Proof Transformation via Interpretation Functions[1]

Piotr Kosiuczenko

Department of Computer Science, University of Leicester

Abstract: Redesign of a class structure requires transformation of the corresponding specification. In a previous paper we have shown how to use, what we call, interpretation functions for transformation of constraints. In this paper we study the way proof are transformed via such functions. We provide a sufficient condition for interpretation functions to preserve proofs using propositional logic with modus ponens, proofs using resolution rule and induction.

1 Introduction

Change is a constant factor in software engineering due to several reasons such as new technology enablers, pattern application, change of customer requirements etc. Design and refactoring patterns are widely used to improve and modify software systems. They help to enhance code maintainability, to increase its performance, to adjust the underlying logical structure to new or changed customer needs and so on. On the other hand, contractual specification proved to be very useful in specification of functional requirements. Nevertheless, research on the impact of refactoring patterns on contractual specification has been underdeveloped. Moreover, contractual specification is still not as widely used in software engineering practice as it could be. One of the reasons is that after any change to the class structure the constraints have to be updated manually, which is very laborious and error prone.

Morphisms of entailment systems have been studied for long time. It is worth of doing even for sake of relating different logics. This paper is motivated by practical applications of a formal approach to an automatic transformation of OCL class invariants, pre- and post-conditions [9]. We have presented this concepts first in [7]. Interpretation function is aimed at comparing equationally defined classes of one sort algebras; the idea is to relate structures with the same behavior but possibly different signatures. This paper deals with the problem of mapping terms of order sorted signature and corresponding proofs. We consider here a simple version of order sorted algebras as introduced by Goguen and Meseguer (see [4]). The notion of order sorted algebras have been extended to so called membership equational logic [2], which is more powerful and flexible. We prove several useful facts concerning proof transformation and preservation of entailment relation by such mappings. We define the notion of compositional function; i.e. a function, which satisfies cer-

1. This an extended and corrected version of the technical report number 2004/27, Department of Computer Science, University of Leicester.

tain compositionality criteria. We provide extendibility criteria allowing extension of mappings from a set of terms to all the terms obtained by substitution. This criteria corresponds to orthogonality in term rewriting systems (cf. [11]). Orthogonal term rewriting systems are, in a sense, very regular and possess several useful properties such as for example confluence. Therefore it is not surprising that functions satisfying similar conditions have many useful properties as well. We call such functions, interpretation functions. We prove that interpretation functions can be extended to most general unifiers. We show also how to deal with equational proofs and relate this formalism to Birkhoff calculus [1]. This is an interesting fact, because equational theories are the base of algebraic specifications approach, which is widely used for defining formal semantics and specifying software systems [12]. Since we assume that compositional functions preserve boolean operations, they preserve propositional tautologies, and deduction rules like modus ponens. Moreover they preserve proofs using resolution rule.

Interpretation functions have several applications in software engineering. They can be used to automatically transform OCL constraints when class diagrams are redesigned [7]. The fact that interpretation functions preserve proofs implies that application of various refactoring patterns [5] and redesign patterns [3] preserve logical consequences. It is a very useful fact, because it means that it is not necessary to redo proofs after redesign of class diagrams. For example one can relay that an invariant implies a pre-condition or that a post-condition implies an invariant, if it was the case in a specification prior to transformation.

2 Formal background

In this section we define the basic notions used in our paper. We consider here a simple version of order sorted algebras introduced by Goguen and Meseguer (see [4]); we define the notion of signature, order sorted signature, term, sort and order sorted algebra. In our case, each function symbol has exactly one type, whereas in the general case, one function symbol can have several types. We assume that there is the greatest sort. Actually [4] makes a weaker assumption that strongly connected components have a greatest sort, but we use this formal framework to formalize OCL which has the greatest element OclAny [9].

An *algebraic signature* is a pair (S, F) where S is a finite set of sorts and F is a set of typed function symbols, e.g. $f : s_1,...,s_n \rightarrow s \in F$, for some $s_1,...,s_n, s \in S$. An *order sorted signature* Σ is a triple $(S, F, \leq)$, where (S, F) is an algebraic signature and $\leq$ is a partial order on S (cf. [4]). For simplicity we assume that S forms a *tree*, or equivalently that one sort cannot be a subsort of two incomparable sorts: if $s \leq s_1$ and $s \leq s_2$, then $s_1 \leq s_2$ or $s_2 \leq s_1$. This corresponds to the assumption that there is no multiple inheritance. We also assume that S includes largest sort; i.e. there is an $s_l \in S$, such that for every $s \in S$, $s \leq s_l$.

We call a total function *mapping*. Let X be a set, let S be a set of sorts, and let $\tau : X \rightarrow S$ be a mapping. We call the elements of X variables. We say that a variable $x \in X$ is of type/sort s iff $\tau(x) = s$. When the typing function τ is clear, we write $x : s$ instead of $\tau(x) = s$. Below we always assume that there are infinitely many variables of every sort.

The notion of term and term type is defined inductively: If $f : s_1,...,s_n \to s$, $t_1 : u_1,..., t_n : u_n$ and $u_i \leq s_i$, then $f(t_1,..., t_n) : s$; we write also $\tau(f(t_1,..., t_n)) = s$. The set of all terms with variables in X is denoted by $T(\Sigma, X, \tau)$. For a term t, var(t) is the set of variables contained in t. The expression $t(x_1,..., x_n)$ means that t contains at most the variables $x_1,..., x_n$; i.e. $var(t) \subseteq \{x_1,..., x_n\}$. We call a mapping $\sigma : X \to T(\Sigma, X, \tau)$ *substitution*, if for every variable x : s, the type of $\sigma(x)$ is a subtype of s. We often write x^σ instead of $\sigma(x)$. The term $t[t_1/x_1,..., t_n/x_n]$ is obtained from t by applying the substitution $[t_1/x_1,..., t_n/x_n]$, which maps variable x_i to t_i, for i = 1,..., n, and leaves other variables of t unchanged. We call this operation *term composition*. We say that a substitution σ *preserves types*, if for every variable $\tau(x) = s$ implies that $\tau(\sigma(x)) = s$. We call a substitution *variable renaming*, if it substitutes variables for variables and preserves types.

Let A be a set of terms. Let Y be a set of variables such that for every $u \in A$, $var(u) \subseteq Y$, and let Y be closed on variables renaming; i.e. for every $x \in Y$, if x is of sort s and if y is a variable of sort s as well, then $y \in Y$. gen(A, Y) is the smallest set of terms satisfying the following conditions:

- $A, Y \subseteq gen(A, Y)$.
- If $y \in Y$, $u, v \in gen(A, Y)$, and [v/y] is well defined, then $u[v/y] \in gen(A,Y)$.

In other words, gen(A, Y) is the smallest set containing A and Y, closed on variable renaming and substitution. We call gen(A, Y) *term space*. For example, if Σ has the form (S, F, $\leq$), then the set $gen(\{f(x_1,..., x_n) \mid f : s_1,...,s_n \to s \in F, \{x_1,..., x_n\} \subseteq X\}, X) = T(\Sigma, X, \tau)$. We say that a term space gen(A, Y) *has a top*, if Y include a variable y of the largest sort s_l.

