAT2` specifies the naturals with the operations divide `N` by 2 (`N /2`) and remainder of N by 2 (`N %2`). `BITS` specifies naturals and sequences of bits together with the operation `# B` which returns the length of the sequence `B` and the operation `[N]` which returns the binary representation of `N`.

Proposition 21 `NATC + BITC` *is a constructor specification for* `BITS`.

Proof: We apply IC_f twice and the transitivity and we obtain that `NATC` is a constructor specification for `NAT2`. Applying the sum rule, we get that `NATC + BITC` is a constructor specification for `NAT2 + BITC`. Then we prove that `NAT2 + BITC` is a constructor specification for `BITS`. Finally we apply the transitivity and we obtain the conclusion of the proposition. □

Definition 22 Operations well inductively defined.
Consider (Σ^c, E^c) *a constructor-specification for* $(\Sigma = \Sigma^c \cup \{f\}, E)$ *such that* $f \in \Sigma_{w,s}$. *We say that* f *is* well inductively defined *iff* $\mathbb{M}_{\Sigma^c,E^c} \cong \mathbb{M}_{\Sigma,E}\restriction_{\Sigma^c}$.

Example 23 Adding "confusions" we may destroy well inductively *defined constructions.*
We first define on natural numbers the relation ">" with the usual meaning:

```
obj NAT is
  pr NAT .
  op _>_ : Nat Nat -> Bool .
  eq M > M = false .
  eq 0 > s N = false .
  eq (s M > 0) = true .
  eq (s M > s N) = M > N .
endo
```

NAT *initially satisfies the equation* $(\forall \mathtt{M}, \mathtt{N})(\mathtt{M} + \mathtt{sN}) > \mathtt{0} = \mathtt{true}$. *This can be proved directly by reduction:*

```
open NAT .
ops (m) (n) : -> Nat .

red (m + s n) > 0 .
```

We add to NAT *the following equation:*

```
obj NAT3 is
  us NAT .
  eq s s s 0 = 0 .
endo
```

The algebra $\{0, 1, 2\}$ *with the usual interpretation of the operators (all computations are made modulo 3) is an initial model. The term* $(\mathtt{ss0} + \mathtt{s0}) > \mathtt{0}$ *can be reduced to both* true *and* false *and we obtain from here the unneeded confusion* $\mathtt{true} = \mathtt{false}$. *The reason for that we obtain this contradiction is of inductive nature: initially,* $>$ *is well inductively defined but adding the equation* s s s 0 = 0 *the "well inductively defined" property is destroyed. The confusions may destroy the inductive definitions!*

The example given above shows that the use of surjectivity in the definition for constructors is not always appropriate. In order to have a correct definition of $<$ in NAT3 we have to use the reliable constructors $\{0, 1, 2\}$.

6 Induction in Initial Order Sorted Logic

Let $IOSA$ denote the institution $\mathcal{IT}(OSA)$, where OSA is the institution of order sorted algebra [7]. We consider here only coherent order sorted specifications. Consequently, for any order sorted specification $(S, \leq, \Sigma, E)$, each connected component of $(S, \leq)$ has a maximal sort.

The rules for equation deduction by induction from the initial many sorted logic remains valid for the initial order sorted logic with the mention that only the maximal operations are used in the rule (IND). If $f \in \Sigma^c_{w',s'} \cap \Sigma^c_{w,s}$ and $w' \leq w$ then $s' \leq s$ by the monotonicity condition. It follows that $(\mathbb{M}_{\Sigma^c,\mathbf{E}^c})_{s'} \subseteq (\mathbb{M}_{\Sigma^c,\mathbf{E}^c})_s$ and therefore we take in (IND) only those $f \in \Sigma^c_{w,s}$ for that (w, s) is maximal.

