ICOT Technical Report: TR-152

TR-152

Computation by Meta-Unification with Constructore

by
T. Kanamori
(Mitsubishi Electric corp.)

October, 1985

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Computation by Meta-Unification with Constructors

Tadashi KANAMORI

Mitsubishi Electric Corporation
Central Research Laboratory
8-1-1 Tsukaguchi-Honmachi
Amagasaki, Hyogo, JAPAN 661

Abstract

A computation mechanism "computation by meta-unification with constructors" is proposed. This view of computation stems from the behavior of an interpreter of an equational language called Talos. In Talos everything is done by controlled sequences of meta-unifications, as is by controlled sequences of unifications in Prolog. This is a generalization of the conventional term rewriting as well. We show a nondeterministic equational meta-unification algorithm to answer whether a set of equations $\mathcal{E}_0$ is metaunifiable by a conditional equational theory $\mathcal{E}$ satisfying some conditions. Then we prove its ground completenes,that is, it computes a more general substitution than any $\mathcal{E}$-unifier, when the $\mathcal{E}$-unifier instanciates $\mathcal{E}_0$ to a set of ground equations. The operational semantics of Talos is given based on the algorithm. The model theoretic semantics is given by the initial algebra of $\mathcal{E}$, or equivalently, the set of all ground equations valid in all models of $\mathcal{E}$. The fixpoint semantics is defined similarly to Prolog. Using the ground completeness, we show these semantics are equivalent.

Keywords : Equational Theories,Term Rewriting Systems,Unification,Semantics.

Contents

1. Introduction
2. Meta-Unification for Conditional Equational Theories
   2.1. Conditional Equational Theories
   2.2. Meta-Unification
   2.3. Consistency and Completeness
3. Syntax of Talos
   3.1. Definition of Data Types
   3.2. Definition of Functions
   3.3. Query
4. Meta-Unification for Conditional Equational Theories with Constructors
   4.1. Conditional Equational Theories with Constructors
   4.2. Meta-Unification with Constructors
   4.3. Consistency and Ground Completeness
5. Semantics of Talos
   5.1. Operational Semantics
   5.2. Model Theoretic Semantics
   5.3. Equivalence of Two Semantics
6. Discussions
7. Conclusions

Acknowledgements
References

1. Introduction

Prolog [4] is a relational language based on first-order predicate calculus. Operational semantics of Prolog is usually explained by the SLD-resolution, a strategy of the resolution complete for Horn clauses. Prominent features of Prolog are procedure invocation by unification and nondeterministic search (automatic backtracking). Results of procedures are passed through variables within each clause, while,in functional programs like Lisp, nested composition of functions is the main construct.

Functional languages are more classical and share semantical clearness with Prolog ([3],[11],[21]). They can be considered special logic programming languages based on equational logic. When an equation is considered a term rewriting rule, equational logic turns into computation, which is the basis of the operational semantics of functional programs. Though functional programs are superior to Prolog in some points (readability etc), they lack some powerfull features of Prolog such as nondeterministic search. When we accomodate these features to functional programming, we need to carry it out not by an *ad hoc* device but by a unified approach. Several such attempts have been done from different point of views ([2],[6],[8],[19],[20]).

In this paper we propose a computation mechanism "computation by meta-unification with constructors". This view of computation stems from the behavior of an interpreter of an equational language called Talos. In Talos everything is done by controlled sequence of meta-uifications, as is by controlled sequences of unifications in Prolog. This is a generalization of the conventional term rewriting as well. Both invocations of functions by unification and automatic backtrackings are integrated into Talos.

This paper is organized as follows. In section 2, we introduce conditional equational theories in general, give an extension of the Fay-Hullot's meta-unification algorithm and prove its completeness. In section 3, we show the syntax of our programming language Talos. In section 4, we introduce conditional equational theories with constructors, give a nondeterministic equational algorithm to answer whether a set of equations $\mathcal{E}_0$ is metaunifiable by a conditional equational theory $\mathcal{E}$ satisfying some conditions. Then we prove its ground completenes,that is, it computes a more general substitution than any $\mathcal{E}$-unifier, when the $\mathcal{E}$-unifier instanciates $\mathcal{E}_0$ to a set of ground equations. In section 5,we discuss the semantics of Talos. The operational semantics of Talos is given based on the equational meta-unification algorithm. The model theoretic semantics is given by the initial algebra of $\mathcal{E}$, or equivalently,the set of all ground equations valid in all models of $\mathcal{E}$. The fixpoint semantics is defined similarly to Prolog. Then using the ground completeness, we show these semantics are equivalent. Lastly in section 6, we discuss the relations to other works.

