PAPER

Primitive Inductive Theorems Bridge Implicit Induction Methods and Inductive Theorems in Higher-Order Rewriting

Keiichirou KUSAKARI[†a)], Masahiko SAKAI[†], *and* Toshiki SAKABE[†], *Members*

SUMMARY Automated reasoning of inductive theorems is considered important in program verification. To verify inductive theorems automatically, several implicit induction methods like the inductionless induction and the rewriting induction methods have been proposed. In studying inductive theorems on higher-order rewritings, we found that the class of the theorems shown by known implicit induction methods does not coincide with that of inductive theorems, and the gap between them is a barrier in developing mechanized methods for disproving inductive theorems. This paper fills this gap by introducing the notion of primitive inductive theorems, and clarifying the relation between inductive theorems and primitive inductive theorems. Based on this relation, we achieve mechanized methods for proving and disproving inductive theorems.

key words: algebraic specification, simply-typed term rewriting system, primitive inductive theorem, inductive theorem, implicit induction method

1. Introduction

Term rewriting systems (TRSs) provide an operational model of functional programming languages. TRSs also give a theoretical foundation of algebraic specification languages [8], [9]. In algebraic specification, many interesting properties of programs can be formally dealt with as inductive theorems, characterized by the initial algebra semantics [10], [12]–[14], [24]. The concept of inductive theorems is extremely important in practical applications. In fact, most data structures used in functional programming are inductive structures such as list and tree structures. As a result, most properties that a program must guarantee are formalized as inductive theorems.

In order to verify programs, automated reasoning for proving and disproving inductive theorems is very important. Hence many inductive reasoning methods in TRSs, called implicit induction methods (the inductionless induction and the rewriting induction methods), have been studied [6], [7], [11], [16], [21]–[23].

Higher-order functions, which can treat functions as values, provide a facility of high-level abstraction and more expressive power. Unfortunately, TRSs cannot express higher-order functions directly. For this reason, the first-author proposed simply-typed term rewriting systems (STRSs) [17]. In STRSs, the *Map*-function, which is one of the most typical higher-order function, is represented as follows:

$$\begin{cases} Map(f, Nil) & \to & Nil \\ Map(f, x :: xs) & \to & f(x) :: Map(f, xs) \end{cases}$$

In this paper, we study inductive theorems on STRSs for automated reasoning and these characterization. We found that the class of the theorems shown by known implicit induction methods does not coincide with that of inductive theorems, and the gap between them is a barrier in developing mechanized methods for disproving inductive theorems. We fill this gap by introducing the notion of primitive inductive theorems, and clarifying the relation between inductive theorems and primitive inductive theorems. Based on this relation, we achieve mechanized methods for proving and disproving inductive theorems.

The main contributions of the paper are:

(1) We give a notion of primitive inductive theorem (Definition 4.1), and characterize inductive theorems by primitive ones (Theorem 4.6).
(2) We show that the existing implicit induction methods (the inductionless induction [23] and the rewriting induction methods [16]) can be naturally extended to STRSs for proving/disproving primitive inductive theorems (Theorem 5.2, 5.7 and 6.2).
(3) We show that the implicit induction methods are also applicable for proving inductive theorems (Theorem 5.2 and 5.7), because every primitive inductive theorem is an inductive theorem (Theorem 4.2).
(4) For disproving inductive theorems, implicit induction methods do not work well because some inductive theorems are not a primitive inductive theorem. To overcome the difficulty, we present a sufficient condition for inductive theorems to coincide with primitive inductive theorems (Theorem 4.17). Under this sufficient condition, the implicit induction method is applicable for disproving inductive theorems (Theorem 6.3).
(5) To give a justification of our definition of inductive theorems, we design a higher-order equational logic, and show that the notion of inductive theorems is characterized by the initial extensional semantics (Theorem 7.10).

The remainder of this paper is organized as follows. The next section gives the preliminaries needed later on. In Sect. 3, we give the definition of inductive theorems. In Sect. 4, we give the notion of primitive inductive theorems, and characterize inductive theorems by primitive inductive theorems. We also present a sufficient condition for inductive theorems to coincide with primitive inductive theorems.

Manuscript received March 28, 2005.

[†]The authors are with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8603 Japan.
a) E-mail: kusakari@is.nagoya-u.ac.jp

DOI: 10.1093/ietisy/e88–d.12.2715

In Sect. 5, we study automated reasoning for proving inductive theorems. In Sect. 6, we also study automated reasoning for disproving inductive theorems. In Sect. 7, we study higher-order equational logic, and show that the notion of inductive theorems is characterized by the initial extensional semantics.

2. Preliminaries

We assume that the reader is familiar with notions of term rewriting systems [4].

2.1 Abstract Reduction System

An *abstract reduction system* (ARS) is a pair $\langle A, \rightarrow \rangle$ where A is a set and $\rightarrow$ is a binary relation on A. The transitive-reflexive closure of a binary relation $\rightarrow$ is denoted by $\overset{*}{\rightarrow}$, the transitive closure is denoted by $\overset{+}{\rightarrow}$, and the transitive-reflexive-symmetric closure is denoted by $\overset{*}{\leftrightarrow}$.

Let $R = \langle A, \rightarrow \rangle$ be an ARS. An element $a \in A$ is said to be a *normal form* if there exists no b such that $a \rightarrow b$. We denote the set of all normal forms by $NF(R)$. An ARS R is said to be *weakly normalizing*, denoted by $WN(R)$, if $\forall a \in A.\ \exists b \in NF(R).\ a \overset{*}{\rightarrow} b$; to be *strongly normalizing* (*terminating*), denoted by $SN(R)$, if there exists no infinite sequence $a_0 \rightarrow a_1 \rightarrow \cdots$; to be *confluent*, denoted by $CR(R)$, if $a_1 \overset{*}{\leftarrow} a \overset{*}{\rightarrow} a_2$ implies $\exists b.\ a_1 \overset{*}{\rightarrow} b \overset{*}{\leftarrow} a_2$ for all $a, a_1, a_2 \in A$. For ARSs R_1 and R_2, R_2 *retrogresses* to R_1, denoted by $RET(R_2, R_1)$, if $a_1 \rightarrow_2 a_2$ implies $a_1 \rightarrow_1 b_1 \rightarrow_2^* b_2 \leftarrow_2^* a_2$ for some b_1 and b_2. Note that $SN(R)$ implies $WN(R)$, and $CR(R)$ implies that the normal form of an element is unique if it exists.

2.2 Untyped Term Rewriting System

Untyped term rewriting systems (UTRSs) introduced in [17][†], which can express higher-order functions directly, are term rewriting systems without arity-typed constraints. In this subsection, we introduce some notions of UTRSs needed later on.

Let Σ be a *signature*, that is a finite set of function symbols, which are denoted by $F, G, \ldots$. Let $\mathcal{V}$ be an enumerable set of variables with $\Sigma \cap \mathcal{V} = \emptyset$. Variables are denoted by $x, y, z, f, \ldots$. *Atom* is a function or variable symbol denoted by $a, a', \ldots$. The set $T(\Sigma, \mathcal{V})$ of (untyped) terms constructed from Σ and $\mathcal{V}$ is the smallest set such that $a(t_1, \ldots, t_n) \in T(\Sigma, \mathcal{V})$ whenever $a \in \Sigma \cup \mathcal{V}$ and $t_1, \ldots, t_n \in T(\Sigma, \mathcal{V})$. If $n = 0$, we write a instead of $a()$. The identity of terms is denoted by $\equiv$. We often write $s(t_1, \ldots, t_m)$ for $a(s_1, \ldots, s_n, t_1, \ldots, t_m)$, where $s \equiv a(s_1, \ldots, s_n)$. We define the *root symbol* of a term $a(t_1, \ldots, t_n)$ by $root(a(t_1, \ldots, t_n)) = a$. $Var(t)$ is the set of variables in t. A term is said to be *closed* if no variable occurs in the term. The set of closed terms is denoted by $T(\Sigma)$. The *size* $|t|$ of t is the number of function symbols and variables in t.

A *substitution* θ is a mapping from variables to terms. We may write θ as $\{x_1 := \theta(x_1), \ldots, x_n := \theta(x_n)\}$ if $\{x_1, \ldots, x_n\} = \{x \in \mathcal{V} \mid \theta(x) \not\equiv x\}$. Each substitution θ is naturally extended over terms, denoted by $\hat{\theta}$, as $\hat{\theta}(F(t_1, \ldots, t_n)) = F(\hat{\theta}(t_1), \ldots, \hat{\theta}(t_n))$ if $F \in \Sigma$; $\hat{\theta}(z(t_1, \ldots, t_n)) = a'(t'_1, \ldots, t'_m, \hat{\theta}(t_1), \ldots, \hat{\theta}(t_n))$ if $z \in \mathcal{V}$ with $\theta(z) = a'(t'_1, \ldots, t'_m)$. For simplicity, we identify θ and $\hat{\theta}$. We write $t\theta$ instead of $\theta(t)$. A *context* is a term which has a special symbol $\Box$, called a hole, at a leaf position. We can also inductively define context as follows: $\Box$ is a context; $a(\ldots, t_{i-1}, C_i[\,], t_{i+1}, \ldots)$ is a context if a is an atom, $C_i[\,]$ is a context and $t_j \in T(\Sigma, \mathcal{V})$ for any $j (\neq i)$. A *suffix context* is a term which has the symbol $\Box$ at the root position. We can also inductively define suffix context as follows: $\Box$ is a suffix context; $(S'[\,])(u)$ is a suffix context if $S'[\,]$ is a suffix context and $u \in T(\Sigma, \mathcal{V})$. For a context $C[\,]$ (a suffix context $S[\,]$), $C[t]$ ($S[t]$) denotes the result of placing t in the hole of $C[\,]$ ($S[\,]$). A term t' is said to be a *subterm* of a term t if there exists a context $C[\,]$ such that $t \equiv C[t']$. We denote by $Sub(t)$ all subterms of t, and define $s \rhd_{sub} t$ by $t \in Sub(s)$ and $s \not\equiv t$.

Example 2.1 For the term $t \equiv f(A, x)$ and the substitution $\theta = \{f := G(0), x := B\}$, we have $t\theta \equiv G(0, A, B)$. We can see that $F(0, \Box)$ and $F(\Box, xs)$ are contexts, $\Box(0)$ and $\Box(0, Nil)$ are suffix contexts, and $\Box$ is both a context and a suffix context. We can also see that $C[t] \equiv a(t, t')$ for $C[\,] \equiv a(\Box, t')$, and $S[a(t)] \equiv a(t, t')$ for $S[\,] \equiv \Box(t')$.