For a partial function g, Dom(g) denotes the *domain* of g; i.e. the set of all elements for which g is defined. Rg(g) denotes the *range* of g; i.e. the set of all values of g. Let $A = (A_s)_{s\in S}$, $B = (B_s)_{s\in S}$ be two S-sorted sets, a function $g : A \to B$ is an S-indexed family of functions $g_s : A_s \to B_s$, for $s \in S$. An *order-sorted algebra* [4] over a signature Σ has the form $A = ((A_s)_{s\in S}, (f^A)_{f\in F})$; it consists of a family of non empty carrier sets $(A_s)_{s\in S}$ such that $A_u \subseteq A_s$, for $u \leq s$, and a family of functions $(f^A)_{f:s1,...,sn\to s\in F}$ such that $f : s_1,...,s_n \to s^A$: $A_{s_1} \times ... \times A_{s_n} \to A_s$. A *specification* is a pair Spec(Σ, Ax) consisting of a signature Σ and a set of formulas Ax over the signature Σ.

3 Interpretation functions

In this section we define the notion of interpretation function. We define the notion of compositional function. We introduce the notion of orthogonal mapping, which is analogous to the notion of orthogonal term rewriting systems (cf. [11]). Interpretation functions are compositional functions generated by orthogonal mappings. We provide a sufficient condition guarantying existence of interpretation functions. Applicability of a term rewriting rule does not depend on a particular names of variables in the redendum and in the redex, therefore we formulate the definitions below independently of variable names.

3.1 Definition

We say that two terms u, v *overlap*, if there exist two variable renamings σ_1 and σ_2 such that any of the following conditions is satisfied:

- u and v are different and u^{σ_1} is unifiable with v^{σ_2}.
- u^{σ_1} is unifiable with a non-variable proper subterm of v^{σ_2}.
- v^{σ_2} is unifiable with a non-variable proper subterm of u^{σ_1}.

We say, that a set of terms is *overlapping free*, if it neither contains variables nor terms which overlap. We say that a term t is *linear*, if for every variable x, t contains x at most once. A set of terms A is *orthogonal*, if A is overlapping free and the terms in A are linear.

The notion of overlapping free sets corresponds to the definition of orthogonal term rewriting systems. In the case of one sorted signatures, a term rewriting system is orthogonal if there are no critical pairs and the terms on the left hand side are linear. In fact, orthogonality of a term rewriting system depends only on the left hand side of the corresponding term rewriting rules.

Let $A \subseteq T(S, F, \leq, X, \tau)$ be a set of terms. A mapping $\varphi : A \to T(S', F', \leq', X', \tau')$ is *orthogonal*, if there exists a partial function on sorts $\rho : S \to S'$ such that the following conditions are satisfied:

a) For every variable x, $\rho(\tau(x))$ is defined iff $x \in X \cap X'$.

b) If $x \in X \cap X'$, then $\rho(\tau(x)) = \tau'(x)$.

c) $var(\varphi(v)) \subseteq var(v)$, for $v \in A$.

d) If $v \in A$, then $\rho(\tau(v))$ is defined and $\rho(\tau(v)) = \tau'(\varphi(v))$.

e) If $\rho(s_1)$, $\rho(s_2)$ are defined and $s_1 \leq s_2$, then $\rho(s_1) \leq' \rho(s_2)$.

f) A (i.e. $Dom(\varphi)$) is orthogonal.

Conditions (a) and (b) say that the sort mapping ρ is determined by types of common variables. Condition (c) eliminates some pathological cases. Condition (d) says that ρ commutes with φ in respect to types. Condition (e) says that ρ is monotone.

Let us observe that an orthogonal mapping $\varphi : A \to T(\Sigma, X, \tau)$ defines an orthogonal term rewriting system, if $A \subseteq T(\Sigma, X, \tau)$, and $\tau(t) = \tau(\varphi(t))$, for every term $t \in A$.

Let $\psi : T(\Sigma, X, \tau) \to T(\Sigma', X', \tau')$ be a partial function. ψ is *compositional* iff for all terms t the following conditions hold:

i) $\psi(x)$ is defined for every variable $x \in X \cap X'$.

ii) If $\psi(t)$ is defined, σ is a variable renaming, then $\psi(t^\sigma) = \psi(t)^\sigma$.

iii) $var(\psi(t)) \subseteq var(t)$, if $\psi(t)$ is defined.

iv) If ψ maps term t_i to the term t_i', for i = 0,..., n, x_1,..., $x_n \in X \cap X'$ and t has the form $t_0[t_1/x_1,..., t_n/x_n]$, then the substitution $t_0'[t_1'/x_1,..., t_n'/x_n]$ is a well defined and $\psi(t)$ has the form $t_0'[t_1'/x_1,..., t_n'/x_n]$.

Conditions (i) and (ii) are analogous to conditions (a), (b) and (c), respectively. Condition (ii) implies that compositional functions do not depend on variables' names. (iv) is a compositionality condition; it allows us to scale up a mapping to complex terms; this condition can be equivalently expressed by requiring that $\tau(t_1) \leq \tau(x_1),..., \tau(t_n) \leq \tau(x_n)$ imply that $\tau'(t_1') \leq' \tau'(x_1),..., \tau'(t_n') \leq' \tau'(x_n)$, for $x_1,..., x_n \in X \cap X'$. This condition is a kind of monotonicity preservation requirement. Let us observe that if $\psi(x)$ is defined, then $\psi(x) = x$. Indeed, if $\psi(x) = t(x)$, then x[x/x] is mapped on t[t(x)/x] = t(t(x)). Since ψ is a function, t(t(x)) must be syntactically identical with t(x) and consequently with x.

Compositional functions determine sort mappings. Moreover, a composition of compositional functions is a compositional function, if there is no conflict between variable names.

3.2 Statement

Let $\psi : T(S, F, X, \tau) \rightarrow T(S', F', X', \tau')$ be a compositional function, then there exist a sort monotone function $\rho : S \rightarrow S'$, such that $\rho(\tau(x)) = \tau'(x)$, for $x \in X \cap X'$.

Let $\psi : T(\Sigma, X, \tau) \rightarrow T(\Sigma', X', \tau')$ and $\zeta : T(\Sigma', X', \tau') \rightarrow T(\Sigma'', X'', \tau'')$ be compositional functions. If $X \cap X'' \subseteq X \cap X' \cap X''$, then $\psi\zeta$ is compositional.

Proof

We define $\rho(s) = s'$, if there is a variable $x \in X \cap X'$, such that $\tau(x) = s$ and $\tau'(x) = s'$. Let us observe that for all variables x and y, if x can be substituted for y, then $\tau(x) \leq \tau(y)$. From condition (iv) follows that for all x, $y \in X \cap X'$, if $\tau(x) \leq \tau(y)$, then $\tau'(x) \leq' \tau'(y)$. Consequently ρ is well defined, since if two variables x, $y \in X \cap X'$ are of the same sort, then one can be substituted for the other, $\tau(x) = \tau(y)$, and consequently $\tau'(x) = \tau'(y)$. From the property mentioned above follows also that ρ is sort monotone (see condition e) above).