The rules for initial constraints remains true provided we consider the following definition in (IC_f):

Definition 24 *Membership extension of a initial constraint.*
Let $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *be a initial constraint. Then:*

1. $MXSig(\mathfrak{ic}) = \Sigma \cup Sig(\mathtt{BOOL}) \cup \{s : \Theta(ms(s)) \to \mathtt{Bool} \mid s \in sort(\Sigma^c)\}$, *where* $ms(s)$ *denotes the maximal sort of the connected component of* s*;*
2. $MXEq(\mathfrak{ic}) = \{s(\Theta(f)) = \mathtt{true} \mid f \in \Sigma^c_{[],s}\} \bigcup$
   $\{(\forall x_1, \ldots, x_n) s(\Theta(f)(x_1 \ldots x_n)) = \bigwedge_{i=1}^n s_i(x_i) \mid f \in \Sigma^c_{s_1 \ldots s_n, s}\} \bigcup$
   $\{(\forall x) s(x) = \mathtt{true\ if}\ s'(x) \mid s' \leq s\}$

Theorem 25 *Let* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *be a initial constraint. If* $(\Sigma, E) \models \mathfrak{ic}$ *then* $[\![s]\!]_{\mathbb{M}_{\mathtt{MXSig(ic)},\mathtt{MXEq(ic)} \cup \mathtt{E}}}([t]) = true$ *for all* $[t] \in (\mathbb{M}_{\Sigma,\mathbf{E}})_s$, $s \in \mathrm{sort}(\Sigma^c)$.

Proof: Similar to Theorem 12. □

Corollary 26 *Let* $\mathfrak{ic} = \langle \Sigma^c, E^c, \Theta : \Sigma^c \to \Sigma \rangle$ *be a initial constraint. If* $(\Sigma, E) \models\!\!\!\models \mathfrak{ic}$ *then* $\mathbb{M}_{\mathtt{MXSig}(\mathfrak{ic}),\mathtt{MXEq}(\mathfrak{ic})\cup \mathtt{E}} \cong \mathbb{M}_{\Sigma,\mathtt{E}}$.

Theorem 27 *The following deduction rules are sound:*

Constant subsort deduction. *If* $s' < s$ *and* t *is a ground term of sorts* s*, then*

$$(CSD)\ \frac{E \models\!\!\!\models (\forall x : s')e, E \cup MXEq(\mathfrak{ic}) \models\!\!\!\models s'(t) = true}{E \models\!\!\!\models e(x \leftarrow t)}.$$

Subsort case analysis. *If* $s_1, \ldots, s_n < s$*, then*

$$(SCA)\ \frac{\begin{array}{l} E \models\!\!\!\models (\forall x : s_i)e \text{ for } i = \overline{1,n}, \\ E \cup MXEq(\mathfrak{ic}) \models\!\!\!\models (\forall x : s) \bigvee_{i=1}^{n} s_i(x) = \mathtt{true\ if}\ s(x) = \mathtt{true} \end{array}}{E \models\!\!\!\models (\forall x : s)e}.$$

Proof: The conclusion follows from Theorem 25. □

The next specification is useful in what follows:

```
obj ZERO is
  sorts Zero .
  op 0 : -> Zero .
end
```

Proposition 28 *Natural numbers III.*
The specification:

```
obj NATC is
  pr ZERO .
  sorts NzNat Nat .
  subsorts Zero < Nat .
  subsort NzNat < Nat .
  op s_ : Nat -> NzNat .
end
```

is a constructor-specification for

```
obj NAT is
  pr NATC .
  op _+_ : Nat Nat -> Nat .
  vars M N : Nat .
  eq M + 0 = M .
  eq M + s N = s(M + N) .
end
```

Proof: The membership extension associated with the initial constraint $\mathfrak{ic}_{\mathtt{NATC,NAT}}$ is:

```
obj NATMX is
  pr NAT .
  op Zero : Nat -> Bool .
  op NzNat : Nat -> Bool .
  op Nat : Nat -> Bool .
  var N : Nat .
  eq Zero(0) = true .
  eq NzNat(s(N)) = Nat(N) .
  eq Nat(N) = true if NzNat(N) .
  eq Nat(N) = true if Zero(N) .
end
```