An equivalence relation $\simeq$ on terms is called a *congruence relation* if it is closed under contexts and suffix contexts, that is, $s \simeq t \Rightarrow C[s] \simeq C[t] \wedge S[s] \simeq S[t]$ for any context $C[\,]$ and suffix context $S[\,]$.

A *rewrite rule* is a pair (l, r) of terms such that $root(l) \notin \mathcal{V}$ and $Var(l) \supseteq Var(r)$. We write $l \rightarrow r$ for (l, r). An *untyped term rewriting system* (UTRS) is a set of rewrite rules. The *reduction relation* $\underset{R}{\rightarrow}$ of R is defined by $s \underset{R}{\rightarrow} t$ iff $s \equiv C[S[l\theta]]$ and $t \equiv C[S[r\theta]]$ for some $l \rightarrow r \in R$, $C[\,]$, $S[\,]$ and θ. We often omit the subscript R whenever no confusion arises.

Example 2.2 The Map-function is represented by the following UTRS R:

$$\begin{cases} Map(f, Nil) & \rightarrow & Nil \\ Map(f, x :: xs) & \rightarrow & f(x) :: Map(f, xs) \end{cases}$$

Note that we use the standard representation for list structures by symbols $::$ and Nil. We often write lists in infix form. For example, $::(x, ::(y, Nil))$ is written as $x :: y :: Nil$. Then we have the following reduction relation sequence.

$$\begin{aligned} & Map(F,\ F(0) :: 0 :: Nil) \\ \underset{R}{\rightarrow}\ & F(F(0)) :: Map(F,\ 0 :: Nil) \\ \underset{R}{\rightarrow}\ & F(F(0)) :: F(0) :: Map(F,\ Nil) \\ \underset{R}{\rightarrow}\ & F(F(0)) :: F(0) :: Nil \end{aligned}$$

[†]In [17], UTRSs were called term rewriting systems with higher-order variables (TRS-HVs). Since there exists no "higher-order variable" in untyped systems, we use UTRS in the paper.

2.3 Simply-Typed Term Rewriting System

A simply-typed version of a UTRS is called a simply-typed term rewriting system (STRS) [17].

A set of *basic types* (*sorts*) is denoted by $\mathcal{B}$. The set $\mathcal{T}$ of *simple types* is generated from $\mathcal{B}$ by the constructor $\to$, that is, $\mathcal{T} ::= \mathcal{B} \mid (\mathcal{T} \to \mathcal{T})$. A *type attachment* τ is a function from $\Sigma \cup \mathcal{V}$ to $\mathcal{T}$. A term $a(t_1, \ldots, t_n)$ has type β if $\tau(a) = (\alpha_1 \to (\cdots \to (\alpha_n \to \beta)\cdots))$ and each t_i has type α_i. A term $t \in T(\Sigma, \mathcal{V})$ is said to be *simply-typed* if it has a simple-type. We denote the set of all simply-typed terms by $T_\tau(\Sigma, \mathcal{V})$. We define the *order* $ord(\alpha)$ of a type α as follows: $ord(\alpha) = 1$ if $\alpha \in \mathcal{B}$, and $ord(\alpha) = \max(1 + \alpha_1, \alpha_2)$ if $\alpha = \alpha_1 \to \alpha_2$. A variable is said to be a higher-order variable if it has a simple type α with $ord(\alpha) > 1$. We use $\mathcal{V}_h$ to stand for the set of higher-order variables. A term t is said to be *ground*† if t is closed and of basic types. We denote all closed simply-typed terms by $T_\tau(\Sigma)$, all closed terms having a simple type α by $T_\alpha(\Sigma)$, all basic-typed terms by $T_{\mathcal{B}}(\Sigma, \mathcal{V})$, and all ground terms by $T_{\mathcal{B}}(\Sigma)$. We also denote the set of all basic-typed subterms of t by $Sub_{\mathcal{B}}(t)$.

To keep the type consistency, we assume that $\tau(x) = \tau(\theta(x))$ for all $x \in \mathcal{V}$ and substitutions θ. We also prepare the hole $\Box_\alpha$ with a simple type α, and for each context $C[\,]$ (suffix context $S[\,]$) with a hole $\Box_\alpha$ we assume that $\tau(t) = \alpha$ whenever we denote $C[t]$ ($S[t]$).

A *simply-typed rewrite rule* is a rewrite rule $l \to r$ such that l and r have the same type. A *simply-typed term rewriting system* (STRS) is a set of simply-typed rewrite rules.

Example 2.3 The UTRS representation of the *Map*-function in Example 2.2 is regarded as an STRS with the type attachment $\tau(Nil) = L$, $\tau(::) = N \to L \to L$ and $\tau(Map) = (N \to N) \to L \to L$.

For an STRS R, $GNF(R)$, $GWN(R)$, $GSN(R)$ and $GCR(R)$ are defined as $NF(R')$, $WN(R')$, $SN(R')$ and $CR(R')$ in ARS $R' = \langle T_{\mathcal{B}}(\Sigma), \underset{R}{\to} \rangle$, respectively. For STRSs R_1 and R_2, $GRET(R_2, R_1)$ is defined as that of ARSs $\langle T_{\mathcal{B}}(\Sigma), \underset{R_1}{\to} \rangle$ and $\langle T_{\mathcal{B}}(\Sigma), \underset{R_2}{\to} \rangle$.

2.4 First-Order Term Rewriting System and Equational Logic

The first-order term rewriting system (TRS) is the usual term rewriting system. In this subsection, we introduce some notions needed later on. All results stated in this subsection can be found in [4].

We suppose that each symbol $F \in \Sigma$ is associated with a natural number n, denoted by $ar(F) = n$. We also suppose that $ar(x) = 0$ for any variable x. A term $a(t_1, \ldots, t_n) \in T(\Sigma, \mathcal{V})$ is said to be a *first-order term* if $ar(a) = n$ and each t_i is also first-order term. We denote the set of first-order terms by $T_{ar}(\Sigma, \mathcal{V})$, and the set of closed first-order terms by $T_{ar}(\Sigma)$. In the first order setting, closeness coincides with groundness. Since $\Box$ is the only suffix context, an equivalence relation on first-order terms is a congruence relation whenever it is closed under contexts.

(Assumption)
$$\frac{}{s = t} \quad \text{if } s = t \in E$$
(Reflexivity)
$$\frac{}{t = t}$$
(Symmetry)
$$\frac{t = s}{s = t}$$
(Transitivity)
$$\frac{s = u \quad u = t}{s = t}$$
(Substitutivity)
$$\frac{s = t}{s\theta = t\theta}$$
(Monotonicity)
$$\frac{s_1 = t_1 \quad \cdots \quad s_n = t_n}{F(s_1, \ldots, s_n) = F(t_1, \ldots, t_n)} \quad \text{if } ar(F) = n$$
(Functionality)
$$\frac{\forall u \in T_{ar}(\Sigma).\ s\{z := u\} = t\{z := u\}}{s = t}$$

Fig. 1 Inference rules for the first-order equational logic.

A *first-order equation* is a pair (s, t) of first-order terms, written as $s = t$. We define $Var(s = t)$ by $Var(s) \cup Var(t)$. Let E be a set of first-order equations. We denote by $\overset{*}{\underset{E}{\leftrightarrow}}$ the congruence relation generated from E. A first-order equation $s = t$ is said to be a *theorem* in E, denoted by $E \vdash s = t$, if it is deducible by inference rules in Fig. 1 except for (Functionality)-rule. A first-order equation $s = t$ is said to be an *inductive theorem* in E, denoted by $E \vdash_{ind} s = t$, if it is deducible by inference rules in Fig. 1. Note that (Functionality)-rule is a kind of meta-rule, and proof trees may be infinitely branching.

A Σ-*algebra* $\mathbb{A}$ is a pair $\langle A, \Sigma^{\mathbb{A}} \rangle$, where A is a carrier, $\Sigma^{\mathbb{A}}$ is a mapping which maps each $F \in \Sigma$ of arity n to a n-ary function $F^{\mathbb{A}}$ on A. The *term algebra* $\mathbb{T}_{ar}(\Sigma, \mathcal{V})$ is the Σ-algebra $\langle T_{ar}(\Sigma, \mathcal{V}), \Sigma^{\mathbb{T}_{ar}(\Sigma,\mathcal{V})} \rangle$, where each $F^{\mathbb{T}_{ar}(\Sigma,\mathcal{V})}$ is defined as $F^{\mathbb{T}_{ar}(\Sigma,\mathcal{V})}(t_1, \ldots, t_n) \overset{\text{def}}{=} F(t_1, \ldots, t_n)$. In the case of $\mathcal{V} = \emptyset$, the term algebra is said to be the *ground term algebra*, denoted by $\mathbb{T}_{ar}(\Sigma)$.

Let $\langle A, \Sigma^{\mathbb{A}} \rangle$ be a Σ-algebra. An equivalence relation $\sim$ on A is said to be a *congruence relation* if it is monotonic, that is, $\bigwedge_{i=1}^{n} a_i \sim a'_i \Rightarrow F^{\mathbb{A}}(a_1, \ldots, a_n) \sim F^{\mathbb{A}}(a'_1, \ldots, a'_n)$ for all $F \in \Sigma$ with $ar(F) = n$. Suppose that $\langle A, \Sigma^{\mathbb{A}} \rangle$ is a

†It is a natural extension of groundness of first-order framework, because all first-order closed terms are basic. It is also useful for our purpose, because our inductive reasoning methods on STRSs are based on properties of closed basic-typed terms (cf. Theorem 5.2, 5.7 and 6.3).

Σ-algebra and $\sim$ is a congruence relation on A. We define $[a]_\sim = \{a' \in A \mid a' \sim a\}$ and $A/\sim = \{[a]_\sim \mid a \in A\}$. We also define the *quotient algebra* $\mathbb{A}/\sim = \langle A/\sim, \Sigma^{\mathbb{A}/\sim}\rangle$ of $\mathbb{A}$ modulo $\sim$, where each $F^{\mathbb{A}/\sim}$ is defined as $F^{\mathbb{A}/\sim}([a_1]_\sim, \ldots, [a_n]_\sim) = [F^{\mathbb{A}}(a_1, \ldots, a_n)]_\sim$. It is known that the quotient algebra $\mathbb{A}/\sim$ is well-defined as a Σ-algebra. Clearly, $\overset{*}{\underset{E}{\leftrightarrow}}$ is a congruence relation on $T_{ar}(\Sigma, \mathcal{V})$, and thus $\mathbb{T}_{ar}(\Sigma, \mathcal{V})/\overset{*}{\underset{E}{\leftrightarrow}}$ is a Σ-algebra, called a quotient term algebra. $\mathbb{T}_{ar}(\Sigma)/\overset{*}{\underset{E}{\leftrightarrow}}$ is also a Σ-algebra, called a quotient ground term algebra.