Condition (i) follows from the assumption that $X \cap X'' \subseteq X \cap X' \cap X''$. Conditions (ii), (iii) and (iv) follow directly from the definition. ♦

A compositional function can be extended to equations:

$\psi(x = y) =_{def} \psi(x) = \psi(y)$.

Propositional formulas can be formalized by boolean terms. We can define compositional function on those terms as follows:

If $\psi(\Phi) = \Phi'$ and $\psi(\Psi) = \Psi'$, then $\psi(\Phi \wedge \Psi) =_{def} \Phi' \wedge \Psi'$,

similarly for other propositional operations. We can extend this definition to quantified formulas:

If $\psi(\Phi) = \Phi'$, then $\psi(\forall_x \Phi) =_{def} \forall_x \Phi'$, for $x \in X \cap X'$, and similarly for the existential quantifier.

The following theorem allows us to extend orthogonal mappings to compositional functions. Let us point out that this theorem holds also in the case when S does not form a tree.

3.3 Theorem

Let $A \subseteq T(\Sigma, X, \tau)$ be a set of terms and let $\varphi : A \rightarrow T(\Sigma', X', \tau')$ be an orthogonal mapping. Then the mapping φ can be uniquely extended to a compositional function

$$\psi : T(\Sigma, X, \tau) \rightarrow T(\Sigma', X', \tau')$$

such that $Dom(\psi) = gen(A, X \cap X')$.

Proof

Let the assumption of this theorem be satisfied. First, we extend φ to a relation ψ defined on the set $gen(A, X \cap X')$. Let ψ_0 be the smallest relation[2] such that for every variable renaming σ, if $u \varphi v$ holds, then $u^\sigma \psi_0 v^\sigma$ holds as well. We define ψ to be the smallest relation satisfying the following conditions:

- ψ contains ψ_0.
- ψ contains all pairs (x, x), such that $x \in X \cap X'$.
- If ψ relates term t_i to t_i', for $i = 1,..., n$, $x_1,..., x_n \in X \cap X'$, $\psi_0(t_0) = t_0'$, the substitutions $[t_1/x_1,..., t_n/x_n]$ and $[t_1'/x_1,..., t_n'/x_n]$ are well defined, then ψ relates $t_0[t_1/x_1,..., t_n/x_n]$ to $t_0'[t_1'/x_1,..., t_n'/x_n]$.

One can easily proof by structural induction that if $u \psi v$, then $var(u) \subseteq var(v)$ (see condition (c) of the definition of orthogonal mappings). First, we prove that ψ is a partial function which maps variables on variables. Let us assume that it is not the case; i.e. that ψ relates a term t to two different terms: $t \psi u$ and $t \psi v$. Let t be a minimal term in respect to high, which has this property.

Composing any term with nonvariable term results in a nonvariable term. Similarly, application of variable renaming to a nonvariable term results in a nonvariable term. Application of variable renaming to a variable results in a variable. Therefore a variable can be related by ψ only to itself. Consequently, t cannot be a variable.

From the definition of ψ and the fact that t cannot be a variable follows that there exist terms $u_0, v_0 \in A$, such that t can be presented in the form $u_0[u_1/x_1,..., u_m/x_m]$ and in the form $v_0[v_1/y_1,..., v_n/x_n]$, and that $u_0[u_1/x_1,..., u_m/x_m] \psi_0 u$ and $v_0[v_1/y_1,..., v_n/y_n] \psi_0 v$.

Since A is orthogonal and since u_0 and v_0 are unifiable, they must be identical. Consequently, we can assume that after suitable index permutation u_i is identical with v_i. Since u_0 and v_0 are mapped on the same term by φ and since t has two different images, there

2. In set theory, relations and in particular functions are sets. The smallest relation means a relation included in all other relations.

exist an i such that u_i (or equivalently v_i) has two different images in respect to ψ. But this contradicts the assumption that t is a minimal term related to two different terms.

To prove the compositionality property we have to show that if ψ maps t_i to t_i' and $x_i \in X \cap X'$, for i = 1,..., n, the term $t_0[t_1/x_1,..., t_n/x_n]$ is well defined and $t_1,..., t_n \in gen(A, X \cap X')$, then the substitution $[t_1'/x_1,..., t_n'/x_n]$ is well defined as well; i.e. $\tau'(\psi(t_i)) \leq' \tau'(x_i)$ for i = 1,..., n. Let us observe that in general it is enough to prove that if ψ is defined on the term t, $x \in X \cap X'$, and $\tau(t) \leq \tau(x)$, then $\tau'(\psi(t)) \leq' \tau'(x)$.

Let t be a variable y. Then $\tau(y) \leq \tau(x)$, $\psi(y) = y$, and $\tau'(\psi(y)) = \tau'(y) = \rho(\tau(y)) \leq' \rho(\tau(x)) = \tau'(x)$ (due to the condition (e)).

Let us assume that the compositionality property is possessed by terms $v_1,..., v_m$. Let $x_1,..., x_m \in X \cap X'$ and let the term t have the form $v_0[v_1/x_1,..., v_m/x_m]$, for $v_0 \in A$. The type of a term is equal to the type of its top; i.e. $\tau(t) = \tau(v_0)$.

Let $\{x_1,..., x_m\} = var(v_0)$. Either $\varphi(v_0)$ is a variable or not. If it is not a variable, then $\tau'(\psi(t)) = \tau'(\varphi(v_0))$ and

$$\tau'(\psi(t)) = \tau'(\varphi(v_0)[\psi(v_1/x_1),..., \psi(v_m/x_m)]) = \tau'(\varphi(v_0)) = \rho(\tau(v_0)) \leq' \rho(\tau(x)) = \tau'(x).$$

If it is a variable, then due to the fact that $var(\varphi(v_0)) \subseteq var(v_0)$ (see the condition (c)), $\varphi(v_0)$ has the form x_j. t is well defined; therefore $\tau(v_j) \leq \tau(x_j)$. Due to the inductive assumption $\tau'(\psi(v_j)) \leq \tau'(x_j)$ holds. Consequently due to the inductive assumption and conditions (b), (d) and (e) the following holds:

$$\tau'(\psi(t)) = \tau'(\varphi(v_0)[\psi(v_1/x_1),..., \psi(v_m/x_m)]) = \tau'(\psi(v_j)) \leq' \tau'(x_j) = \tau'(\varphi(v_0)) = \rho(\tau(v_0)) \leq \rho(\tau(x)) = \tau'(x)$$

The domain of ψ is equal to $gen(A, X \cap X')$, since the domain is the smallest set which contains A as well as $X \cap X'$, and which is closed on term composition.