In this paper we assume familiarity with (many-sorted) equational logic and term rewriting systems. As syntactical variables, we use X, Y, Z for variables, f, g, h for function symbols, a, b, c for constants, $r, s, t, \alpha, \beta, \gamma, \delta$ for terms, u, v for occurrences and $\theta, \sigma, \tau, \mu, \nu, \zeta, \eta, \rho$ for substitutions, possibly with primes and subscripts. $\equiv$ is used to denote the syntactical identity. We denote the set of all terms on a signature Σ and variables $\mathcal{V}$ by $\mathcal{T}(\Sigma \bigcup \mathcal{V})$ (or simply $\mathcal{T}$), the set of all ground terms on a signature Σ by $\mathcal{G}(\Sigma)$ (or simply $\mathcal{G}$), set of all variables in a syntactical object e by $\mathcal{V}(e)$, subterm of t at an occurrence u by t/u, replacements of a subterm of t at an occurrence u with a term s by $t[u \Leftarrow s]$, the set of all occurrences of non-variable subterms of a term s by $\mathcal{O}(s)$ and restriction of a substitution σ to a set of variables V by $\sigma|V$. (see [12],[15]).

2. Meta-Unification for Conditional Equational Theories

We generalize the concepts for unconditional equational theories to those for conditional equational theories first.

2.1. Conditional Equational Theories

A *conditional equational theory* $\mathcal{E}$ is a first order theory with one infix binary predicate $=$, a set of axioms,called *proper axioms* of $\mathcal{E}$,of the form $(m \geq 0)$

$$\gamma_1 = \delta_1 \wedge \gamma_2 = \delta_2 \wedge \cdots \wedge \gamma_m = \delta_m \supset \gamma = \delta$$

and four axioms called *equality axioms*

$$\begin{aligned}
&X=X,\\
&X=Y \supset Y=X,\\
&X=Y \wedge Y=Z \supset X=Z,\\
&X=Y \supset f(Z_1,\ldots,X,\ldots,Z_n) = f(Z_1,\ldots,Y,\ldots,Z_n) \quad \text{for all function symbols } f.
\end{aligned}$$

When $s = t$ is provable in $\mathcal{E}$, we denote it by $=_{\mathcal{E}}$. The quotient algebra of $\mathcal{G}$ by the congruence relation defined by all ground equations provable in $\mathcal{E}$ is called the *initial algebra* of $\mathcal{E}$, or more exactly, said to be isomorphic to the initial algebra.

Example 2.1.1. A theory $\mathcal{E}$ with proper axioms

insert(X,∅)=tree(∅,X,∅),
X=Y ⊃insert(X,tree(L,Y,R))=tree(L,Y,R),
less-than(X,Y)=true ⊃insert(X tree(L,Y,R))=tree(insert(X,L),Y,R),
less-than(Y,X)=true ⊃insert(X,tree(L,Y,R))=tree(L,Y,insert(X,R))

is a conditional equational theory.

A *conditional term rewriting system* $\mathcal{R}$ is a first order theory with three infix binary predicates $\rightarrow, \rightarrow^*$ and $\downarrow$, a set of axioms,called *proper axioms of* $\mathcal{R}$,of the form $(m \geq 0)$

$$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$$

and four axioms called *reducibility axioms*

$$\begin{aligned}
&X \rightarrow^* X,\\
&X \rightarrow Y \wedge Y \rightarrow^* Z \supset X \rightarrow^* Z,\\
&X \rightarrow Y \supset f(Z_1,\ldots,X,\ldots,Z_n) \rightarrow f(Z_1,\ldots,Y,\ldots,Z_n) \quad \text{for all function symbols } f,\\
&X \rightarrow^* Z \wedge Y \rightarrow^* Z \supset X \downarrow Y.
\end{aligned}$$

Example 2.1.2. A theory $\mathcal{R}$ with proper axioms

insert(X,∅) → tree(∅,X,∅),
X ↓ Y ⊃insert(X,tree(L,Y,R)) → tree(L,Y,R),
less-than(X,Y) ↓ true ⊃insert(X tree(L,Y,R)) → tree(insert(X,L),Y,R),
less-than(Y,X) ↓ true ⊃insert(X,tree(L,Y,R)) → tree(L,Y,insert(X,R))

is a conditional term rewriting system.