Let $\mathbb{A} = \langle A, \Sigma^{\mathbb{A}}\rangle$ be a Σ-algebra. An *assignment* into $\mathbb{A}$ is a mapping σ from $\mathcal{V}$ to A. The *interpretation* of a first-order term t by an assignment σ into $\mathbb{A}$, denoted by $[\![t]\!]_\sigma$, is defined as $[\![x]\!]_\sigma = \sigma(x)$ for any $x \in \mathcal{V}$; $[\![F(t_1, \ldots, t_n)]\!]_\sigma = F^{\mathbb{A}}([\![t_1]\!]_\sigma, \ldots, [\![t_n]\!]_\sigma)$ for any $F \in \Sigma$.

Let $s = t$ be a first-order equation, $\mathbb{A}$ a Σ-algebra and σ an assignment into $\mathbb{A}$. We denote $\mathbb{A}, \sigma \models s = t$ if $[\![s]\!]_\sigma = [\![t]\!]_\sigma$. We also denote $\mathbb{A} \models s = t$ if $[\![s]\!]_\sigma = [\![t]\!]_\sigma$ for any assignment σ into $\mathbb{A}$.

Let $\mathbb{A}$ be a Σ-algebra and E be a set of first-order equations. We say that $\mathbb{A}$ is a *model* of E, denoted by $\mathbb{A} \models E$, if $\mathbb{A} \models s = t$ for all $s = t \in E$. We say that $s = t$ is a *semantic consequence* of E, denoted by $E \models s = t$, if $\mathbb{A} \models s = t$ for any model $\mathbb{A}$ of E. We denote by $Alg_\Sigma(E)$ the class of all models of E.

Proposition 2.4 ([4]) Let E be a set of first-order equations and $s = t$ be a first-order equation. Then the following properties are equivalent.

(1) $E \models s = t$
(2) $s \overset{*}{\underset{E}{\leftrightarrow}} t$
(3) $E \vdash s = t$

Let $\mathbb{A} = \langle A, \Sigma^{\mathbb{A}}\rangle$ and $\mathbb{B} = \langle B, \Sigma^{\mathbb{B}}\rangle$ be Σ-algebras. A mapping ϕ from A to B is said to be a *homomorphism* if $\phi(F^{\mathbb{A}}(a_1, \ldots, a_n)) = F^{\mathbb{B}}(\phi(a_1), \ldots, \phi(a_n))$ for any $F \in \Sigma$ and $a_i \in A$. We denote by $Hom(\mathbb{A}, \mathbb{B})$ the class of all homomorphisms from $\mathbb{A}$ to $\mathbb{B}$. Let K be a class of Σ-algebra. An algebra $\mathbb{I} \in K$ is said to be an *initial algebra* in K if for any $\mathbb{A} \in K$ there exists a unique homomorphism from $\mathbb{I}$ to $\mathbb{A}$, that is, $|Hom(\mathbb{I}, \mathbb{A})| = 1$.

Proposition 2.5 ([4]) Let E be a set of first-order equations and $s = t$ be a first-order equation. The quotient ground term algebra $\mathbb{T}_{ar}(\Sigma)/\overset{*}{\underset{E}{\leftrightarrow}}$ is an initial algebra in the class $Alg_\Sigma(E)$. Moreover the following properties are equivalent.

(1) $\mathbb{T}_{ar}(\Sigma)/\overset{*}{\underset{E}{\leftrightarrow}} \models s = t$
(2) $s\theta_g \overset{*}{\underset{E}{\leftrightarrow}} t\theta_g$ for all ground substitution θ_g
(3) $E \vdash_{ind} s = t$

We often use the property (2) to define inductive theorems.

(Assumption)

$$\frac{}{s = t} \quad \text{if } s = t \in E$$

(Reflexivity)

$$\frac{}{t = t}$$

(Symmetry)

$$\frac{t = s}{s = t}$$

(Transitivity)

$$\frac{s = u \quad u = t}{s = t}$$

(Substitutivity)

$$\frac{s = t}{s\theta = t\theta}$$

(Monotonicity)

$$\frac{s_0 = t_0 \quad \cdots \quad s_n = t_n}{s_0(s_1, \ldots, s_n) = t_0(t_1, \ldots, t_n)} \quad \text{if } s_0(s_1, \ldots, s_n) \in T_\tau(\Sigma)$$

(Functionality)

$$\frac{\forall u \in T_{\tau(z)}(\Sigma).\ s\{z := u\} = t\{z := u\}}{s = t}$$

(Extensionality)

$$\frac{s(z) = t(z)}{s = t} \quad \text{if } z \notin Var(s = t)$$

Fig. 2 Inference rules for the higher-order equational logic.

3. Inductive Theorems

Basic ingredients of the theory of algebraic specification are equational logic and its semantics is based on Σ-algebra [10], [12]–[14], [24]. Algebraic specifications for higher-order languages were studied in [19], [20]. In this section, based on Meinke's formulation [19], we give a syntax of a higher-order equational logic on STRSs, and define inductive theorems in STRSs.

Definition 3.1 A *simply-typed equation*, written by $s = t$, is a pair of simply-typed terms with the same types ($\tau(s) = \tau(t)$). We also denote $Var(s) \cup Var(t)$ by $Var(s = t)$.

Note that STRSs are often regarded as simply-typed equation sets.

Next we define theorems and inductive theorems in the higher-order equational logic by using inference rules displayed in Fig. 2.

Definition 3.2 Let E be a set of simply-typed equations. A simply-typed equation $s = t$ is said to be a *theorem* in E, denoted by $E \vdash s = t$, if it is deducible by the inference rules in Fig. 2 except for (Functionality) and (Extensionality)-rules. A simply-typed equation $s = t$ is said to be an *inductive theorem* in E, denoted by $E \vdash_{ind} s = t$, if it is deducible by the inference rules in Fig. 2.

The differences of inference rules between first-order and higher-order settings are as follows: (Monotonicity) and (Functionality)-rules are modified, and (Extensionality)-rule is added.

Example 3.3 Consider the following STRS R with $\tau(Rev) = \tau(Frev) = L \to L$, $\tau(Ap) = \tau(F) = L \to L \to L$, $\tau(::) = N \to L \to L$ and $\tau(Nil) = L$ (this system can also be considered as TRS):

$$\left\{\begin{array}{rcl} Ap(Nil, xs) & \to & xs \\ Ap(x :: xs, ys) & \to & x :: Ap(xs, ys) \\ Rev(Nil) & \to & Nil \\ Rev(x :: xs) & \to & Ap(Rev(xs), x :: Nil) \\ F(Nil, xs) & \to & xs \\ F(x :: xs, ys) & \to & F(xs, x :: ys) \\ Frev(xs) & \to & F(xs, Nil) \end{array}\right.$$

Note that the transformation from Rev to $Frev$ is a typical example of program optimization.

Both Rev and $Frev$ in R represent the same list-reverse function, and we have

$$R \vdash_{ind} Rev(xs) = Frev(xs),$$
$$R \vdash_{ind} Rev = Frev.$$

However the notion of theorems do not equate Rev and $Frev$, that is,

$$R \nvdash Rev(xs) = Frev(xs),$$
$$R \nvdash Rev = Frev.$$

4. Primitive Inductive Theorems

In this section, we give the notion of primitive inductive theorems, and study the relation between primitive inductive theorems and inductive theorems.

4.1 Characterizing Inductive Theorems by Primitive Inductive Theorems

The notion of primitive inductive theorems is a natural extension of the property (2) in Proposition 2.5.

Definition 4.1 Let R be a set of simply-typed equations. A simply-typed equation $s = t$ is said to be a *primitive inductive theorem* in R, denoted by $R \vdash_{pind} s = t$, if $S_g[s\theta_c] \overset{*}{\underset{R}{\leftrightarrow}} S_g[t\theta_c]$ for all closed substitution θ_c (i.e. $\forall x \in Var(s = t).\ \theta_c(x) \in T_\tau(\Sigma)$) and ground suffix context $S_g[\,] \in T_{\mathcal{B}}(\Sigma \cup \{\Box\})$. We also denote $R \vdash_{pind} E$ if $R \vdash_{pind} s = t$ for all $s = t \in E$.

If an equation $s = t$ has a basic type then we have $R \vdash_{pind} s = t \iff s\theta_c \overset{*}{\underset{R}{\leftrightarrow}} t\theta_c$ for all closed substitution θ_c, since $\Box$ is the only suffix context having a basic-typed hole.

Theorem 4.2 Let R be a set of simply-typed equations and $s = t$ a simply-typed equation. Then

$$R \vdash_{pind} s = t \Rightarrow R \vdash_{ind} s = t$$

Proof. Suppose that $S_g[s\theta_c] \overset{*}{\underset{R}{\leftrightarrow}} S_g[t\theta_c]$ for all closed substitution θ_c and ground suffix context $S_g[\,]$. Let $X = \{u = v \mid R \vdash_{ind} u = v\}$. From the definition of $\underset{R}{\to}$, we have $\underset{R}{\to} \subseteq X$ by rules (Assumption), (Substitutivity) and (Monotonicity). Thus it follows that $\overset{*}{\underset{R}{\leftrightarrow}} \subseteq X$ from rules (Reflexivity), (Symmetry) and (Transitivity). Hence $S_g[s\theta_c] = S_g[t\theta_c] \in X$. Let $S_g[\,] \equiv \Box(u_1, \ldots, u_m)$ and $S[\,] \equiv \Box(z_1, \ldots, z_m)$ where $z_1, \ldots, z_m$ are fresh variables. We define θ'_c by $\theta'_c(z_i) = u_i$ $(i = 1, \ldots, n)$ and $\theta'_c(z) = \theta_c(z)$ for the other variables. Then $S_g[s\theta_c] \equiv S[s]\theta'_c$ and $S_g[t\theta_c] \equiv S[t]\theta'_c$. From (Functionality)-rule, we have $S[s] = S[t] \in X$. From (Extensionality)-rule, we have $s = t \in X$. □

Since the notion of primitive inductive theorems is a natural extension of the property (2) in Proposition 2.5, one might think that the simply-typed version of Proposition 2.5 would be obtained. However this is not true; the inverse of Theorem 4.2 does not hold.

Example 4.3 Consider again the STRS R shown in Example 3.3. Suppose that $Apply \in \Sigma$ with $\tau(Apply) = (L \to L) \to L \to L$. Then we have

$$R \vdash_{ind} Apply(Rev, xs) = Apply(Frev, xs),$$
$$R \nvdash_{pind} Apply(Rev, xs) = Apply(Frev, xs).$$

From this example, some readers might guess that inductive theorems coincide with the monotonic closure of primitive inductive theorems. However, this is also not true.