The proof of uniqueness follows by structural induction: Let θ be another compositional function extending φ. θ and ψ must coincide on variables from $X \cap X'$ and on terms from A. Let us assume that those functions coincide on terms r, $t_1,..., t_n$. In turn, they must coincide on the term $r[t_1/x_1,..., t_n/x_n]$ due to compositionality. ♦

A compositional function is an interpretation function, if it is determined by an orthogonal mapping; formally, we call a compositional function ψ *interpretation function*, if there exists an orthogonal mapping φ such that $Dom(\psi) = gen(Dom(\varphi), X \cap X')$, and for every $t \in Dom(\varphi)$, $\varphi(t) = \psi(t)$. In that case, we say also hat ψ is *generated by* φ.

Every interpretation function, in the sense of abstract algebra [10], can be defined as a total compositional function, which uniquely extends a mapping on the set

$$\{f(x_1,..., x_n) \mid f : s_1,...,s_n \rightarrow s \in F\}.$$

In the following sections we show that interpretation functions have several useful properties.

4 Coloring terms

In the theory of term rewriting, orthogonal term rewriting systems play the central role (cf. e.g. [11]). They have several useful properties such as for example confluence. The concept of orthogonal mapping is analogous to the concept of orthogonal term rewriting systems; therefore it is not surprising that sets generated by orthogonal sets are in a sense regular and have many useful properties. In this section we show some useful properties of orthogonal sets. We prove for example that most general unifiers preserve such sets.

To reason about terms, one needs precise notions allowing for an accurate description of term structure. A *position* is a sequence of natural numbers $(i_1,..., i_n)$. For two positions $p_1 = (i_1,..., i_m)$ and $p_2 = (j_1,..., j_n)$, p_1p_2 denotes the *concatenation*; i.e.

$p_1p_2 =_{def} (i_1,..., i_m, j_1,..., j_n)$.

Let p be a position and let P be a set of positions, we define $pP =_{def} \{pq \mid q \in P\}$. Let t be a term; the *term at position* () in t is t itself (i.e. () is the *top position*). If u occurs in t at position $(i_1,..., i_m)$, u is of the form $f(t_1,..., t_n)$ and $1 \le k \le n$, then the term at the position $(i_1,..., i_m, k)$ is equal to t_k. For a position p, we define the projection function π_p:

$\pi_p(t) =_{def}$ u if and only if u is a subterm of t, which occurs at position p.

For a term t, Pos(t) denotes positions of nonvariable subterms of t.

By $u_p[v]$ we mean a term of the form u[v/x], where x occurs in u at position p and only at p. We call u_p *context* (cf. [11]).

Let us consider a term space gen(A, Y) and let A be an orthogonal set. For every term t, we can in a sense color t using the terms from A. Let p be a position in t, let $w \in A$ and let $p_1 \in Pos(w)$. If $\pi_p(t) = w^\sigma$ for a substitution σ, then we say that the *position* pp_1 *is colored with* w (see Fig. 1). We exclude the positions of variables from a domain of a color, since they are only placeholders for other terms.

A term t is *covered*, if $t \in gen(A, Y)$. A *equation* u = v is *covered*, if u and v are covered. A *sequence of terms is covered*, if every term in the sequence is covered.

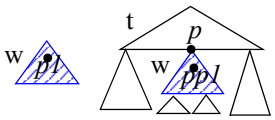

Fig. 1. Positions and coloring

The following lemma says that for an orthogonal set A, the coloring with A is unique; i.e. for an arbitrary term t and for any position p in this term there is at most one color coloring

p and moreover this position is colored only once by the same color. It says also that the property of being covered is preserved by term decomposition.

4.1 Lemma

Let A be an orthogonal, let gen(A, Y) be a term space and let t be an arbitrary term.

- If there exist two positions p_1, p_2 and two colors u, v ∈ A, such that $\pi_{p1}(t) = u^{\sigma 1}$, $\pi_{p2}(t) = v^{\sigma 2}$ and the set of positions $p_1Pos(u) \cap p_2Pos(v)$ is not empty, then $p_1 = p_2$ and u ≡ v.
- Let t have the form $u[u_1/x_1,..., u_n/x_n]$, for an u ∈ A such that $var(u) = \{x_1,..., x_n\}$, then t is covered iff and only if the terms $u_1,..., u_n$ are covered.

Proof

The first part of the lemma follows from orthogonality of A.

Let us observe that if $u_1,..., u_n$ are covered, then t is covered as well.

Let us also observe that every nonvariable term r in gen(A, Y) can be represented in the form $v[v_1/y_1,..., v_m/y_m]$, for a term v ∈ A, and $v_1,..., v_m$ ∈ gen(A, Y). If t has the form $v[v_1/y_1,..., v_m/y_m]$, for a term v ∈ A, $v_1,..., v_m$ ∈ gen(A, Y), then due to orthogonality u is identical with v and u_i must be identical with v_j after a suitable index permutation. Consequently, the terms u_i are covered. ♦

A substitution σ is a *unifier* of u and v, if u^σ is identical with v^σ. σ is a *most general unifier* (mgu), if every other unifier ζ can be presented in the form σκ, for a substitution κ. The following lemma says that most general unifiers preserve the property of being covered.

4.2 Lemma

Let A be an orthogonal set and let gen(A, Y) be a term space which has a top. Let u, v be covered terms. If there is a unifier σ of u and v, then there is a most general unifier of u and v. If σ is a most general unifier of u and v, then u^σ ∈ gen(A, Y) and σ substitutes covered terms for variables in u and v.

Proof

In the case of single sorted algebras, if a most general unifier exists, then it can be computed using Martelli-Montanari algorithm (MMA) [8]. Every step in this algorithm can be seen as a rule of the form G → G´, where G, G´ are sets of equations. This algorithm includes the following rules:

a) Decomposition

$\{f(u_1,..., u_n) = f(v_1,..., v_n)\} \cup G \rightarrow \{u_1 = v_1,..., u_n = v_n\} \cup G$

b) Elimination of trivial equations

$\{x = x\} \cup G \rightarrow G$

c) Swap: If t is not a variable, then

$\{t = x\} \cup G \rightarrow \{x = t\} \cup G$

d) Substitution

$\{x = t\} \cup G \rightarrow \{x = t\} \cup G\ [t/x]$ (i.e. we substitute t for x in all terms in G)

Plus two termination rules for the case when no mgu exists.

We need to modify rules of this algorithm in such a way that the modified algorithm applies to order sorted algebras and the rules preserve the property of being covered. I.e. if $\{u_1 = v_1,..., u_n = v_n\}$ is the set of equations obtained, then u_i, v_i are covered terms for every i. Only the decomposition rule may violate this property. On the other hand, the resulting set should contain well formed equations. The MMA substitution rule may be incorrect; if the type of t is not smaller than or equal to the type of x, then we cannot substitute t for x.