A binary relation $\rightarrow$ on the set of all terms T is said to be *stable* iff $\sigma(s) \rightarrow \sigma(t)$ for any substitution σ when $s \rightarrow t$ and said to be *compatible* iff $r[u \Leftarrow s] \rightarrow r[u \Leftarrow t]$ for any occurence u of r when $s \rightarrow t$ ([12] p.809). Let $\mathcal{R} = (\rightarrow, T)$ be a compatible stable relation and $\rightarrow^*$ be

the reflexive transitive closure of $\rightarrow$. $\mathcal{R}$ is said to be *confluent* when,for any terms t,t_1,t_2 such that $t \rightarrow^* t_1$ and $t \rightarrow^* t_2$, there exists a term t' such that $t_1 \rightarrow^* t'$ and $t_2 \rightarrow^* t'$. When $\mathcal{R} = (\rightarrow, \mathcal{T})$ is confluent, $\mathcal{R}$ dsefines a binary congruence relation $=_{\mathcal{R}}$ which is the reflexiove symmetric transitive closure of $\rightarrow$. $\mathcal{R}$ is said to be *terminating* when,for any term t_0,there is no infinite derivation in $\mathcal{R}$ $t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots$ such that $t_i \rightarrow t_{i+1}$ is in $\mathcal{R}$ $(0 \leq i)$. A term s is said to be *in $\mathcal{R}$-normal form* when there is no t such that $s \rightarrow t$ is in $\mathcal{R}$. A term t is called *$\mathcal{R}$-normal form* of a term s and denoted by $s\downarrow$ when $s \rightarrow^* t$ holds for $\mathcal{R}$ and t is in $\mathcal{R}$-normal form. A substitution η is said to be *$\mathcal{R}$-normalized* iff $\eta(X)$ is in $\mathcal{R}$-normal form for all X. (By abuse of notation,we use the same symbol $\mathcal{R}$ and $\mathcal{E}$ to denote both theories and concrete relations.)

Example 2.1.3. Let $\overline{\mathcal{R}}$ be a relation on $\mathcal{T}$ such that $s \rightarrow t$ is in $\overline{\mathcal{R}}$ iff it is a logical consequence of a conditional term rewriting system $\mathcal{R}$ with the following proper axioms.

a $\rightarrow$ b.
a $\rightarrow$ c.
f(b) $\rightarrow$ g(c).
f(Y) $\downarrow$ g(Y) $\supset$ b $\rightarrow$ 0.
f(Y) $\downarrow$ g(Y) $\supset$ c $\rightarrow$ 0.
f(Y) $\downarrow$ g(Y) $\supset$ f(X) $\rightarrow$ suc(X).
f(Y) $\downarrow$ g(Y) $\supset$ g(X) $\rightarrow$ suc(X).

Then it is trivial that $\overline{\mathcal{R}}$ is confluent and terminating. $\mathcal{G}/=_{\overline{\mathcal{R}}}$ is isomorphic to the set of all natural numbers **N**.

2.2. Meta-Unification

2.2.1. Meta-Unification Problem

Let $\mathcal{E}$ be a congruence relation on $\mathcal{T}$. s and t are said to be *$\mathcal{E}$-unifiable* iff there exists a substitution θ such that $\theta(s) =_{\mathcal{E}} \theta(t)$. Such a substitution θ is called an *$\mathcal{E}$-unifier* of s and t. The set of all $\mathcal{E}$-unifiers of s and t is denoted by $\mathcal{U}_{\mathcal{E}}(s,t)$. In general the most general $\mathcal{E}$-unifier does not always exists when $\mathcal{E}$ is not the syntactical identity.

To show a generalization of the most general unifier, we introduce an ordering. For $s,t \in \mathcal{T}(\Sigma \bigcup \mathcal{V})$, $s \preceq_{\mathcal{E}} t$ iff there exists ρ such that $\rho(s) =_{\mathcal{E}} t$. $\preceq_{\mathcal{E}}$ is extended to substitutions by $\sigma \preceq_{\mathcal{E}} \tau[\![V]\!]$ iff there exists a substitution ρ such that $\rho \circ \sigma(X) =_{\mathcal{E}} \tau(X)$ for all $X \in V$, where V is a set of variables. (When $\mathcal{E}$ is the syntactical identity,all these definitions correspond to the usual definitions of $s \preceq t$ and $\sigma \preceq \tau[\![V]\!]$. See [12] pp.806-808.)