Example 4.4 We consider the following STRS R:

$$\left\{\begin{array}{rcl} I(x) & \to & x \\ I'(x) & \to & x \\ Apply(I, x) & \to & I(x) \end{array}\right.$$

where $\tau(I) = \tau(I') = N \to N$ and $\tau(Apply) = (N \to N) \to N \to N$. Suppose that X is the monotonic closure of primitive inductive theorems in R, that is, X is the smallest set such that $R \vdash_{pind} s = t$ implies $s = t \in X$, and $s_i = t_i \in X$ $(i = 0, \ldots, n)$ implies $s_0(s_1, \ldots, s_n) = t_0(t_1, \ldots, t_n)$. Then $Apply(I', x) = Apply(I, x) \in X$ follows from $R \vdash_{pind} I' = I$, and $Apply(I, x) = x \in X$ holds. However X is not transitive because of $Apply(I', x) = x \notin X$. Since $R \vdash_{ind} Apply(I', x) = x$, the monotonic closure X of primitive inductive theorems cannot characterize inductive theorems.

Definition 4.5 Let R be a set of simply-typed equations and $s = t$ a simply-typed equation. We define $R \vdash^1_{pind} s = t$ by $R \vdash_{pind} s = t$; $R \vdash^{n+1}_{pind} s = t$ by $R' \vdash_{pind} s = t$ where $R' = \{u = v \mid R \vdash^n_{pind} u = v\}$.

Theorem 4.6 Let R be a set of simply-typed equations and $s = t$ a simply-typed equation. Then $R \vdash_{ind} s = t$ iff $R \vdash^{n}_{pind} s = t$ for some n.

Proof. We prove that $R \vdash^{n}_{pind} s = t$ implies $R \vdash_{ind} s = t$ by induction on n. The case $n = 1$ is Theorem 4.2. Suppose that $n > 1$. Let $R' = \{u = v \mid R \vdash^{n-1}_{pind} u = v\}$. Then $R' \vdash_{ind} s = t$ follows from Theorem 4.2. From the induction hypothesis, $R \vdash_{ind} R'$. Thus we have $R \vdash_{ind} s = t$.

We prove that $R \vdash_{ind} s = t$ implies $R \vdash^{n}_{pind} s = t$ for some n by induction on the depth of the proof tree of $R \vdash_{ind} s = t$. Suppose that $s = t$ is deduced by an inference rule from a subproof P', and E (possibly empty) is the consequence of P'. In the case that the inference rule is either (Assumption) or (Reflexivity), $R \vdash_{pind} s = t$. In other cases, $R \vdash^{n}_{pind} E$ for some n follows from the induction hypothesis, hence $R \vdash^{n+1}_{pind} s = t$. □

4.2 Sufficient Condition

We are now going to explore a sufficient condition for the inverse of Theorem 4.2. This condition plays an important roll for disproving inductive theorems (see Sect. 6).

Before we present the condition, we prepare several notions and lemmas. In the following, we assume Σ is partitioned into $\mathcal{D}$ and C, that is, $\Sigma = \mathcal{D} \cup C$ and $\mathcal{D} \cap C = \emptyset$. Elements of $\mathcal{D}$ are called *defined symbols*, and those of C are called *constructors*.

Definition 4.7 A term $t \in T(\Sigma)$ is said to be a *value* if $root(t') \in C$ for any $t' \in Sub_{\mathcal{B}}(t)$. We denote the set of all values by $Val(C, \mathcal{D})$, and denote the set of all basic-typed values by $Val_{\mathcal{B}}(C, \mathcal{D})$. An STRS R is said to be *quasi-reducible*, denoted by $QR(R)$, if any basic-typed term $F(t_1, \ldots, t_n)$ is not a normal form whenever $t_1, \ldots, t_n \in Val(C, \mathcal{D})$ and $F \in \mathcal{D}$.

Definition 4.8 A term $t \in T(C, \mathcal{V}_h)$ is said to be a *pseudo-value* if any variable occurrence is at a leaf position. We denote all pseudo-values by $PVal(C, \mathcal{V}_h)$. An STRS R is said to be *strongly quasi-reducible*, denoted by $SQR(R)$, if any basic-typed term $F(t_1, \ldots, t_n)$ is reducible whenever $t_1, \ldots, t_n \in PVal(C, \mathcal{V}_h)$ and $F \in \mathcal{D}$.

Example 4.9 Let $C = \{0, P\}$, $\mathcal{D} = \{Add\}$ and $f, g \in \mathcal{V}$ such that $\tau(0) = N$, $\tau(P) = (N \to N) \to N \to N$, $\tau(Add) = \tau(g) = N \to N \to N$ and $\tau(f) = N \to N$. Then 0, Add and $P(Add(0), 0)$ are values; 0, g and $P(f, 0)$ are pseudo-values; $Add(0, 0)$ is neither a value nor a pseudo-value.

Lemma 4.10 If $SQR(R)$ then $GNF(R) \subseteq Val_{\mathcal{B}}(C, \mathcal{D})$.

Proof. From the definitions of GNF and $Val_{\mathcal{B}}$, we have $GNF(R) \subseteq T_{\mathcal{B}}(\Sigma)$ and $Val_{\mathcal{B}}(R) \subseteq T_{\mathcal{B}}(\Sigma)$. Hence it suffices to show $t \notin Val_{\mathcal{B}}(C, \mathcal{D}) \Rightarrow t \notin GNF(R)$ for all $t \in T_{\mathcal{B}}(\Sigma)$.

Suppose that $t \in T_{\mathcal{B}}(\Sigma)$ and $t \notin Val_{\mathcal{B}}(C, \mathcal{D})$. Let u be a minimal size term in $Sub_{\mathcal{B}}(t)$ such that $root(u) = F \in \mathcal{D}$. Let $u_1, \ldots, u_n$ be terms such that $u \equiv C[u_1, \ldots, u_n]$, each $root(u_i)$ is a defined symbol and $C[\,]$ is a constructor context except for the root position. By the minimality of u, each u_i is not of basic types. Hence there exist $v_1, \ldots, v_m \in PVal(C, \mathcal{V}_h)$ such that $F(v_1, \ldots, v_m) \equiv C[z_1, \ldots, z_n]$ where z_j are distinct fresh variables. From $SQR(R)$, $C[z_1, \ldots, z_n]$ is reducible. Hence $C[u_1, \ldots, u_n]$ is reducible, which implies the reducibility of t. Thus $t \notin GNF(R)$. □

Definition 4.11 An STRS R is said to be *sufficient complete*, denoted by $SC(R)$, if $\forall t \in T_{\mathcal{B}}(\Sigma)$. $\exists v \in Val_{\mathcal{B}}(C, \mathcal{D})$. $t \xrightarrow[R]{*} v$.

Lemma 4.12 If $GWN(R)$ and $SQR(R)$ then $SC(R)$ holds.

Proof. Let $t \in T_{\mathcal{B}}(\Sigma)$. By $GWN(R)$, $t \xrightarrow[R]{*} u$ for some $u \in GNF(R)$. By Lemma 4.10, $u \in Val_{\mathcal{B}}(C, \mathcal{D})$. □

Definition 4.13 If $ord(\tau(c)) \leq 2$ for any $c \in C$ then we say that the set C of constructors has a *first-order structure.*

Note that for any simply-typed term $c(t_1, \ldots, t_n)$ with $c \in C$, each t_i is of basic types whenever C has a first-order structure.

Lemma 4.14 If C has a first-order structure then $Val_{\mathcal{B}}(C, \mathcal{D}) = T_{\mathcal{B}}(C)$.

Proof. $Val_{\mathcal{B}}(C, \mathcal{D}) \supseteq T_{\mathcal{B}}(C)$ is trivial. Assume that $Val_{\mathcal{B}}(C, \mathcal{D}) \setminus T_{\mathcal{B}}(C) \neq \emptyset$. Let t be a minimal size term in $Val_{\mathcal{B}}(C, \mathcal{D}) \setminus T_{\mathcal{B}}(C)$, and $t \equiv a(t_1, \ldots, t_n)$. We have $a \in C$ because of $t \in Val_{\mathcal{B}}(C, \mathcal{D})$. Since C has a first-order structure, each t_i is of basic types, and hence $\forall i.\ t_i \in Val_{\mathcal{B}}(C, \mathcal{D})$. From the minimality of t, each t_i is in $T_{\mathcal{B}}(C)$. Hence $a(t_1, \ldots, t_n) \in T_{\mathcal{B}}(C)$. It is a contradiction. □

Lemma 4.15 Let R be an STRS. If the following conditions hold:

(a) $GSN(R)$,
(b) $GCR(R)$,
(c) $SQR(R)$,
(d) C has a first-order structure, and
(e) $l \notin T_{\tau}(C, \mathcal{V})$ for any $l \to r \in R$,

then $R \vdash_{pind} t\theta_c = t\theta'_c$ for any $t \in T_{\mathcal{B}}(\Sigma, \mathcal{V})$ and any closed substitutions θ_c and θ'_c such that $R \vdash_{pind} z\theta_c = z\theta'_c$ for all $z \in Var(t)$.

Proof. We prove the claim by induction on $t\theta_c$ with respect to $(\xrightarrow[R]{} \cup \rhd_{sub})^+$. Note that the well-foundedness on $T_{\mathcal{B}}(\Sigma)$ of $(\xrightarrow[R]{} \cup \rhd_{sub})^+$ is guaranteed by the condition (a). Let $t \equiv a(t_1, \ldots, t_n)$. There are three cases.

- Suppose that $a \in C$. Since each t_i is of basic types from the condition (d), we have the induction hypothesis $R \vdash_{pind} t_i\theta_c = t_i\theta'_c$ for each i. Hence $t_i\theta_c \overset{*}{\underset{R}{\leftrightarrow}} t_i\theta'_c$, which implies $t\theta_c \overset{*}{\underset{R}{\leftrightarrow}} t\theta'_c$. Thus $R \vdash_{pind} t\theta_c = t\theta'_c$.
- Suppose that $a \in \mathcal{D}$. Consider t_i of basic types. From the induction hypothesis, $R \vdash_{pind} t_i\theta_c = t_i\theta'_c$, that is $t_i\theta_c \overset{*}{\underset{R}{\leftrightarrow}} t_i\theta'_c$. Since we have $SC(R)$ from conditions (a) and (c), and Lemma 4.12, there exist ground constructor terms u_i and v_i such that $t_i\theta_c \overset{*}{\underset{R}{\rightarrow}} u_i$ and $t_i\theta'_c \overset{*}{\underset{R}{\rightarrow}} v_i$ by the condition (d) and Lemma 4.14. Here, we have $u_i, v_i \in GNF(R)$ from the condition (e). Hence $u_i \equiv v_i \in T_\tau(C)$ follows from the condition (b).