Now we prove that $Eq(\mathtt{NATMX}) \cup \{\mathfrak{ic}_{\mathtt{NATC,NAT}}\} \models (\forall \mathtt{M}, \mathtt{N})\mathtt{Nat}(\mathtt{M} + \mathtt{N}) = \mathtt{true}$ `if Nat(M) and Nat(N)`:

```
open .
ops m n : -> Nat .
*** basis
eq Nat(m) = true .
red Nat(m + 0) .
*** induction
eq Nat(s n) = true .   *** condition
eq Nat(n) = true .     *** rule (ASD)
eq Nat(m + n) = true . *** inductive hypothesis
red Nat(m + s n)  .    *** it should be true
close
```

The conclusion of the proposition follows now applying IC_f. □

The next proposition is proved in a similar way.

Proposition 29 *Nonpozitive integers.*
The specification:

```
obj NPINTC is
  pr ZERO .
  sorts NegInt NpInt .
  subsort Zero < NpInt .
  subsort NegInt < NpInt .
  op p_ : NpInt -> NegInt .
end
```

is a constructor specification for

```
obj NPINT is
  pr NPINTC .
  op _+_ : NpInt NpInt -> NpInt .
  vars X Y : NpInt .
  eq X + 0 = X .
  eq X + p Y = p(X + Y) .
end
```

Proposition 30 *Integers.*
The specification:

```
obj INTC is
  pr NATC + NPINTC .
  sort  Int .
  subsort Nat < Int .
  subsort NpInt < Int .
  op s_ : Int -> Int .
  op p_ : Int -> Int .
  vars I J : Int .
  eq s p I = I .
  eq p s I = I .
end
```

is a constructor specification for

```
obj INT is
  pr NAT + NPINT + INTC .
  op _+_ : Int Int -> Int .
  vars M N : Nat . vars X Y : NpInt .
  eq s M + p Y = M + Y .
  eq p X + s N = X + N .
end
```

Proof: The membership extension of the initial constraint $\mathfrak{ic}_{\texttt{INTC,INT}}$ is:

```
obj INTMX is
  pr INT .
  op Zero : Int -> Bool .
  op NzNat : Int -> Bool .
  op Nat : Int -> Bool .
  op NegInt : Int -> Bool .
  op NpInt : Int -> Bool .
  op Int : Int -> Bool .
  vars M N : Nat . vars X Y : NpInt .
  vars I J : Int .
  eq Zero(0) = true .
  eq NzNat(s(I)) = Nat(I) .
  eq NegInt(p(I)) = NpInt(I) .
  eq Int(s(I)) = Int(I) .
  eq Int(p(I)) = Int(I) .
  eq Int(I) = true if Nat(I) .
  eq Int(I) = true if NpInt(I) .
  eq Nat(I) = true if NzNat(I) .
  eq Nat(I) = true if Zero(I) .
  eq NpInt(I) = true if NegInt(I) .
  eq NpInt(I) = true if Zero(I) .
end
```

We prove first the next three lemmas.

Lemma 31 $Eq(\texttt{INTMX}) \cup \{\mathfrak{ic}_{\texttt{INTC,INT}} \models \texttt{(}\forall\texttt{A:Int)Nat(A) or NpInt(A) = true if Int(A) .}$

Proof: By induction on `A`. The basis is straightforward. The inductive basis is checked by cases:

```
obj INT-PROOF1 is
  pr INTMX .
  op a : -> Int .
  eq Int(s a) = true .  *** condition
  eq Int(a) = true .    *** rule (ASD)
  eq Nat(a) or NpInt(a) = true . *** ind. hyp.
endo

*** case 1
open .
eq Nat(a) = true .
red Nat(s a) or NpInt(s a) .
close
*** case 2
eq NpInt(a) = true .
red Nat(p a) or NpInt(p a) .
close
```

In the first case we used only the constructors for the sort `Nat` because we have `Nat(a) = true`. Similarly, in the second case we used only the constructors for the sort `NpInt`. In fact, we used here the deduction rule (CSD). □

Lemma 32 $Eq(\texttt{INTMX}) \cup \{\mathfrak{ic}_{\texttt{INTC,INT}} \models \{\texttt{(}\forall\texttt{ M:Nat,Y:NpInt) M + p(Y) = p(M + Y), (}\forall\texttt{ M:Nat,Y:NpInt) s(M) + Y = s(M + Y)}\}$.