These steps of MMA are repeated as long as there is a rule, which modifies G. This algorithm finds a mgu independently of the order in which those rules are applied. Consequently, the term decomposition rule can be applied an arbitrary number of times. Due to the fact that the terms in A are linear, we can replace the decomposition rule by a new rule which can be seen as several applications of the decomposition rule:

If $r \in A$ and $var(r) = \{x_1,..., x_n\}$, then

$\{r[r_1/x_1,..., r_n/x_n] = r[t_1/x_1,..., t_n/x_n]\} \cup G \rightarrow \{r_1 = t_1,..., r_n = t_n\} \cup G$

We can also modify the substitution rule as follows:

If $\tau(t) \leq \tau(x)$, $\{x = t\} \cup G \rightarrow \{x = t\} \cup G\ [t/x]$

First, we apply the modified algorithm to u and v. As in the case of MMA, at every stage of this algorithm execution, if E is the set of produced equations, then a substitution η is a unifier E iff η unifies u and v. This follows from the fact that for a unifier η, that property holds for all stages in the MMA application, when we abstract from typing, and from the fact that steps of the modified algorithm can be simulated by the MMA algorithm. We call this property *unifier-invariancy*.

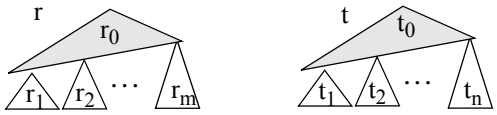

Fig. 2. Decomposition of terms

Let us assume that $\{r = t\} \cup G$ is a set of covered unifiable equations. Let neither r nor t be a variable. Since both terms are unifiable and belong to gen(A, Y) and since the terms in A are linear, we can present them in the form: $r \equiv r_0[r_1/x_1,..., r_n/x_m]$, $t \equiv t_0[t_1/y_1,..., t_n/y_n]$, where r_0 and t_0 belong to A and where terms $r_1,..., r_m$, $t_1,..., t_n$ are covered (see lemma 4.1 and Fig. 2). The fact that A is orthogonal and the fact that r and t are unifiable imply that

$r_0 \equiv t_0$ and we can assume that m = n and that x_i is identical with y_i. Now we can replace in one step $\{r = t\} \cup G$, by $\{r_1 = t_1,..., r_n = t_n\} \cup G$. This corresponds to decomposing terms r and t using the MMA decomposition rule until r_0 (or equivalently t_0) is fully decomposed. The resulting set $\{r_1 = t_1,..., r_n = t_n\} \cup G$ consists of covered equations.

Let us assume that there exists a unifier σ of u and v. Then MMA terminates and produces a mgu, if we neglect types. But this means that the modified algorithm terminates as well, since the modified substitution rule is just a weaker form of the original substitution rule and the decomposition rule compresses a number of applications of the original decomposition rule.

We can apply rules (b), (c), the modified decomposition and substitution rules as long as the set of equations is modified. Let H be the resulting set of equations. We show that MMA decomposition rule doesn't apply either. If this rule was applicable, then we would have a covered equation r = t in G, such that r and t are unifiable and neither r nor t is a variable. But then we could apply the modified decomposition rule again, a contradiction.

Let us observe that H does not contain equations of the form t = r where t is not a variable, since the swap rule and decomposition rule do not apply anymore. Consequently, the modified algorithm produces a covered set of equations H of the form x = y and x = t, where t is not a variable. We can extend those equations to an equivalence relation. Let us observe that equivalence classes of this relation are mapped by any unifier on the same term. Moreover, an equivalence class either contains variables only, or a nonvariable term and a number of variables, but it cannot contain two different nonvariable terms, as the rules, in particular substitution rule and the swap rule, are no more applicable. Let us assume the opposite, then H contains an equation of the form x = t, where t is not a variable, and an equation of the form x = r or the form r = x. But then we can substitute t for x in x = r or in r = x; this is due to the fact that σ(x) is identical with σ(t) (see the unifier-invariancy property) and to the fact that the type of t is identical with the type of σ(x), since t is a nonvariable term. However according to our assumption, no rule of the modified algorithm is applicable; in particular the substitution rule is not applicable.

Let $Y_1,..., Y_m$ be equivalence classes containing only variables. For every $x, y \in Y_i$, x^σ is identical with y^σ due to the unifier-invariancy property. Let us consider the set Y_i. Let S_i be the set of sorts of its variables; i.e. $S_i = \{s \mid$ there is a variable $x \in Y_i$ of sort $s\}$. Since there is no multiple inheritance, the set of sorts forms a tree. $\tau(\sigma(x)) \leq \tau(x)$, for any $x \in Y_i$, since substitution σ is well defined. In a tree every set which has a lower bound is linearly ordered. Consequently S_i is linearly ordered. For i = 1,..., m, let s_i be the smallest sort in S_i and let z_i be a fresh variable of this type. Let the substitution ζ be defined on variables of u and v as follows:

If x belongs to an equivalence class Y_i, then $\zeta(x) =_{def} z_i$, else $\zeta(x) =_{def} x$.

Let us define substitution κ on variables belonging to H as follows:

If x belongs to an equivalence class Y_i, then $\kappa(x) =_{def} \zeta(x)$, else there exist an equation of the form x = t in H and we define $\kappa(x) =_{def} t^{\zeta}$.

Let us observe that κ is a substitution, since for an equation x = t, where t is a nonvariable term, $\sigma(x) \equiv \sigma(t)$; consequently $\tau(\sigma(t)) = \tau(\sigma(x)) \leq \tau(x)$, since σ is a substitution. On the other hand, H is covered and ζ substitutes for variables from Y, therefore the substitution κ is well defined. It maps variables from Y to covered terms. (Let us remind that gen(A, Y) is closed on substituting covered terms for variables.)

For every equation x = t in H, where t is a nonvariable term, t^{κ} is identical with t^{ζ}. Let it be otherwise and let y be a variable of t, such that κ(y) is different from ζ(y). It means that there is an equation y = r in H, such that r is not a variable. However, this contradicts the assumption that the substitution rule does not modify H.

κ is covered, since it is obtained from the covered set H by substituting variables from Y for variables from Y. We show that κ is a mgu. It is a unifier, since it unifies Y_i, for i = 1,..., m, and if x = t belongs to H, then $(x = t)^{\kappa}$ is identical with $x^{\kappa} = t^{\kappa}$ and the terms x^{κ}, t^{κ} are identical with t^{ζ}. On the other hand, σ can be obtained from κ; i.e. $\sigma = \kappa\xi$, for a substitution ξ defined as follows: If x, y are variables from a set Y_i, then $\sigma(x) \equiv \sigma(y)$ and we can define ξ to map z_i to σ(x). Similarly, if x = t belongs to H, then σ(x) is identical with t^{σ}, and t^{σ} can be obtained from t^{ζ} by replacing z_i, by $\xi(z_i)$; i.e. $t^{\sigma} \equiv t^{\zeta\xi} \equiv t^{\kappa\xi}$. Since σ is an arbitrary substitution, κ is a mgu.