  For each t_i of higher-order types, let $u_i \equiv v_i \equiv f_i$ where f_i is a fresh variable. Let $\sigma' = \{f_i := t_i \mid t_i \text{ is of higher-order types }\}$.

  Now we have

$$\begin{aligned} t\theta_c &\equiv a(t_1\theta_c, \ldots, t_n\theta_c) \\ &\overset{*}{\underset{R}{\rightarrow}} a(u_1\sigma'\theta_c, \ldots, u_n\sigma'\theta_c) \\ &\equiv a(u_1, \ldots, u_n)\sigma'\theta_c. \end{aligned}$$

  By the condition (c), there exist $l \rightarrow r \in R$ and σ such that $l\sigma \equiv a(u_1, \ldots, u_n)$. Hence, we have

$$a(u_1, \ldots, u_n)\sigma'\theta_c \equiv l\sigma\sigma'\theta_c \underset{R}{\rightarrow} r\sigma\sigma'\theta_c.$$

  Similarly, we have $t\theta'_c \overset{+}{\underset{R}{\rightarrow}} r\sigma\sigma'\theta'_c$. From the induction hypothesis, we have $R \vdash_{pind} r\sigma\sigma'\theta_c = r\sigma\sigma'\theta'_c$, which implies $R \vdash_{pind} t\theta_c = t\theta'_c$.
- Suppose that $a \in \mathcal{V}$. Let $t' \equiv (a\theta_c)(t_1, \ldots, t_n)$. We consider the equation $t'\theta_c = t'\theta'_c$. Since $t\theta_c \equiv t'\theta_c$ and $root(t') \in C \cup \mathcal{D}$, $R \vdash_{pind} t'\theta_c = t'\theta'_c$ follows from previous cases. Hence $t\theta_c \equiv t'\theta_c \overset{*}{\underset{R}{\leftrightarrow}} t'\theta'_c \equiv (a\theta_c)(t_1\theta'_c, \ldots, t_n\theta'_c)$. Since $R \vdash_{pind} a\theta_c = a\theta'_c$, by taking a ground suffix context $\Box(t_1\theta'_c, \ldots, t_n\theta'_c)$, we have $(a\theta_c)(t_1\theta'_c, \ldots, t_n\theta'_c) \overset{*}{\underset{R}{\leftrightarrow}} (a\theta'_c)(t_1\theta'_c, \ldots, t_n\theta'_c) \equiv t\theta'_c$. Hence $R \vdash_{pind} t\theta_c = t\theta'_c$. □

Lemma 4.16 Let R be an STRS. Suppose that conditions (a)–(e) in Lemma 4.15 hold. If $R \vdash_{pind} s_i = t_i$ $(i = 0, 1, \ldots, n)$ then $R \vdash_{pind} s_0(s_1, \ldots, s_n) = t_0(t_1, \ldots, t_n)$.

Proof. We show that

$$S_g[s_0(s_1, \ldots, s_n)\theta_c] \overset{*}{\underset{R}{\leftrightarrow}} S_g[t_0(t_1, \ldots, t_n)\theta_c]$$

for any closed substitution θ_c and ground suffix context $S_g[\,]$. Let $s'_i \equiv s_i\theta_c$, $t'_i \equiv t_i\theta_c$ $(i = 0, 1, \ldots, n)$ and $S_g[\,] \equiv \Box(s'_{n+1}, \ldots, s'_m) \equiv \Box(t'_{n+1}, \ldots, t'_m)$. From the assumption, $R \vdash_{pind} s'_i = t'_i$ for $i = 0, 1, \ldots, m$. Let θ and θ' be substitutions such that $\theta(z_i) = s'_i$ and $\theta'(z_i) = t'_i$ for each i. Clearly, θ and θ' are closed and for all $i = 0, \ldots, n$, $R \vdash_{pind} z_i\theta = z_i\theta'$. From Lemma 4.15, $R \vdash_{pind} z_0(z_1, \ldots, z_m)\theta = z_0(z_1, \ldots, z_m)\theta'$, that is, $R \vdash_{pind} S_g[s_0(s_1, \ldots, s_n)\theta_c] = S_g[t_0(t_1, \ldots, t_n)\theta_c]$. Hence $S_g[s_0(s_1, \ldots, s_n)\theta_c] \overset{*}{\underset{R}{\leftrightarrow}} S_g[t_0(t_1, \ldots, t_n)\theta_c]$. □

Theorem 4.17 Let R be an STRS. Suppose that conditions (a)–(e) in Lemma 4.15 hold. Then

$$R \vdash_{pind} s = t \iff R \vdash_{ind} s = t$$

Proof. The ($\Rightarrow$)-part is Theorem 4.2. Suppose $R \vdash_{ind} s = t$. We prove $R \vdash_{pind} s = t$ by induction on the depth of the proof tree of $R \vdash_{ind} s = t$. In the case that $s = t$ is deduced by (Monotonicity)-rule, $R \vdash_{pind} s = t$ follows from the induction hypothesis and Lemma 4.16. Other cases are routine. □

This theorem plays an important role in the implicit induction methods for disproving inductive theorems (Theorem 6.3).

5. Proving Inductive Theorems

Implicit induction methods are intended to prove inductive theorems. In the first-order setting, two kind of implicit induction methods are known: inductionless induction and rewriting induction [6], [7], [11], [16], [21]–[23]. In this section, we formulate some methods to prove primitive inductive theorems using the results in [16], [23]. These methods are also successfully applied to prove inductive theorems using Theorem 4.2.

5.1 Inductionless Induction

In this subsection, we state how to apply the inductionless induction method in [23] to STRSs.

Proposition 5.1 ([23]) Let $R_1 = \langle A, \underset{1}{\rightarrow}\rangle$ and $R_2 = \langle A, \underset{2}{\rightarrow}\rangle$ be ARSs. Suppose that all of the following conditions hold:

(i) $\underset{1}{\rightarrow} \subseteq \overset{*}{\underset{2}{\leftrightarrow}}$
(ii) $WN(R_1)$
(iii) $CR(R_2)$
(iv) $NF(R_1) \subseteq NF(R_2)$

Then we have $\overset{*}{\underset{1}{\leftrightarrow}} = \overset{*}{\underset{2}{\leftrightarrow}}$.

Based on this proposition, we show an abstract theorem for proving inductive theorems.

Theorem 5.2 Let R and R' be STRSs, and E be a set of equations. Suppose that all of the following conditions hold:

(i) $\overset{*}{\underset{R\cup E}{\leftrightarrow}} \subseteq \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$
(ii) $GWN(R)$
(iii) $GCR(R')$
(iv) $GNF(R) \subseteq GNF(R')$

Then $R \vdash_{pind} E$ and hence $R \vdash_{ind} E$.

Proof. From the condition (i), we have $\underset{R}{\rightarrow} \subseteq \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. By applying Proposition 5.1 with $T_{\mathcal{B}}(\Sigma)$ as A, we have $\overset{*}{\underset{R}{\leftrightarrow}} = \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. Hence $\overset{*}{\underset{E}{\leftrightarrow}} \subseteq \overset{*}{\underset{R}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$, which implies $R \vdash_{pind} E$. We have $R \vdash_{ind} E$ by Theorem 4.2. □

We prepare a lemma for checking $GNF(R) \subseteq GNF(R')$.

Lemma 5.3 Let R and R' be STRSs. Suppose that for any $l \to r \in R'$ there exists $l' \in Sub_{\mathcal{B}}(l)$ such that $root(l') \in \mathcal{D}$. Then $Val_{\mathcal{B}}(C, \mathcal{D}) \subseteq GNF(R')$. Moreover, if $SQR(R)$ additionally holds then $GNF(R) \subseteq GNF(R')$ holds.

Proof. Assume that $t \in Val_{\mathcal{B}}(C, \mathcal{D})$ and $t \notin GNF(R')$. Then $t \equiv C[S[l\theta]]$ for some $C[\,], S[\,], \theta, l \to r \in R$. By assumption, there exists $l' \in Sub_{\mathcal{B}}(l)$ such that $root(l') \in \mathcal{D}$. Then $l'\theta \in Sub_{\mathcal{B}}(t)$ and $root(l'\theta) \in \mathcal{D}$. This contradicts $t \notin Val_{\mathcal{B}}(C, \mathcal{D})$.

Suppose that $SQR(R)$ additionally holds. From Lemma 4.10, $GNF(R) \subseteq Val_{\mathcal{B}}(C, \mathcal{D})$. Hence $GNF(R) \subseteq GNF(R')$. □

Example 5.4 We consider the following STRS R:

$$\begin{cases} Map(f, Nil) & \to & Nil \\ Map(f, x :: xs) & \to & f(x) :: Map(f, xs) \\ Ap(Nil, xs) & \to & xs \\ Ap(x :: xs, ys) & \to & x :: Ap(xs, ys) \end{cases}$$

Suppose that $C = \{0, S, Nil, ::\}$, $\mathcal{D} = \{Ap, Map\}$ and $\tau(0) = N$, $\tau(S) = N \to N$, $\tau(Nil) = L$, $\tau(::) = N \to L \to L$, $\tau(Ap) = L \to L \to L$ and $\tau(Map) = (N \to N) \to L \to L$. Based on Theorem 5.2, we prove that the following equation is an inductive theorem in R:

$$Map(f, Ap(xs, ys)) = Ap(Map(f, xs), Map(f, ys))$$

Let R' be the union of R and the above equation. We can prove $SN(R)$ by the recursive path order in [18], and $CR(R')$ by the critical pair criterion. Hence $GWN(R)$ and $GCR(R')$ hold. Since $SQR(R)$ hold, and for any $l \to r \in R'$ there exists $l' \in Sub_{\mathcal{B}}(l)$ such that $root(l') \in \mathcal{D}$, the inclusion $GNF(R) \subseteq GNF(R')$ follows from Lemma 5.3. Therefore conditions (i)–(iv) of Theorem 5.2 hold and thus the equation above is an inductive theorem in R.