Proof: By induction on A for the first equation and by induction on B for the second:

```
obj INT-PROOF2 is
  pr INTMX .
```

```
      op m : -> Nat .
      op y : -> NpInt .
    end
open .
*** basis
red (0 + p y) == p(0 + y) .
red (s m + 0) == s(m + 0) .
*** induction
eq (m + p y) = p(m + y) .  *** inductive hypothesis
eq (s m + y) = s(m + y) .  *** inductive hypothesis
red (s m + p y) == p(s m + y) .
red (s m + p y) == s(m + p y) .
close
```

□

The next lemma is proved in a similar way:

Lemma 33 $Eq(\texttt{INTMX}) \cup \{\mathfrak{ic}_{\texttt{INTC,INT}} \models\!\simeq \{(\forall\ \texttt{X:NpInt, N:Nat})\ \texttt{p(X) + N = p(X + N)}, (\forall\ \texttt{X:NpInt, N:Nat})\ \texttt{X + s(N) = s(X + N)}\}$.

Now are ready to prove the proposition showing that:

$Eq(\texttt{INTMX}) \cup \{\mathfrak{ic}_{\texttt{INTC,INT}} \models\!\simeq (\forall \texttt{A,B})\texttt{Int(A+B)=true if Int(A) and Int(B)}$.

We proceed by induction on B.

```
    obj INT-PROOF3 is
      pr INTMX .
      ops a b : -> Int .
      eq Int(a) = true .
      eq Nat(a) or NpInt(a) = true . *** first lemma
      vars M N : Nat . vars X Y : NpInt .
      eq M + p Y = p(M + Y) .   *** second lemma
      eq X + s N = s(X + N) .   *** third lemma
    endo

*** basis: case 1: eq Nat(a) = true .
```

```
open .
op a : -> Nat .
red Int(a + 0) .
close
*** basis: case 2: eq NpInt(a) = true .
open .
op a : -> NpInt .
red Int(a + 0) .
close
*** induction
    obj INT-PROOF4 is
      pr INT-PROOF3 .
      eq Int(s b) = true .           *** condition
      eq Int(p b) = true .            *** condition
      eq Int(b) = true .              *** rule (ASD)
      eq Int(a + b) = true .          *** ind. hyp.
      eq Nat(b) or NpInt(b) = true . *** first lemma
    endo

*** induction: case 1
***    eq Nat(a) = true .
***    eq Nat(b) = true .
open .
op a : -> Nat .
op b : -> Nat .
red Int(a + s b) .
close
*** induction: case 2
***    eq Nat(a) = true .
***    eq NpInt(b) = true .
open .
op a : -> Nat .
op b : -> NpInt .
red Int(a + p b) .
close
```

```
*** induction: case 3
***    eq NpInt(a) = true .
***    eq Nat(b) = true .
open .
op a : -> NpInt .
op b : -> Nat .
red Int(a + s b) .
close
*** induction: case 4
***    eq NpInt(a) = true .
***    eq NpInt(b) = true .
open .
op a : -> NpInt .
op b : -> NpInt .
red Int(a + p b) .
close
```

The conclusion of the proposition follows now applying (SCA). □

Proposition 34 $\texttt{INT} \models \{(\forall \texttt{I})\texttt{I} + \texttt{0} = \texttt{I}, (\forall \texttt{I}, \texttt{J})\texttt{I} + \texttt{s}(\texttt{J}) = \texttt{s}(\texttt{I} + \texttt{J}), (\forall \texttt{I}, \texttt{J})\texttt{I} + \texttt{p}(\texttt{J}) = \texttt{p}(\texttt{I} + \texttt{J})\}$.