Let σ be a mgu. We have to show that σ is covered as well. But this follows from the fact that $\sigma = \kappa\xi$ and that there is a substitution ν, such that $\sigma\nu = \kappa$ and $\sigma\nu = \kappa\xi\nu = \kappa$. Moreover, ξν is an identity on variables of occurring in H. Therefore, σ and κ are equivalent modulo variable renaming. ♦

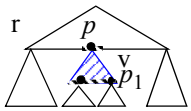

Fig. 3. Boundary positions in t

We call a position in a term r boundary if it corresponds to a position of a covered term u or a position corresponding to a variable of u. A maximal not covered subterm of a term is called red. More precisely:

4.3 Definition

Let A be an orthogonal set and let gen(A, X) be a term space. Let r be a term of the form $r \equiv u_p[v^{\sigma}]$, where u ∈ A (i.e. u occurs at position p in r). Let p_1 be a position of a variable in u. Then, p and pp_1 are called *boundary positions* in r in respect to A (see Fig. 3).

Let t be a term and let u be a non-variable subterm of t. We call a term u *red*, if u is neither a variable from Y nor can be presented in the form $r[r_1/x_1,..., r_n/x_n]$, for an $r \in A$. u is a *red subterm of t*, if u is a maximal subterm of t having the following properties:

- u occurs at a boundary position in t or $u \equiv t$.
- u is not of the form a^σ, for an $a \in A$.

We call a *position* p in t *red*, if p occurs in a red subterm; we call all other positions in t *green*. We call a term t *red*, if t is a red subterm of itself.

All suffixes of a red position are red (see Fig. 4). All prefixes of green positions are green. Let us observe that a position p is green iff p is a position of a non-variable term and p as well as all proper prefixes of p are colored.

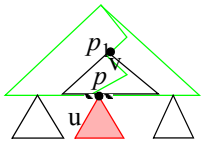

Fig. 4. Red subterms

Let gen(A, Y) be a term space. S_Y denotes the set of sorts of variables in Y; i.e. $S_Y = \{s \in S \mid \text{there is an } x \in Y \text{ such that } \tau(x) = s\}$. We define a set of fresh variables as follows: Let s be a sort and let s_1 be the smallest sort in S_Y such that $s \leq s_1$. Let us observe that such a sort exists since S is a tree in respect to $\leq$, $\{s_1 \in S_Y \mid s \leq s_1\}$ is linearly ordered, S_Y contains the largest sort and S is finite. For a term t : s, we define a new variable x_t of sort s_1. We use those fresh variables to define a function R on terms as follows:

- If t is a variable from Y then R(t) is equal t.
- If t has the form $r[r_1/x_1,..., r_n/x_n]$, for an $r \in A$ such that $var(r) = \{x_1,..., x_n\}$, then $R(r[r_1/x_1,..., r_n/x_n])$ has the form $r[R(r_1/x_1),..., R(r_n/x_n)]$.
- If t is neither a variable from Y nor has the above form (i.e. t is red), then R(t) is defined to be x_t.

This function replaces a red subterm r of t by the fresh variable x_r. Let us observe that for every term t, R(t) is a well defined term. This is due to the fact that for every $u \in A$ and every variable $x : s \in var(u)$, and for every term v, such that v can be substituted for x: $\tau(v) \leq \tau(x_v) \leq \tau(x)$. The last inequality is due to the fact that x belongs to Y and the sort of x_v is the smallest sort in S_Y, which is larger than or equal to $\tau(v)$.

Below to avoid unnecessary complexity of definitions, we will treat the fresh variables as elements of Y. In general, fresh variables are used in logic to avoid name conflicts. If the underlying set of variables is infinite, then it is always possible to find fresh variables. Let us notice that under this assumption, R(R(t)) is equal R(t), since the term R(t) is covered.

4.4 Lemma

Let gen(A, Y) be a term space, which has a top, and let A be orthogonal.

a) Let t be a term. Then there is a covered term u and a substitution σ such that t has the form u^σ and σ maps variables of u to red terms.

b) R is sort monotone: If $\tau(t) \leq \tau(u)$ then $\tau(R(t)) \leq \tau(R(u))$.

c) If the term t has the form u^η, for a term u and a substitution η, then R(t) has the form $R(u)^\gamma$, for a substitution γ.

Proof

(a) Let χ map every variable x_r on the term r. From the definition of variables x_r follows that $\tau(r) \leq \tau(x_r)$, and consequently χ is a substitution. The term t can be represented in the form $R(t)^\chi$.

(b) This property follows from the fact that S is a tree in respect to $\leq$. If $s_1 \leq s_2$, and a_i is the smallest sort belonging to S_Y such that $s_i \leq a_i$, for i = 1, 2, then $s_1 \leq a_1 \leq a_2$. The last inequality follows from the fact that a_1 is the smallest sort in S_Y, which is larger than s_1, $s_2 \leq a_2$ and a_2 belongs to S_Y.

(c) Let us observe that $\tau(t) \leq \tau(u)$. Indeed, if u is a variable, then $\eta(u) = t$ and $\tau(t) \leq \tau(u)$, since η is a substitution. If u is not a variable, then t is not a variable and $\tau(t) = \tau(u)$. From (b) follows that in both cases $\tau(R(t)) \leq \tau(R(u))$.

Let γ be defined on variables of R(u) as follows:

$\gamma(y) = R(y^\eta)$, for any variable y which is not of the form x_v.

$\gamma(x_v) = R(v^\eta)$, in the other case.

We have to show that $R(r)^\gamma = R(r^\eta)$ holds for every term r. From this property follows that in the case of u, $R(u)^\gamma = R(u^\eta) = R(t)$.

Let r be a variable, which is not of the form x_v. Then $R(r)^\gamma = r^\gamma = R(r^\eta)$, since in this case $R(r) = r$. Let r be a red term, then $R(r) = x_r$ and $x_r^\gamma = R(r^\eta)$. We have shown that γ has the required property for variables and red terms. Let γ have the required property in the case of terms $u_1,..., u_n$ and let r have the form $a[u_1/x_1,..., u_n/x_n]$, for a term $a \in A$ such that $var(a) = \{x_1,..., x_n\}$. Then $R(a[u_1/x_1,..., u_n/x_n])^\gamma$ can be equivalently represented as $a[R(u_1)^\gamma/x_1,..., R(u_n^\gamma)/x_n]$. From the inductive assumption follows that the second term can be presented in the form $a[R(u_1^\eta)/x_1,..., R(u_n^\eta)/x_n]$. From the definition of R follows that $a[R(u_1^\eta)/x_1,..., R(u_n^\eta)/x_n]$ can be represented as $R(a[u_1^\eta/x_1,..., u_n^\eta/x_n])$ and consequently as $R(a[u_1/x_1,..., u_n/x_n]^\eta)$. ♦

5 Proof transformation

In the previous section, we have investigated coloring of terms using orthogonal sets. In this section we use those results to transform proofs and to show that interpretation functions preserve proofs using propositional tautologies and resolution. The first theorem says that interpretation functions map most general unifiers on unifiers. The second theorem deals with the case of propositional reasoning and resolution rule. The third theorem deals with the case of induction. We also explain how to deal with equations, and relate our formal framework to Birkhoff's equational calculus (see [1]).

Since compositional functions preserve boolean operations, they preserve propositional tautologies as well as deduction rules such as modus ponens and resolution. Let us point out that results presented in this section hold under the assumption that the underlying set of sorts S in the function domain forms a finite tree. Moreover, we need to assume that the largest sort is mapped by the underlying sort mapping. This corresponds to the assumption that the underlying term space has a top; i.e. it includes variables of the largest sort.