Example 5.5 We consider the following STRS R:

$$\begin{cases} I(x) & \to & x \\ \wedge(T, y) & \to & y \\ \wedge(F, y) & \to & F \\ Ands(Nil) & \to & T \\ Ands(T :: xs) & \to & Ands(xs) \\ Ands(F :: xs) & \to & F \\ \forall(p, Nil) & \to & T \\ \forall(p, x :: xs) & \to & \wedge(p(x), \forall(p, xs)) \\ Map(f, Nil) & \to & Nil \\ Map(f, x :: xs) & \to & f(x) :: Map(f, xs) \end{cases}$$

Suppose that $\tau(T) = \tau(F) = B$, $\tau(Nil) = L$, $\tau(::) = B \to L \to L$, $\tau(I) = B \to B$, $\tau(\wedge) = B \to B \to B$, $\tau(Ands) = L \to B$, $\tau(\forall) = (B \to B) \to L \to B$, and $\tau(Map) = (B \to B) \to L \to L$. In the way similar to Example 5.4, we can prove:

$$R \vdash_{ind} \forall(I, xs) = Ands(Map(I, xs))$$

5.2 Rewriting Induction

In this subsection, we apply the rewriting induction method, proposed in [22] and formalized in [16], to STRSs.

Proposition 5.6 ([16]) Let $R_1 = \langle A, \underset{1}{\rightarrow} \rangle$ and $R_2 = \langle A, \underset{2}{\rightarrow} \rangle$ be ARSs. Suppose that all of the following conditions hold:

(i) $\underset{1}{\rightarrow} \subseteq \overset{+}{\underset{2}{\rightarrow}}$
(ii) $SN(R_2)$
(iii) $RET(R_2, R_1)$

Then we have $\overset{*}{\underset{1}{\leftrightarrow}} = \overset{*}{\underset{2}{\leftrightarrow}}$.

Theorem 5.7 Let R and R' be STRSs, and E be a set of equations. Suppose that all of the following conditions hold:

(i) $\underset{R}{\rightarrow} \subseteq \overset{+}{\underset{R'}{\rightarrow}}$ and $\overset{*}{\underset{E}{\leftrightarrow}} \subseteq \overset{*}{\underset{R'}{\leftrightarrow}}$ hold in $T_{\mathcal{B}}(\Sigma)$
(ii) $GSN(R')$
(iii) $GRET(R', R)$

Then $R \vdash_{pind} E$ and $R \vdash_{ind} E$.

Proof. By applying Proposition 5.6 with $T_{\mathcal{B}}(\Sigma)$ as A, we have $\overset{*}{\underset{R}{\leftrightarrow}} = \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. Hence $\overset{*}{\underset{E}{\leftrightarrow}} \subseteq \overset{*}{\underset{R}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$, which implies $R \vdash_{pind} E$. Hence we have $R \vdash_{ind} E$ by Theorem 4.2. □

By using the rewriting induction, we can also prove inductive theorems in Example 5.4 and 5.5.

6. Disproving Inductive Theorems

For program verification, not only proving inductive theorems but also disproving ones is important. In this section, we present an automated reasoning method for disproving inductive theorems, based on the methods proposed in [11], [21] and formalized in [16].

Proposition 6.1 ([16]) Let $R_1 = \langle A, \underset{1}{\rightarrow} \rangle$ and $R_2 = \langle A, \underset{2}{\rightarrow} \rangle$ be ARSs. Suppose that all of the following conditions hold:

(i) $\underset{1}{\rightarrow} \subseteq \overset{+}{\underset{2}{\rightarrow}}$
(ii) $SN(R_2)$
(iii) $CR(R_1)$
(iv) $NF(R_1) \not\subseteq NF(R_2)$

Then we have $\overset{*}{\underset{1}{\leftrightarrow}} \neq \overset{*}{\underset{2}{\leftrightarrow}}$.

Based on this abstract result, we design implicit induction methods for disproving primitive inductive theorems.

Theorem 6.2 Let R and R' be STRSs, and E be a set of equations. Suppose that all of the following conditions hold:

(i) $\underset{R}{\rightarrow} \subseteq \overset{+}{\underset{R'}{\rightarrow}}$ and $\overset{*}{\underset{R\cup E}{\leftrightarrow}} = \overset{*}{\underset{R'}{\leftrightarrow}}$ hold in $T_{\mathcal{B}}(\Sigma)$
(ii) $GSN(R')$
(iii) $GCR(R)$
(iv) $GNF(R) \not\subseteq GNF(R')$

Then we have $R \not\vdash_{pind} E$.

Proof. From Proposition 6.1, we have $\overset{*}{\underset{R}{\leftrightarrow}} \neq \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. Assume that $R \vdash_{pind} E$. Then it is easily seen that $\overset{*}{\underset{E}{\leftrightarrow}} \subseteq \overset{*}{\underset{R}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. Hence $\overset{*}{\underset{R}{\leftrightarrow}} = \overset{*}{\underset{R\cup E}{\leftrightarrow}} = \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. It is a contradiction. □

We have already presented a sufficient condition of primitive inductive theorems to be inductive theorems (Theorem 4.17). It means that $R \not\vdash_{pind} E$ guarantees $R \not\vdash_{ind} E$ provided that R satisfies conditions (a)–(e) of Lemma 4.15. From this fact, we can use the implicit induction method to disprove inductive theorems.

Theorem 6.3 Let R and R' be STRSs, and E be a set of equations. Suppose that in addition to the properties (i),(ii),(iii) and (iv) in Theorem 6.2, all of the following properties hold:

(v) $SQR(R)$,
(vi) C has a first-order structure, and
(vii) $l \notin T_\tau(C, \mathcal{V})$ for any $l \rightarrow r \in R$.

Then we have $R \not\vdash_{ind} E$.

Proof. From Theorems 6.2 and 4.17. Note that $GSN(R')$ implies $GSN(R)$. □

Example 6.4 Let R be the following STRS:

$$\begin{cases} Add(x,0) & \rightarrow & 0 \\ Add(x,S(y)) & \rightarrow & S(Add(x,y)) \end{cases}$$

where $\mathcal{D} = \{Add\}$ and $C = \{0, S\}$. We prove

$$R \not\vdash_{ind} Add(0) = S$$

based on Theorem 6.3. Let $E = \{Add(0) = S\}$ and $R' = R \cup \{Add(0,z) \rightarrow S(z), S(0) \rightarrow 0\}$. Since $\forall t \in T_{\mathcal{B}}(\Sigma).Add(0,t) \underset{E}{\leftrightarrow} S(t)$ and $S(0) \underset{E}{\leftrightarrow} Add(0,0) \underset{R}{\rightarrow} 0$, we have $\overset{*}{\underset{R\cup E}{\leftrightarrow}} \supseteq \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. From the construction of R', we have $\underset{R}{\rightarrow} \subseteq \overset{+}{\underset{R'}{\rightarrow}}$ and $\overset{*}{\underset{R\cup E}{\leftrightarrow}} \subseteq \overset{*}{\underset{R'}{\leftrightarrow}}$ hold in $T_{\mathcal{B}}(\Sigma)$. Hence the condition (i) holds. The conditions (v),(vi) and (vii) can be easily shown. We can prove $SN(R')$ by the recursive path order in [18], and $CR(R)$ by the critical pair criterion. Hence $GSN(R')$ and $GCR(R)$ hold. The condition (iv) holds because $S(0) \in GNF(R)$ and $S(0) \notin GNF(R')$. Therefore we have $R \not\vdash_{ind} Add(0) = S$.

Example 6.5 Let R be the following STRS:

$$\begin{cases} I(x) & \rightarrow & x \\ \wedge(T,y) & \rightarrow & y \\ \wedge(F,y) & \rightarrow & F \\ Ands(Nil) & \rightarrow & T \\ Ands(T :: xs) & \rightarrow & Ands(xs) \\ Ands(F :: xs) & \rightarrow & F \\ \forall(p, Nil) & \rightarrow & \underline{F} \\ \forall(p, x :: xs) & \rightarrow & \wedge(p(x), \forall(p, xs)) \\ Map(f, Nil) & \rightarrow & Nil \\ Map(f, x :: xs) & \rightarrow & f(x) :: Map(f, xs) \end{cases}$$

This example is slightly different from Example 5.5, that is, $\forall(p, Nil) \rightarrow \underline{F}$, which is problematic. Here, we prove

$$R \not\vdash_{ind} \forall(I, xs) = Ands(Map(I, xs))$$

based on Theorem 6.3. Let $E = \{\forall(I, xs) = Ands(Map(I, xs))\}$ and $R' = R \cup \{\forall(I, xs) \rightarrow Ands(Map(I, xs)), F \rightarrow T\}$. Since $F \underset{R}{\leftarrow} \forall(I, Nil) \underset{E}{\leftrightarrow} Ands(Map(I, Nil)) \underset{R}{\rightarrow} Ands(Nil) \underset{R}{\rightarrow} T$, we have $\overset{*}{\underset{R\cup E}{\leftrightarrow}} = \overset{*}{\underset{R'}{\leftrightarrow}}$ in $T_{\mathcal{B}}(\Sigma)$. Hence the condition (i) holds. The conditions (v), (vi) and (vii) trivially hold. We can prove $SN(R')$ by the recursive path order in [18], and $CR(R)$ by the critical pair criterion. Hence $GSN(R')$ and $GCR(R)$ hold. The condition (iv) holds because $F \in GNF(R)$ and $F \notin GNF(R')$. Therefore we have $R \not\vdash_{ind} \forall(I, xs) = Ands(Map(I, xs))$.

7. Characterizing Inductive Theorems

Algebraic specification is based on equational logic whose semantics is given on Σ-algebra [10], [12]–[14], [24]. Higher-order theories have also been considered in [19], [20].

In Sect. 3, we gave a syntax of a higher-order equational logic, and the definition of inductive theorems based on Meinke's formulation [19]. To justify our definition, we should give a semantic counter parts and show that our inductive theorems can be characterized by the initial extensional semantics, because our framework (STRSs) is different from Meinke's one.

7.1 Syntax

From this subsection to subsection 7.3, we study higher-order equational logic on untyped systems (UTRSs). For any set E of untyped equations and an untyped equation $s = t$, we define $E \vdash s = t$ and $E \vdash_{ind} s = t$ as similar to ones in simply-typed systems except for type constraints.

Here we prepare for a lemma needed later on.

Lemma 7.1 Let E be a set of equations. Then $E \vdash s = t \iff s \overset{*}{\underset{E}{\leftrightarrow}} t$.