Proposition 35 *The specification:*

```
obj INTC2 is
  pr Zero .
  sort  Int .
  subsort Zero < Int .
  op s_ : Int -> Int .
  op p_ : Int -> Int .
  vars I J : Int .
  eq s p I = I .
  eq p s I = I .
end
```

is a constructor specification for INT.

7 Induction in Initial Membership Equational Logic

Let IMA denote the institution $\mathcal{IT}(MA)$, where MA is the institution of membership algebra [11].

Because the membership equational logic (MEL) is fully implemented in Maude [3], we use in this section the Maude syntax. The basic predicates in MEL are equations and membership assertions $t : s$ stating that the term t belongs to the sort s. The rule (IND) is extended naturally for membership assertions. Moreover the membership extension of a initial constraint is replaced by a MEL specification. E.g., the module `NAT+MX` from 17 is replaced by the following Maude functional specification:

```
fmod NAT is
  sorts Nat Nat? .
  subsort Nat < Nat? .
  op 0 : -> Nat [ctor] .
  op s_ : Nat -> Nat [ctor] .
  op _+_ : Nat? Nat? -> Nat? .
  vars M N : Nat .
  eq M + 0 = M .
  eq M + s N = s(M + N) .
endfm
```

The verification of the assumption of the rule IC_f is straightforward:

```
fmod NAT-PROOF is
  pr NAT .
  ops m n : -> Nat .
  mb (m + n) : Nat .       *** inductive hypothesis
endfm
red (m + 0) : Nat .
red (m + s n) : Nat .
```

This can be showed automatically using the inductive theorem prover ITP [2]:

```
(goal
    fmod NAT is
      including BOOL .
      sorts Nat Nat? .
      subsort Nat < Nat? .
      op 0 : -> Nat [ctor] .
      op s_ : Nat -> Nat [ctor] .
      op _+_ : Nat? Nat? -> Nat? .
      vars M N : Nat .
      eq (M + 0) = (M) .
      eq (M + s N) = (s(M + N)) .
    endfm
|- {M ; N}((M + N) : Nat ) .
)

(ind (1) on N .)
(simp (1 . 2) . )
(simp (1 . 1) . )
```

After we proved that 0 and `s_` are constructors indeed, we can use ITP for proving properties like associativity:

```
(goal
    fmod NAT is
      including BOOL .
      sort Nat .
      op 0 : -> Nat [ctor] .
      op s_ : Nat -> Nat [ctor] .
      op _+_ : Nat Nat -> Nat .
      vars M N P : Nat .
      eq (M + 0) = (M) .
      eq (M + s N) = (s(M + N)) .
    endfm
|- {M ; N ; P}((M + (N + P)) = ((M + N) + P)) .
)
```

```
(ind (1) on P .)
(simp (1 . 2). )
(simp (1 . 1). )
```

ITP implements partially the deduction rules we presented in this paper. E.g., ITP fails to prove the following goal:

```
(goal
    fmod INT is
      sorts Zero NzNat Nat NegInt NpInt Int .
      subsort Zero < Nat .
      subsort Nat < Int .
      subsort NzNat < Nat .
      subsort Zero < NpInt .
      subsort NpInt < Int .
      subsort NegInt < NpInt .
      op 0 : -> Zero [ctor] .
      op s_ : Nat -> NzNat  [ctor] .
      op p_ : NpInt -> NegInt  [ctor] .
      op s_ : Int -> Int [ctor]  .
      op p_ : Int -> Int  [ctor] .
      op _+_ : Zero Zero -> Zero .
      op _+_ : Nat Nat -> Nat .
      op _+_ : NpInt NpInt -> NpInt .
      op _+_ : Int Int -> Int .
      vars I J : Int .
      eq (s p I) = (I) .
      eq (p s I) = (I) .
      vars M N : Nat .
      eq (M + 0) = (M) .
      eq (M + s N) = (s(M + N)) .
      vars X Y : NpInt .
      eq (X + 0) = (X) .
      eq (X + p Y) = (p(X + Y)) .
      eq (s M + p Y) = (M + Y) .
      eq (p X + s N) = (X + N) .
```

```
    endfm
|- {I}((I + 0) = (I)) .
)

(ind (1) on I . )
(simp (1 . 2). )
(simp (1 . 1). )
```