5.1 Theorem

Let $\psi : T(S, F, \leq, X, \tau) \rightarrow T(S', F', \leq', X', \tau')$ be an interpretation function. Let the underlying sort mapping ρ be defined on the largest sort of S and let ψ be defined on an equation $u = v$. If σ is a most general unifier of $u = v$, then $\sigma\psi$ is a unifier of $\psi(u) = \psi(v)$.

Proof

Let ψ be generated by orthogonal mapping φ and let the assumption of this theorem be satisfied. We can assume that σ maps only variables occurring in those equations. From lemma 4.2 applied to $gen(Dom(\varphi), X \cap X')$ follows that $\psi(x^\sigma)$ is defined on variables occurring in E, since σ substitutes covered terms for variables. Moreover, due to compositionality of ψ we have the following:

$\psi(u = v)^{\sigma\psi}$ is identical with $\psi(u)^{\sigma\psi} = \psi(v)^{\sigma\psi}$ and consequently with $\psi(u^\sigma) = \psi(v^\sigma)$

♦

In the following we investigate to what extend interpretation functions preserve reasoning using propositional tautologies and resolution rule.

Modus ponens rule allows deriving formula B from formulas: $A \Rightarrow B$ and A. This rule is used in propositional logic. *SLD-resolution rule* can be formulated as follows:

Let A be an atom; i.e. let A have the form $p(t_1,..., t_n)$, for a predicate p and terms $t_1,..., t_n$.

If σ is the mgu of A and B, then from $C \leftarrow A$ and $B \leftarrow B_1,..., B_k$ follows $C^\sigma \leftarrow B_1^\sigma,..., B_k^\sigma$.

A clause of the form $B \leftarrow B_1,..., B_k$ (as it is understood in logic programming) corresponds to the implication $B_1 \wedge ... \wedge B_k \Rightarrow B$. We can equivalently express the resolution rule in predicate logic; the formula $B_1^\sigma \wedge ... \wedge B_k^\sigma \Rightarrow C^\sigma$ follow from $B_1 \wedge ... \wedge B_k \Rightarrow B$ and $A \Rightarrow C$, if A is an atom and σ is a mgu of A and B. We assume here that for k = 0, the formula

$B_1 \wedge ... \wedge B_k \Rightarrow B$ is equal B. Let us observe that in order for this rule to be sound it is enough to require that σ is a unifier. This form of rule is called *general resolution rule*. We call general resolution rule *resolution rule*, if the underlying substitution is a mgu. Let us also observe that under the above interpretation, modus ponens is a special kind of resolution rule in which the unifier is an empty mapping. Moreover the general resolution rule does not provide any additional deductive power, and proofs can be equivalently performed using resolution rule and substitution (cf. e.g. [11]).

A propositional tautology A can be instantiated by substituting in A first order formulas for propositional variables.

Birkhoff proposed a sound and complete system of reasoning rules for equations (see [1]). In our framework, one can specify rules such as reflexivity, symmetry and transitivity using predicate formulas:

$$x = x, \qquad x = y \Rightarrow y = x, \qquad x = y \wedge y = z \Rightarrow x = z$$

Let us notice that the substitution rule which says that if u = v, then u[t/x] = v[t/x] can be obtained from the reflexivity axiom and from the tautology $u[t/x] = v[t/x] \Rightarrow u[t/x] = v[t/x]$ using resolution rule and the substitution [t/x] as the mgu:

u[t/x] = v[t/x] follows from u = v and from the tautology $u[t/x] = v[t/x] \Rightarrow u[t/x] = v[t/x]$, by application the unifier [t/x]; in fact, this is a most general unifier.

A *proof* is a sequence of formulas $A_1,..., A_m$, such that for every i, A_i is an axiom, or A_i follows from the proceeding formulas using reasoning rules. We call proofs using propositional tautologies and resolution rule *PTR-proofs*:

$A_1,..., A_m$ is a PTR-proof if and only if for every i

- A_i is an axiom or
- A_i is obtained from a propositional tautology by substituting a first order formulas for propositional variables or
- A_i follows from the proceeding formulas by application of the resolution rule.

A *proof is covered* if every element of the corresponding sequence is covered. The next theorem says that the interpretation function preserve PTR-proofs.

5.2 Lemma

Let gen(A, Y) be a term space, which has a top, let A be orthogonal, and let propositional terms $x \wedge y$, $x \Rightarrow y$ and $\neg x$ belong to A[3]. Let $Ax \subseteq gen(A, Y)$, and let $A_1,..., A_m$ be a PTR-proof. Then for k = 1,..., m the formula $R(A_k)$ satisfies one of the following conditions:

- It belongs to Ax.

3. Let us remind that propositions can be treated as boolean valued terms.

- It is a propositional tautology.
- There exist i, j < k, such that $R(A_k)$ follows from $R(A_i)$ and $R(A_j)$ by an application of the resolution rule and the tautology $R(A_k) \Rightarrow R(A_k)$.

Proof

If A_k is an axiom, then from the definition of R follows that $R(A_k)$ is equal A_k. Similarly, if A_k is a propositional tautology, then $R(A_k)$ is a propositional tautology, since by definition R commutes with terms from A and A contains propositional operations; e.g. $R(A \vee \neg A) = R(A) \wedge R(\neg A)$.

Let A_k follow from A_i and A_j by application of the resolution rule. Let A_i have the form $B_1 \wedge ... \wedge B_m \Rightarrow B$, let A_j have the form $A \Rightarrow C$, and let σ be the corresponding mgu (the other case is symmetric). Then C^σ is identical with A_k.

We show that R(A) and R(B) are unifiable. Let us define substitution θ as follows: $\theta(x_t) = t$, and $\theta(x) = x$, for all other x. Then θ maps R(A) and R(B) on A and B, respectively. $\theta\sigma$ is the required unifier. Let ν be the mgu of R(A) and R(B). Let D_k be the formula resulting from the application of resolution rule and the mgu ν. Since ν is a mgu, there is a mapping η, such that $\nu\eta$ is equal σ and D_k^η identical with A_k.

From lemma 4.4 (c) follows that there exist a substitution γ, such that $R(D_k)^\gamma$ is identical with $R(A_k)$. From lemma 4.2 follows that D_k is covered, since R(A) and R(B) are covered, and therefore D_k is equal to $R(D_k)$. Clearly, γ is a mgu of $R(D_k)$ is identical with $R(A_k)$. Therefore we can deduce $R(A_k)$ from $R(D_k)$ and $R(A_k) \Rightarrow R(A_k)$ using resolution rule and the mgu γ. ♦

5.3 Theorem

Let $\psi : T(S, F, \leq, X, \tau) \rightarrow T(S', F', \leq', X', \tau')$ be an interpretation function. Let the underlying sort mapping ρ be defined on the largest sort of S. Let ψ be defined on a set of formulas Ax and on the formula A. If there is a PTR-proof of formula A, then there is a PTR-proof of $\psi(A)$ using $\psi(Ax)$.