Proof. ($\Rightarrow$) By induction on the depth of proof trees. ($\Leftarrow$) It suffices to show that $E \vdash s = t$ whenever $s \underset{E}{\rightarrow} t$. Let $s \equiv C[S[e_1\theta]]$ and $t \equiv C[S[e_2\theta]]$. The equation $e_1 = e_2$ is deducible by (Assumption)-rule. $e_1\theta = e_2\theta$ is deduced by (Substitutivity)-rule. $S[e_1\theta] = S[e_2\theta]$ is deduced by (Monotonicity)-rule. Finally, we obtain $E \vdash C[S[e_1\theta]] = C[S[e_2\theta]]$ by (Monotonicity)-rule. □

7.2 $\Sigma^@$-Algebra

We present the semantic counter parts by curried terms, which is one of the most standard way for handling higher-order function in first-order systems. For example, the *Map*-function is represented in the first-order term rewriting as follows:

$$\begin{cases} @(@(Map, f), Nil) \rightarrow Nil \\ @(@(Map, f), @(@(Cons, x), xs)) \\ \quad \rightarrow @(@(Cons, @(f, x)), @(@(Map, f), xs)) \end{cases}$$

Definition 7.2 We define $\Sigma^@ = \Sigma \cup \{@\}$, and denote by $T^@(\Sigma, \mathcal{V})$ the set of first-order terms generated by $\Sigma^@$ and $\mathcal{V}$ under $ar(@) = 2$ and $ar(a) = 0$ for any symbol a ($\neq @$). We also denote the term algebra $\mathbb{T}_{ar}(\Sigma^@, \mathcal{V})$ by $\mathbb{T}^@(\Sigma, \mathcal{V})$. The ground term algebra $\mathbb{T}^@(\Sigma, \emptyset)$ is denoted by $\mathbb{T}^@(\Sigma)$.

Definition 7.3 For any untyped term $t \in T(\Sigma, \mathcal{V})$, we inductively define first-order term $t^@ \in T^@(\Sigma, \mathcal{V})$ as follows:

- $a^@ = a$ for any $a \in \Sigma \cup \mathcal{V}$
- $a(t_1, \ldots, t_n)^@ = @(a(t_1, \ldots, t_{n-1})^@, t_n^@)$ if $n \geq 1$

We naturally extend the notion over substitutions as $\theta^@(x) = (\theta(x))^@$, and over sets of pairs (like equations or rules) as $E^@ = \{(s^@, t^@) \mid (s, t) \in E\}$.

We notice that $T^@(\Sigma, \mathcal{V}) = \{t^@ \mid t \in T(\Sigma, \mathcal{V})\}$, and any context on $T^@(\Sigma, \mathcal{V})$ has a form $C^@[S^@[\,]]$ for some context $C[\,]$ and suffix context $S[\,]$ on $T(\Sigma, \mathcal{V})$.

Lemma 7.4 $C[S[t\theta]]^@ \equiv C^@[S^@[t^@\theta^@]]$

Proof. Firstly we prove $(t\theta)^@ \equiv t^@\theta^@$ by induction on $|t|$. Let $t \equiv a(t_1, \ldots, t_n)$. The case $n = 0$ is trivial. Suppose that $n > 0$ and $u \equiv a(t_1, \ldots, t_{n-1})$. Then $(t\theta)^@ \equiv (u(t_n)\theta)^@ \equiv ((u\theta)(t_n\theta))^@ \equiv @((u\theta)^@, (t_n\theta)^@) \equiv @(u^@\theta^@, t_n^@\theta^@) \equiv @(u^@, t_n^@)\theta^@ \equiv t^@\theta^@$.

We can prove $S[t\theta]^@ \equiv S^@[(t\theta)^@]$ by induction on $S[\,]$, and $C[S[t\theta]]^@ \equiv C^@[S[t\theta]^@]$ by induction on $C[\,]$. Hence we obtain $C[S[t\theta]]^@ \equiv C^@[S^@[t^@\theta^@]]$. □

Lemma 7.5 Let E be a set of equations. Then $s \underset{E}{\rightarrow} t \iff s^@ \underset{E^@}{\rightarrow} t^@$.

Proof. ($\Rightarrow$) The claim follows from Lemma 7.4. ($\Leftarrow$) Let $s^@ \underset{E^@}{\rightarrow} t^@$. Then $s^@ \equiv C^@[S^@[e_1^@\theta^@]]$ and $t^@ \equiv C^@[S^@[e_2^@\theta^@]]$ for some $C^@[\,]$, $S^@[\,]$, $\theta^@$ and $e_1^@ = e_2^@ \in E^@$, because any context on $T^@(\Sigma, \mathcal{V})$ has a form $C^@[S^@[\,]]$. From Lemma 7.4, $s \equiv C[S[e_1\theta]]$ and $t \equiv C[S[e_2\theta]]$. Hence $s \underset{E}{\rightarrow} t$. □

Definition 7.6 Let $s = t$ be an equation and E be a set of equations. For any $\Sigma^@$-algebra $\mathbb{A}$, we denote $\mathbb{A} \models_@ s = t$ instead of $\mathbb{A} \models s^@ = t^@$. We also denote $E \models_@ s = t$ if $\mathbb{A} \models_@ s = t$ for any model $\mathbb{A}$ of $E^@$.

Thanks to Lemma 7.1, Lemma 7.5 and Proposition 2.4, we obtain the following theorem:

Theorem 7.7 Let E be a set of equations and $s = t$ be an equation. Then $E \vdash s = t \iff E \models_@ s = t$.

7.3 Initial Extensional Model

In first-order algebraic specification, inductive theorems are characterized by the initial algebra semantics, that is, characterized in $\mathbb{T}_{ar}(\Sigma)/\overset{*}{\underset{E}{\leftrightarrow}}$, which is an initial algebra in $Alg_\Sigma(E)$.

The initial algebra semantics cannot characterize inductive theorems in higher-order settings, because the extensionality is built in the syntax. In this subsection, we show that there exists an initial extensional model, which characterizes inductive theorems. For any $t^@ \in T^@(\Sigma)$, we denote its interpretation by $[\![t^@]\!]$, because $[\![t^@]\!]_\sigma$ is independent of assignment σ.

Definition 7.8 A $\Sigma^@$-algebra $\mathbb{A} = \langle A, \Sigma^A \rangle$ is said to be *extensional* if $\forall t^@ \in T^@(\Sigma).@^A(a, [\![t^@]\!]) = @^A(a', [\![t^@]\!]) \Rightarrow a = a'$ for all $a, a' \in A$. Let $s = t$ be an equation and E be a set of equations. We denote by $Alg^\eta_{\Sigma^@}(E^@)$ the class of all extensional model of $E^@$. We denote $E \models^\eta_@ s = t$ if $\mathbb{A} \models_@ s = t$ for all $\mathbb{A} \in Alg^\eta_{\Sigma^@}(E^@)$.

Theorem 7.9 We define $s^@ \simeq_E t^@$ as $E \vdash_{ind} s = t$. The quotient ground term algebra $\mathbb{T}^@(\Sigma)/\simeq_E$ is an initial algebra in $Alg^\eta_{\Sigma^@}(E)$.

Proof. The outline is displayed in Fig. 3. It follows easily from the definition of inductive theorems that $\simeq_E$ is a congruence relation. Hence the quotient algebra $\mathbb{T}^@(\Sigma)/\simeq_E$ is well-defined. Let $\mathbb{A} \in Alg^\eta_{\Sigma^@}(E^@)$. Since $T^@(\Sigma) = T_{ar}(\Sigma^@)$, $\mathbb{T}^@(\Sigma)/\overset{*}{\underset{E^@}{\leftrightarrow}}$ is an initial algebra in $Alg_{\Sigma^@}(E^@)$ by Proposition 2.5. Since $Alg^\eta_{\Sigma^@}(E^@) \subseteq Alg_{\Sigma^@}(E^@)$, we have the unique homomorphism ψ from $\mathbb{T}^@(\Sigma)/\overset{*}{\underset{E^@}{\leftrightarrow}}$ to $\mathbb{A}$.

From Lemmata 7.1 and 7.5, $E \vdash s = t \iff s^@ \overset{*}{\underset{E^@}{\leftrightarrow}} t^@$. Since $E \vdash s = t$ implies $E \vdash_{ind} s = t$, $\overset{*}{\underset{E^@}{\leftrightarrow}} \subseteq \simeq_E$ holds. Hence there exists a projection $p \in Hom(\mathbb{T}^@(\Sigma)/\overset{*}{\underset{E^@}{\leftrightarrow}}, \mathbb{T}^@(\Sigma)/\simeq_E)$, which maps $[t]_{\overset{*}{\underset{E^@}{\leftrightarrow}}}$ to $[t]_{\simeq_E}$.

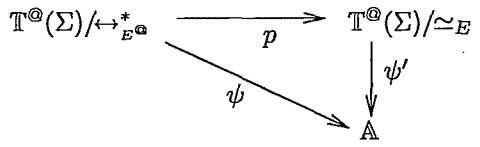


Fig. 3 Proof of Theorem 7.9.

Here we show that $|Hom(\mathbb{T}^{@}(\Sigma)/\simeq_E, \mathbb{A})| \leq 1$. Assume that $\psi_1, \psi_2 \in Hom(\mathbb{T}^{@}(\Sigma)/\simeq_E, \mathbb{A})$ such that $\psi_1 \neq \psi_2$. Since p is surjective, $\psi_1 \cdot p \neq \psi_2 \cdot p$. Since $\psi_1 \cdot p$, $\psi_2 \cdot p \in Hom(\mathbb{T}^{@}(\Sigma)/\overset{*}{\underset{E^{@}}{\leftrightarrow}}, \mathbb{A})$, $\psi_1 \cdot p = \psi = \psi_2 \cdot p$. It is a contradiction.

It remains to show that there exists a homomorphism $\psi' \in Hom(\mathbb{T}^{@}(\Sigma)/\simeq_E, \mathbb{A})$. For this, it suffices to show $s^{@} \simeq_E t^{@}$ implies $\psi([s^{@}\theta_g^{@}]) = \psi([t^{@}\theta_g^{@}])$ for any ground substitution $\theta_g^{@}$, because it guarantees the existence ψ' such that $\psi' \cdot p = \psi$. The proof proceeds by induction on the depth of proof trees for $E \vdash_{ind} s = t$. We prove only cases of (Functionality) and (Extensionality).

Suppose that $s = t$ is deduced by (Functionality)-rule. From the induction hypothesis, $\forall u^{@} \in T^{@}(\Sigma)$. $\psi([s^{@}\{z := u^{@}\}\sigma_g^{@}]) = \psi([t^{@}\{z := u^{@}\}\sigma_g]^{@})$ for all ground substitution $\sigma_g^{@}$. Hence $\psi([s^{@}\theta_g^{@}]) = \psi([t^{@}\theta_g^{@}])$ for all ground substitution $\theta_g^{@}$.