One reason for that ITP fails is that it does not implement subsort case analysis rule.

8 Conclusions

The paper studies the initial equational logic, that is the logic of the statements that are true in initial algebra. This logic is formalized as an institution and a deductive calculus for inductive proofs is proposed. The future research will focus on improving the deductive calculus by a careful analysis of the case of structured specifications. Another future research direction is the extension of the calculus from initial algebras to free algebras.

References

[1] A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. Technical report, SRI International, 1988.

[2] M. Clavel, F. Durán, S. Eker, and J. Meseguer. Building equational proving tools by reflection in rewriting logic. In *Cafe: An Industrial-Strength Algebraic Formal Method*. Elsevier, 2000.

[3] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 2001. To appear.

[4] J. Goguen. *Theorem Proving and Algebra*. MIT Press. to appear.

[5] J. Goguen and R. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[6] J. Goguen and J. Meseguer. Completeness of many-sorted equational logic. *Houston J. Math*, 11(3):307–334, 1985.

[7] J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105(2):217–273, 1992.

[8] Joseph Goguen, Timothy Winkler, José Mesegure, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. Technical report, SRI International, Computer Science Laboratory, 1993.

[9] D. Lucanu. Initial many sorted algebra. In D. Petcu, V. Negru, D. Zaharie, and T. Jebelean, editors, *Symbolic and Numeric Algorithms for Scientific Computing*, pages 166–180, Timişoara, Romania, 2002. Mirton.

[10] J. Meseguer. General logics. In H.-D. Ebbinghaus et al., editor, *Logic Colloquium '87*, pages 275–329, North Holland, Amsterdam, 1989.

[11] J. Meseguer. Membership algebra as a logical framework for equational specification. In *WADT'97*, volume 1376 of *LNCS*, pages 18–61, Tarquinia, Italy, June 1997. Invited Talk.

A Institutions

We recall from [5] the definitions for institutions and constraints.

Definition 36 Institutions.
An institution *is a quadruple* $\mathcal{I} = (\underline{\mathbf{Sign}}, \mathbf{sen}, \underline{\mathbf{Mod}}, \models)$ *where* $\underline{\mathbf{Sign}}$

is a category whose objects are called signatures, **sen** *is a functor* $\mathbf{sen} : \underline{\mathbf{Sign}} \to \underline{\mathbf{Set}}$ *which associates with each signature a set whose elements are called* sentences, $\underline{\mathbf{Mod}} : \underline{\mathbf{Sign}}^{op} \to \underline{\mathbf{Cat}}$ *is a functor which associates with each signature a category whose objects are called* models, *and* $\models$ *is a function which associates with each signature* Σ *a binary relation* $\models_\Sigma \subseteq \mathcal{P}(|\underline{\mathbf{Mod}}|\,(\Sigma)) \times \mathbf{sen}(\Sigma)$, *called* satisfaction relation, *such that for each morphism* $\Phi : \Sigma \to \Sigma'$ *the* satisfaction condition

$$\underline{\mathbf{Mod}}(\Phi)(M') \models_\Sigma e \iff M' \models_{\Sigma'} \Phi(e).$$

holds for each model $M' \in \underline{\mathbf{Mod}}(\Sigma')$ *and each sentence* $e \in \mathbf{sen}(\Sigma)$.