The set of all variables X such that $\sigma(X) \not\equiv X$ is called the *domain* of σ and denoted by $\mathcal{D}(\sigma)$. The set of all variables in $\sigma(X)$ for all $X \in \mathcal{D}(\sigma)$ is called the *variables introduced by* σ and denoted by $\mathcal{I}(\sigma)$. A substitution σ is said to be away *from* a set of variables W when $\mathcal{I}(\sigma) \bigcap W = \emptyset$.

Let W be a set of variables containing $V = \mathcal{V}(s) \bigcup \mathcal{V}(t)$. A set of $\mathcal{E}$-unifiers $\mathcal{U}$ is called a *complete* set of $\mathcal{E}$-unifiers of s and t away from W iff it satisfies the following conditions. (Use of W is technical for avoiding conflicts of variable names.)

(a) $\forall \theta \in \mathcal{U}$ $(\mathcal{D}(\theta) \subseteq V$ & θ is away from $W)$.
(b) $\mathcal{U} \subseteq \mathcal{U}_{\mathcal{E}}(s,t)$.
(c) $\forall \sigma \in \mathcal{U}_{\mathcal{E}}(s,t)\, \exists \theta \in \mathcal{U}$ $\theta \preceq_{\mathcal{E}} \sigma[\![V]\!]$.

Complete sets of $\mathcal{E}$-unifiers always exist ([15]).

Example 2.2.1. Let $\mathcal{E}$ be the equational theory of the associativity,i.e. $\mathcal{E} = \{(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z)\}$ or the minimum congruence relation on $\mathcal{T}$ satisfying the theory. Let s, t be terms $X \bullet a$ and $a \bullet X$ respectively. Then

$\theta_i = < X \Leftarrow a \bullet (a \bullet (a \bullet \cdots (a \bullet a))) >$

are all $\mathcal{E}$-unifiers of s and t, where the term substituted for X and Y consists of i a's. There is no $\preceq$ relations between these substitutions. Hence there is no finite complete set of $\mathcal{E}$-unifiers of s and t. A complete set of associative unifiers is not finite in general.

2.2.2. Narrowing

Let $\mathcal{R}$ be a conditional term rewriting system, s be a term, W be a set of variables containing $\mathcal{V}(s)$ and $\alpha \rightarrow \beta$ be an instance of the head rule which is provable in $\mathcal{R}$ and numbered k in some numbering. A substitution θ is called a *logical narrowing substitution* of s away from W, if a nonvariable subterm s/u and the left hand side α is unifiable by a most general unifier θ. We assume $\mathcal{V}(\alpha)$ is away from W by renaming away the variables in $\alpha \rightarrow \beta$ from W. s is said to be *logically narrowed to* $t \equiv \theta(s[u \Leftarrow \beta])$ and denoted by $s \overset{\theta}{\rightsquigarrow}_{(u,k,\varepsilon)} t$. In particular,when $s \overset{\theta}{\rightsquigarrow}_{(u,k,\varepsilon)} t$ and $\theta | \mathcal{V}(s)$ is the empty substitution $<>$, s is said to be *logically reduced to* $t \equiv \theta(s[u \Leftarrow \beta])$ and denoted by $s \rightarrow_{(u,k,\varepsilon)} t$. Note that the logical reduction is included in the logical narrowing, i.e.,$\rightarrow \subseteq \rightsquigarrow$. The set of all narrowing substitutions for s away from W is denoted by $NS(s, W)$.

The logical reductions in $\mathcal{R}$ define a relation $\overline{\mathcal{R}}$ on $\mathcal{T}$. Let $\overline{\mathcal{R}}_0, \overline{\mathcal{R}}_1, \overline{\mathcal{R}}_2, \ldots$ be a sequence of relations as follows.

$\overline{\mathcal{R}}_0 = \emptyset$.

$\overline{\mathcal{R}}_{d+1} =$ compatible closure of

$\{\rho(\gamma \rightarrow \delta) \mid$ there exists a proper axiom

$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$

such that $\rho(\gamma_1 \downarrow \delta_1), \rho(\gamma_2 \downarrow \delta_2), \ldots, \rho(\gamma_m \downarrow \delta_m)$ hold for $\overline{\mathcal{R}}_d\}$.