Suppose that $s = t$ is deduced by (Extensionality)-rule. From the induction hypothesis, $\exists z \notin Var(s = t)$. $\psi([@(s^{@}, z)\theta_g^{@}]) = \psi([@(t^{@}, z)\theta_g^{@}])$ for all ground substitution $\theta_g^{@}$. Since ψ is a homomorphism, $@^{\mathbb{A}}(\psi([s^{@}\theta_g^{@}]), \psi([z\theta_g^{@}])) = \psi([@(s^{@}\theta_g^{@}, z\theta_g^{@})]) = \psi([@(t^{@}\theta_g^{@}, z\theta_g^{@})]) = @^{\mathbb{A}}(\psi([t^{@}\theta_g^{@}]), \psi([z\theta_g^{@}]))$ for all ground substitution $\theta_g^{@}$. Since $\mathbb{A}$ is an extensional model, $\psi([s^{@}\theta_g^{@}]) = \psi([t^{@}\theta_g^{@}])$ for any ground substitution $\theta_g^{@}$. □

7.4 Simply-Typed Systems

In algebraic specification, type information is very important and useful. In this last subsection, we discuss how to incorporate our previous results based on untyped systems into simply-typed systems. Since our higher-order equational logic is designed independently of type structure, it is easy to combine with type systems. To be precise, we need the following simply-typed constraint for a $\Sigma^{@}$-algebra $\mathbb{A}$:

- there exists a type attachment $\tau^{\mathbb{A}}$ on A such that
  - $\tau^{\mathbb{A}}(F^{\mathbb{A}}) = \tau(F)$ for any $F \in \Sigma$, and
  - $\tau^{\mathbb{A}}(@^{\mathbb{A}}(a_1, a_2)) = \beta$ if $\tau^{\mathbb{A}}(a_1) = \alpha \to \beta$ and $\tau^{\mathbb{A}}(a_1) = \alpha$.

We also restrict σ to run over all the assignments with

- $\tau^{\mathbb{A}}(\sigma(x)) = \tau(x)$ for any $x \in \mathcal{V}$

when defining $\models_{@}$ for simply typed case.

Under the simply-typed constraint and the restriction for assignments, any properties in this section still holds on simply-typed systems. Hence inductive theorems with simply-typed systems can be characterized by the initial extensional semantics, and the simply-typed version of Proposition 2.5 is obtained.

Theorem 7.10 Let E be a set of simply-typed equations and $s = t$ be a simply-typed equation. We define $s^{@} \simeq_E t^{@}$ as $E \vdash_{ind} s = t$. Then the quotient ground term algebra $\mathbb{T}_{\tau}^{@}(\Sigma)/\simeq_E$ is an initial algebra in the class of all extensional model of E. Moreover the following properties are equivalent.

(1) $\mathbb{T}_{\tau}^{@}(\Sigma)/\simeq_E \models_{@} s = t$
(2) $E \vdash_{pind}^{n} s = t$ for some n
(3) $E \vdash_{ind} s = t$

This theorem shows that inductive theorems can be characterized by the initial extensional semantics.

8. Concluding Remarks

Under simply-typed systems, higher-order theories have also been studied in [19], [20]. Our syntactical definition is based on one in [19], and $\Sigma^{@}$-algebra is based on one in [20]. However, for initial algebra semantics, minimality (w.r.t. subalgebra relation) is required in [19], and reachability is required in [20]. On the other hand, our algebra does not require any such restrictions. It is a future subject to study why such a difference comes out. Moreover, since our algebra is designed independently of type structure, it is so easy to combine with arbitrary type systems. It is important to incorporate not only simply-typed systems but also complicated ones like polymorphic-typed systems.

We would feel that our proving and disproving methods for primitive inductive theorems (Theorem 5.2, 5.7 and 6.3) were natural extensions of results in [16], [23]. Since these approach can apply only for the class of primitive inductive theorems, it is a critical problem for disproving inductive theorems. To overcome the difficulty, we presented a sufficient condition which guarantees that inductive theorems and primitive ones coincide (Theorem 4.17). The sufficient condition is indispensable for disproving inductive theorems (Theorem 6.3).

A higher-order Knuth-Bendix procedure and its application to inductionless induction were implemented by the first author Kusakari, as a post-doctorate sub-theme at the Japan Advanced Institute of Science and Technology (JAIST) in 1999. Example 5.4, which presents an application to inductionless induction (Theorem 5.2), was presented at that time. Results similar to these with respect to inductionless induction were also presented in 2003 by Aoto, Yamada and Toyama [1]. Unfortunately, all of the above results confused the difference between inductive theorems and primitive inductive theorems, and erroneously used the term "inductive theorems". In this paper, we make a clear distinction between the concepts of inductive theorems and primitive inductive ones. At the request of Toyama, Kusakari presented a lecture on the results in this paper to Aoto and Yamada in August 2003. In the following

year, Aoto, Yamada and Toyama presented results on the automated proving of inductive theorems [2] using a formalization different from our results.

Acknowledgments

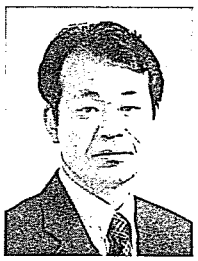
We would like to thank TOYAMA Yoshihito for fruitful discussion, and the anonymous referees for their useful and detailed comments.

This work is partly supported by MEXT. KAKENHI #15500007 and #16300005, by Artificial Intelligence Research Promotion Foundation, and by the 21st Century COE Program (Intelligent Media Integration for Social Infrastructure), Nagoya University.

References

[1] T. Aoto, T. Yamada, and Y. Toyama, "Proving inductive theorems of higher-order functional programs," Proc. Forum on Information Technology 2003 (FIT2003), Information Technology Letters, vol.2, pp.21–22, 2003.
[2] T. Aoto, T. Yamada, and Y. Toyama, "Inductive theorems for higher-order rewriting," Proc. 15th Int. Conf. on Rewriting Techniques and Applications (RTA 2004), LNCS 3091, pp.269–284, 2004.
[3] T. Nipkow, "Higher-order critical pairs," Proc. 6th IEEE Symp. Logic in Computer Science, pp.342–349, IEEE Computer Society Press, 1991.
[4] F. Baader and T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.
[5] G. Birkhoff, "On the structure of abstract algebras," Proc. Cambridge Philosophical Society, vol.31, pp.433–454, 1935.
[6] A. Bouhoula, Automated Theorem Proving by Test Set Induction, J. Symbolic Computation, vol.23, pp.47–77, 1997.
[7] R.S. Boyer and J.S. Moore, A Computational Logic, Academic Press, NewYork, 1979.
[8] R. Diaconescu and K. Futatsugi, CafeOBJ Report, AMAST Series in Computing 6, World Scientific, 1998.
[9] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.P. Jouannaud, "Introducing OBJ," in Software Engineering with OBJ: Algebraic specification in action, ed. Goguen and Malcolm, Kluwer, 2000.
[10] J. Goguen, Theorem Proving and Algebra, MIT Press, to appear.
[11] G. Huet and J.M. Hullot, "Roof by induction in equational theories with constructors," J. Comput. Syst. Sci., vol.25, pp.239–266, 1982.
[12] Y. Inagaki and T. Sakabe, "Fundamentals of algebraic specification of abstract data types (1): Many sorted algebras and equational logic," Inf. Process., vol.25, no.1, pp.47–53, 1984.
[13] Y. Inagaki and T. Sakabe, "Fundamentals of algebraic specification of abstract data types (2): Abstract data types," Inf. Process., vol.25, no.5, pp.491–501, 1984.
[14] Y. Inagaki and T. Sakabe, "Fundamentals of algebraic specification of abstract data types (3): Models of abstract type constructors and initial algebra semantics," Inf. Process., vol.25, no.7, pp.708–716, 1984.
[15] D. Kapur, P. Narendran, and H. Zhang, "Automating inductionless induction using test set," J. Symbolic Computation, vol.11, pp.83–111, 1991.
[16] H. Koike and Y. Toyama, "Comparison between inductionless induction and rewriting induction," JSSST Computer Software, vol.17, no.6, pp.1–12, 2000.
[17] K. Kusakari, "On proving termination of term rewriting systems with higher-order variables," Trans. IPSJ Programming, vol.42, no.SIG 7 (PRO 11), pp.35–45, 2001.
[18] K. Kusakari, "Higher-order path orders based on computability," IEICE Trans. Inf. & Syst., vol.E87-D, no.2, pp.352–359, Feb. 2004.
[19] K. Meinke, "Proof theory of higher-order equations: Conservativity, normal forms and term rewriting," J. Comput. Syst. Sci., vol.67, pp.127–173, 2003.
[20] B. Möller, A. Tarlecki, and M. Wirsing, "Algebraic specifications of reachable higher-order algebras," Proc. 5th Workshop on Specification of Abstract Data Types, LNCS 332, pp.154–169, 1987.
[21] D.R. Musser, "On proving induction properties of abstract data types," Proc. 7th ACM Symp. Principles of Programming Languages, pp.154–162, 1980.
[22] U.S. Reddy, "Term rewriting induction," Proc. 10th International Conference on Automated Deduction, LNAI 449 (CADE 90), pp.162–177, 1990.
[23] Y. Toyama, "How to prove equivalence of term rewriting systems without induction," Theor. Comput. Sci., vol.90, no.2, pp.369–390, 1991.
[24] M. Wirsing, "Algebraic specification, formal models and semantics," vol.B of Handbook of Theoretical Computer Science, chapter 13, pp.676–788, Elsevier Science, 1990.

Keiichirou Kusakari received B.E. from Tokyo Institute of Technology in 1994, M.E. and D.E. from Japan Advanced Institute of Science and Technology (JAIST) in 1996 and 2000. From 2000 to 2003, he had been a research associate of Research Institute of Electrical Communication (RIEC), Tohoku University. He is currently an assistant professor of Graduate School of Information Science, Nagoya University. His research interests include term rewriting systems, program theory, and automated theorem proving. He is a member of IPSJ and JSSST.



Masahiko Sakai completed graduate course of Nagoya University in 1989 and became Assistant Professor, where he obtained a D.E. degree in 1992. From April 1993 to March 1997, he was Associate Professor in JAIST, Hokuriku. In 1996 he stayed at SUNY at Stony Brook for six months as Visiting Research Professor. From April 1997, he was Associate Professor in Nagoya University. Since December 2002, he has been Professor. He is interested in term rewriting system, verification of specification and software generation. He received the Best Paper Award from IEICE in 1992. He is a member of JSSST.

Toshiki Sakabe received B.E., M.E. and D.E. degrees from Nagoya University in 1972, 1974 and 1978, respectively. He was a research associate at Nagoya University during 1977–1985, and an associate professor at Mie University and Nagoya University during 1985–1987 and 1987–1993, respectively. He has been a professor of Nagoya University since 1993. His research interests are in the field of theoretical foundations of software including algebraic specifications, rewriting computation, object-oriented computation and so on. He is a member of IPSJ, JSAI, JSSST and EATCS.