Notation 37 *If* $\Phi : \Sigma \to \Sigma'$ *is a signature morphism, then the* Σ*-model* $\underline{\mathbf{Mod}}(\Phi)(M')$ *is also denoted by* $M'|_\Phi$ *and we call it* the Φ-reduct of M'.

Definition 38 *1.* $M \models_\Sigma E$ *if* $M \models e$ *for each* $e \in E$.

2. $\underline{\mathbf{Mod}}(\Sigma, E) = \{M \mid M \models_\Sigma E\}$.

3. $E \models_\Sigma e$ *if* $M \models e$ *for each model* $M \in \underline{\mathbf{Mod}}(\Sigma, E)$. *We say that* e *is a* semantical consequence *of* E.

Definition 39 *Let* Σ *be a signature. Then, a* Σ-constraint *is a pair*

$$\langle F : (\Sigma'', E'') \to (\Sigma', E'), \Theta : \Sigma' \to \Sigma \rangle$$

consisting of a theory morphism and a signature morphism. A Σ*-model* M *satisfies the* Σ*-constraint* $c = \langle F : (\Sigma'', E'') \to (\Sigma', E'), \Theta : \Sigma' \to \Sigma \rangle$ *iff* $M|_\Theta$ *satisfies* E' *and is* F*-free, i.e.,* $M|_\Theta$ *has a free extension along* F *such that* $(1_{M|_\Theta})^\# : (M|_\Theta)^\$ \to M|_\Theta$ *is an isomorphism. We write* $M \models_\Sigma c$.

Definition 40 *Let* $\Phi : \Sigma \to \Sigma'$ *be a signature morphism and let* $c = \langle F, \Theta \rangle$ *be a constraint. Then the* translation of c by Φ *is the* Σ'*-constraint* $\langle F, \Theta; \Phi \rangle$*; we write* $\Phi(c)$.

Definition 41 *Given an arbitrary institution* $\mathcal{I}$, *construct the institution* $\mathcal{C}(\mathcal{I})$ *as follows:*

1. *the category of signatures of* $\mathcal{C}(\mathcal{I})$ *is the category* $\underline{\mathbf{Sign}}$ *of signatures of* $\mathcal{I}$*;*
2. *if* Σ *is* $\mathcal{I}$*-signature, then* $\mathbf{sen}_{\mathcal{C}(\mathcal{I})}(\Sigma)$ *is the disjoint union of the collection* $\mathbf{sen}_{\mathcal{I}}(\Sigma)$ *of all* Σ*-sentences from* $\mathcal{I}$ *with the collection of all* Σ*-constraints;*
3. $\underline{\mathbf{Mod}}(\Sigma)$ *is the same for* $\mathcal{C}(\mathcal{I})$ *as for* $\mathcal{I}$*;*
4. *the satisfaction for* $\mathcal{C}(\mathcal{I})$ *is as in* $\mathcal{I}$ *for* Σ*-sentences in* $\mathcal{I}$*, and is as in Definition 39 for* Σ*-constraints.*

Theorem 42 *If* $\mathcal{I}$ *is an institution then* $\mathcal{C}(\mathcal{I})$ *is an institution.*

Definition 43 *Let* $\mathcal{I}$ *be an institution, let* $\mathcal{M}$ *denote a class of model morphisms for each signature* Σ *of* $\mathcal{I}$*, let* $c = \langle F : (\Sigma'', E'') \to (\Sigma', E'), \Theta : \Sigma' \to \Sigma \rangle$ *be a constraint from* $\mathcal{I}$*, and let* M *be a* Σ*-model from* $\mathcal{I}$*. Then,* M $\mathcal{M}$-satisfies c *iff* $M{\restriction_\Theta}$ *satisfies* E' *and has a free extension along* F *such that* $(1_{M{\restriction_\Theta}})^{\#}$ *lies in* $\mathcal{M}$*.*

Theorem 44 *Modifying the construction of* $\mathcal{C}(\mathcal{I})$ *to use* $\mathcal{M}$*-satisfaction gives an institution.*