Note that $\overline{\mathcal{R}}_0 \subseteq \overline{\mathcal{R}}_1 \subseteq \overline{\mathcal{R}}_2 \subseteq \cdots$. $\overline{\mathcal{R}}$ is defined by $\bigcup_{d=0}^{\infty} \overline{\mathcal{R}}_d$. An atom in $\overline{\mathcal{R}}_d$ is said to be *with logical degree less than or equal to* d. It is easy to see that $s =_{\overline{\mathcal{R}}}$ iff $s =_{\mathcal{E}} t$ when $\overline{\mathcal{R}}$ is confluent.

The previous definition is logical in the sense that it depends on the concept "$\alpha \rightarrow \beta$ is provable in $\mathcal{R}$". We need to define it operationally, i.e. show how to compute $\theta(s[u \Leftarrow \beta])$ without proving the atom $\alpha \rightarrow \beta$ all the way. We define *operational narrowing* and *operational meta-unifiability* mutually recursively as follows.

(a) Let s be a term, W be a set of variables containing $\mathcal{V}(s)$ and $\gamma \rightarrow \delta$ be an unconditioal rule numbered k in $\mathcal{R}$. Then a substitution σ is called a *pre-narrowing substitution* of s away from W, if a nonvariable subterm s/u and the left hand side γ of the head rule is unifiable by a most general unifier σ. We assume $\mathcal{V}(\gamma)$ is away from W by renaming away the variables in $\gamma \rightarrow \delta$ from W. s is said to be *operationally narrowed to* $t \equiv \sigma(s[u \Leftarrow \delta])$ *with operational degree* 1 and denoted by $s \overset{\sigma}{\rightsquigarrow}_{[u,k,<>\circ\sigma]} t$.

(b) Let s be a term, W be a set of variables containing $\mathcal{V}(s)$ and $\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$ be a conditional rule numbered k in $\mathcal{R}$. Then a substitution σ is called a *pre-narrowing substitution* of s away from W, if a nonvariable subterm s/u and the left hand side γ of the head rule is unifiable by a most general unifier σ. We assume $\mathcal{V}(\gamma)$ is away from W by renaming away the variables in $\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$ from W. s is said to be *operationally narrowed to* $t \equiv \tau \circ \sigma(s[u \Leftarrow \delta])$ with operational degree $d + 1$ and denoted by $s \overset{\sigma}{\rightsquigarrow}_{[u,k,\tau\circ\sigma]} t$ when the instance of two terms composed of the condition part $\sigma(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m))$ and $\sigma(h_m(\delta_1, \delta_2, \ldots, \delta_m))$ are operationally

metaunifiable with operational degree d by τ away from $W + I(\sigma)$, where h_m is a fresh m-ary function symbol.

(c) Let s_0 and t_0 be two terms, W_0 be a set of variables containing $\mathcal{V}(s_0) \bigcup \mathcal{V}(t_0)$ and h_2 be a fresh binary function symbol. Then s_0 and t_0 is said to be *operationally meta-unifiable with operational degree* d by $\theta' \circ (\tau_{n-1} \circ \sigma_{n-1}) \circ \cdots \circ (\tau_1 \circ \sigma_1) \circ (\tau_0 \circ \sigma_0)$ away from W_0 when there exists a sequence

$$h_2(s_0, t_0) \curlywedge_{[u_0, k_0, \tau_0 \circ \sigma_0]} h_2(s_1, t_1) \curlywedge_{[u_1, k_1, \tau_1 \circ \sigma_1]} \cdots \curlywedge_{[u_{n-1}, k_{n-1}, \tau_{n-1} \circ \sigma_{n-1}]} h_2(s_n, t_n)$$

such that each $\curlywedge_{[u_i, k_i, \tau_i \circ \sigma_i]}$ is a narrowing with operational degree less than d away from W_i and s_n and t_n are unifiable by a most general unifier θ', where $W_{i+1} = W_i + I(\tau_i \circ \sigma_i)$.