Proof

Let the assumption of this theorem be satisfied. We have to prove that ψ preserves propositional tautologies and maps an application of the resolution rule on an application of the general resolution rule. Let us assume that A_i is a propositional tautology. Then $\psi(A)$ is a tautology as well, since by definition compositional functions preserve propositional operators (e.g. $\psi(A \vee \neg A)$ is transformed to $\psi(A) \vee \neg\psi(A)$). The case when A_i is derived by application of modus ponens is analogous to the case of tautologies, since interpretation functions preserve propositional operations.

Let gen(A, Y) be the domain of ψ. Since $\psi(A)$ is defined, R(A) is equal to A. Let A_1,..., A_m be the proof of A, and let us consider the sequence $R(A_1)$,..., $R(A_m)$. For every i, if A_i is a propositional tautology, then $R(A_i)$ is a propositional tautology as well. Similarly, if A_i is an axiom from Ax, then $R(A_i)$ is identical with A_i, since ψ is defined on Ax. We will extend the sequence to a covered proof of A. Let the extension be done for $l < k$, and let it have the form B_1,..., B_{ik}. If A_k is an axiom, then we extend that proof to B_1,..., B_{ik}, A_k. If A_k is a propositional tautology, then we extend that proof to B_1,..., B_{ik}, $R(A_k)$. If A_k follow from A_i and A_j by application of the resolution rule, then we extend the proof to B_1,..., B_{ik}, $R(D_k)$, $R(A_k) \Rightarrow R(A_k)$, $R(A_k)$ (see the proof above). From lemma 4.2 follows that D_k is covered. From the construction follows that in this way we obtain a covered PTR-proof of A.

Let $\psi(A)$, $\psi(B)$ be defined and let σ be their most general unifier. From lemma 5.1 follow that $\sigma\psi$ is a unifier, but not necessarily a mgu. Let $\psi(K_1)$,..., $\psi(K_n)$ be the resulting proof. As stated above, this proof uses unifiers and the general resolution rule instead of mgus and resolution rule.

Let $\psi(K_k)$ follow from $\psi(K_i)$ and $\psi(K_j)$, by application of the general resolution rule. Since $\psi(K_i)$ and $\psi(K_j)$ are unifiable, there exist a mgu due to lemma 4.2. Let C_k be the formula which is obtained by application of resolution rule to $\psi(K_i)$ and $\psi(K_j)$. $\psi(K_k)$ can be obtained from C_k and from the tautology $\psi(K_k) \Rightarrow \psi(K_k)$ as in lemma 5.2. This proves that $\psi(K_1)$,..., $\psi(K_n)$ can be extended to a PTR-proof. ♦

Let us observe that from the previous theorem follows that proofs using reflexivity, symmetry and transitivity of equations are preserved as well. This is due to the fact that the corresponding axioms are preserved by interpretation functions; i.e. $\psi(x = x) \equiv \psi(x) = \psi(x)$, $\psi(x = y \Rightarrow y = x) \equiv \psi(x) = \psi(y) \Rightarrow \psi(y) = \psi(x)$ (see above).

The following theorem says that interpretation functions preserve proofs by induction. For simplicity, we consider here only one sort, constant and unary constructors, the proof for the case of many sorts and arbitrary constructors is analogous.

5.4 Theorem

Let $\psi : T(S, F, \leq, X, \tau) \rightarrow T(S', F', \leq', X', \tau')$ be an interpretation function. Let the underlying sort mapping ρ be defined on the largest sort of S. Let ψ be defined on a set of formulas Ax and on the formula $\Phi(x : s)$. Let the set A_s corresponding to sort s be generated by a constant constructors c and a constructor of the form $f(x : s, s_1 : y_1,.., s_n : y_n)$, i.e. let A_s be the smalles set containing c^A and such that if $a \in A_s$, $a_1 \in A_{s1}$,.., $a_n \in A_{sn}$, then $f^A(a, a_1,.., a_n) \in A_s$. Let the sort s be mapped on a sort s´, and let the constructors of the sort s´

be images of c and f. If there is a PTR-proof of $\Phi(c)$, and if there is a PTR-proof of $\Phi(x) \Rightarrow \Phi(f(x, y_1,..., y_n))$, then there is an inductive proof of Φ preserved by ψ.

Proof

Let us observe that the proof of $\psi(\Phi(c))$, can be obtained from the proof of $\Phi(c)$, due to theorem 5.3. Similarly, the proof of $\psi(\Phi(x) \Rightarrow \psi(\Phi(f(x, y_1,..., y_n))$, can be obtained from the proof of $\Phi(x) \Rightarrow \Phi(f(x, y_1,..., y_n)))$. ♦

Concluding remarks

In this paper we have shown that compositional functions generated by orthogonal mappings preserve different entailment relations. In particular, they preserves restricted form of equational deduction, propositional tautologies and deduction rules like modus ponens and resolution. These results show that different kinds of class structure refactorings [5] and applications of design patterns preserve entailment relation of deduction systems. This paper leaves some questions open, in particular to what extend proofs in first order logic or proofs in Hoare Logic are preserved. The notion of order sorted algebras have been extended to so called membership equational logic [2], which is much more powerful and flexible. It would be interesting to extend presented results to investigate to what extend presented results can be extended to this logic.

References

1 Birkhoff, G.: Lattice Theory. American Mathematical Society, Providence, RI, 1979.
2 Bouhoula, A. Jouannaud, J.P., Meseguer, J.: Specification and Proof in Membership Equational Logic. Theoretical Computer Science, vol. 236(1-2), 2000.
3 Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Addison-Wesley, Reading, 1995.
4 Goguen, J., Meseguer, J.: Order sorted algebra. Theoretical Computer Science, vol. 105(2), Elsevier, Amsterdam, 1992, pp 167-215.
5 Fowler, M.: Refactoring: improving the design of existing code. Reading, Massachusetts, Addison-Wesley, 2000.
6 Graetzer, G.: Universal Algebra. The University Series in Higher Mathematics, Van Nostrand, Princeton, 2nd Edition. Springer, New York, 1979.
7 Kosiuczenko, P.: Formal Redesign of UML Class Diagrams. In (A. Evans, R. France, A. Moreira, B. Rumpe eds): Proc. of pUML Workshop on Practical UML-Based Rigorous Development Methods, Toronto. GI-Edition, Lecture Notes in Informatics, 2001.
8 Martelli, A., Montanari, U.: An Efficient Unification Algorithm. Transactions on Programming Languages and Systems, 4(2), 1982, pp. 258 - 282.
9 OMG. Unified Modeling Language Specification. Version 2.0, 2004.
10 Taylor, W.: Characterizing Mal'cev conditions. Algebra Universalis, 3, Springer, Berlin, 1973, pp. 351-397.
11 Terese et. al.: Term rewriting systems. Cambridge University Press, 2003.
12 Wirsing, M.: Algebraic specification. In: (J. van Leeuwen ed.): Handbook of Theoretical Computer Science. Elsevier, Amsterdam, 1990, pp. 677-780.