In particular, when $s \curlywedge_{[u,k,\nu\circ\mu]} t$ and $\nu \circ \mu | \mathcal{V}(s)$ is the empty substitution $<>$, s is said to be *operationally reduced to* $t \equiv \nu \circ \mu(s[u \Leftarrow \delta])$ and denoted by $s \rightarrow_{[u,k,\nu\circ\mu]} t$. Again the operational reduction is included in the operational narrowing, i.e., $\rightarrow \subseteq \curlywedge$. The set of all pre-narrowing substitutions for s away from W is denoted by $NS_{pre}(s, W)$. This is computable from s and the conditional rules of $\mathcal{R}$ directly.

The operational reductions in $\mathcal{R}$ define a relation $\underline{\mathcal{R}}$ on $\mathcal{T}$. Let $\underline{\mathcal{R}}_0, \underline{\mathcal{R}}_1, \underline{\mathcal{R}}_2, \ldots$ be a sequence of relations as follows.

$\underline{\mathcal{R}}_0 = \emptyset$,

$\underline{\mathcal{R}}_d =$ stable closure of

$\{s \rightarrow t \mid s \rightarrow t$ is an operational reduction with degree less than or equal to $d\}$.

Note that $\underline{\mathcal{R}}_0 \subseteq \underline{\mathcal{R}}_1 \subseteq \underline{\mathcal{R}}_2 \subseteq \cdots$. $\underline{\mathcal{R}}$ is defined by $\bigcup_{d=0}^{\infty} \underline{\mathcal{R}}_d$. $\underline{\mathcal{R}}_d$ is not necessarily confluent even if $\underline{\mathcal{R}}$ is confluent. But $\underline{\mathcal{R}}_d$ is always terminating when $\underline{\mathcal{R}}$ is terminating and a term s is in $\underline{\mathcal{R}}_d$-normal form when it is in $\underline{\mathcal{R}}$-normal form.

Note that $\underline{\mathcal{R}}_1 = \overline{\mathcal{R}}_1$, because the rule in the definition of $\underline{\mathcal{R}}_1$ is unconditional and the narrowing can go without the meta-unification of the condition part in this case,i.e. $NS(s, W) = NS_{pre}(s, W)$.

Example 2.2.2. Let s be $insert(A, insert(B, tree(\emptyset, 1, S)))$ and u be the occurrence of $insert(B, tree(\emptyset, 1, S))$. Because $insert(B, tree(\emptyset, 1, S))$ is unifiable with the left hand side of the third rewrite rule in the definition of *insert* by an mgu $\sigma = < B \Leftarrow X_1, S \Leftarrow S_1, X \Leftarrow X_1, L \Leftarrow \emptyset, Y \Leftarrow 1, R \Leftarrow S_1 >$ and the condition part is satisfied by $\tau = < X_1 \Leftarrow 0 >$, s is narrowed to $s_1 \equiv \tau \circ \sigma(\ insert(A, tree(insert(B, \emptyset), 1, S)))$, i.e. $insert(A, tree(insert(0, \emptyset), 1, S_1))$. $NS_{pre}(s, W)$ includes another two pre-narrowing substitutions corresponding to the second and the fourth rewrite rules. For s_1, we have four pre-narrowing substitutions corresponding to the occurrences of s_1 itself and $insert(0, \emptyset)$.

2.2.3. An Extension of the Fay-Hullot's Algorithm

The following is an adaptation of the nondeterministic $\mathcal{E}$-unification algorithm by Fay [5] (revised by Hullot [16]) for unconditional equational theories. W is initialized to W_0 ($\supseteq \mathcal{V}(s, t)$) before *meta-unify*$(s, t)$ and global during the computation. Note that at **then** branches unnecessary search detected in the **if** test is pruned away.

Example 2.2.3. Let s be $insert(A, insert(B, tree(\emptyset, 1, S)))$ and t be $tree(tree(\emptyset, C, \emptyset), 1, T)$. Because s and t are not unifiable, the Fay-Hullot's algorithm selects the second **when** since $NS_{pre}(t, W) = \emptyset$. Then s can be narrowed to

$s_1 \equiv insert(A, tree(insert(0, \emptyset), 1, S_1))$

by $\sigma \mid V = < B \Leftarrow 0, S \Leftarrow S_1 >$. After appropriate two succeeding narrowings, we have

$s_3 \equiv tree(tree(\emptyset, 0, \emptyset), 1, insert(suc(suc(A_3)), S_3))$.
Then in the next repetition, s_3 is unifiable with $t \equiv tree(tree(\emptyset, C, \emptyset), 1, T)$ by $\sigma = <$
$A_3 \Leftarrow A_4, S_3 \Leftarrow S_4, C \Leftarrow 0, T \Leftarrow insert(suc(suc(A_4)), S_4) >$. Hence
$< A \Leftarrow suc(suc(A_4)), B \Leftarrow 0, C \Leftarrow 0, S \Leftarrow S_4, T \Leftarrow insert(suc(suc(A_4)), S_4) >$
is a meta-unifier of s and t. There are another four meta-unifiers
$< A \Leftarrow 0, B \Leftarrow suc(suc(B_4)), C \Leftarrow 0, S \Leftarrow S_4, T \Leftarrow insert(suc(suc(B_4)), S_4) >$,
$< A \Leftarrow 0, B \Leftarrow 0, C \Leftarrow 0, S \Leftarrow S_4, T \Leftarrow S_4 >$,
$< A \Leftarrow 1, B \Leftarrow 0, C \Leftarrow 0, S \Leftarrow S_4, T \Leftarrow S_4 >$,
$< A \Leftarrow 0, B \Leftarrow 1, C \Leftarrow 0, S \Leftarrow S_4, T \Leftarrow S_4 >$.

meta-unify(s, t:term) : substitution;
$\theta := <>$;
repeat
  when s and t are unifiable by θ' away from W
    stop with answer $\theta' \circ \theta$
  when $NS_{pre}(s, W) \neq \emptyset$
    select $\sigma \in NS_{pre}(s, W)$ and let the corresponding conditional rule be
    "$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$" $(\sigma(\gamma) \equiv \sigma(s/u))$;
    $W := W + I(\sigma)$; let τ be meta-unify($\sigma(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m)), \sigma(h_m(\delta_1, \delta_2, \ldots, \delta_m))$);
    if there exists a variable $X \in W$ for which $\tau \circ \sigma(X)$ is not in $\underline{\mathcal{R}}$-normal form
    then stop with failure
    else $s := \tau \circ \sigma(s[u \Leftarrow \delta])$; $t := \tau \circ \sigma(t)$; $\theta := (\tau \circ \sigma) \circ \theta$
  when $NS_{pre}(t, W) \neq \emptyset$
    select $\sigma \in NS_{pre}(t, W)$ and let the corresponding conditional rule be
    "$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$" $(\sigma(\gamma) \equiv \sigma(t/v))$;
    $W := W + I(\sigma)$; let τ be meta-unify($\sigma(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m)), = \sigma(h_m(\delta_1, \delta_2, \ldots, \delta_m))$);
    if there exists a variable $X \in W$ for which $\tau \circ \sigma(X)$ is not in $\underline{\mathcal{R}}$-normal form
    then stop with failure
    else $s := \tau \circ \sigma(s[u \Leftarrow \delta])$; $\theta := (\tau \circ \sigma) \circ \theta$
endrepeat

Figuare 2.2. Extended Fay-Hullot's Meta-Unification Algorithm

2.3. Consistency and Completeness

We have defined different narrowings and reductions, i.e. the logical ones and the operational ones. Corresponding to each reduction, we have two binary relations on $T(\Sigma \bigcup \mathcal{V})$. One is $\underline{\mathcal{R}}$ corresponding to the operational reduction. Another is $\overline{\mathcal{R}}$ corresponding to the logical reduction. $\underline{\mathcal{R}} \subseteq \overline{\mathcal{R}}$ holds in general (see Lemma 3), but they are not necessarly identical and the proof of consistency and completeness of the algorithm does not go in the completely same way as by Hullot [16]. (The distinction of these two is necessary because we are considering conditional cases. When $\mathcal{E}$ and $\mathcal{R}$ are unconditional, these two are identical.)

Now on,we assume $\underline{\mathcal{R}}$ is confluent and terminating. This means $\underline{\mathcal{R}}$-normal form is unique and $\underline{\mathcal{R}}$ defines a congruence relation $=_{\underline{\mathcal{R}}}$, that is, the reflxive symmetric transitive closure of the operational reduction $\rightarrow$. We define $\preceq_{\underline{\mathcal{R}}}$ similarly to one in 2.2.1.

We introduce a concept, which is abstracted from the theorem by Hullot [16] pp.323-

324 and generalized for conditional cases. It says any operational $\curvearrowright$-derivation issuing from $\eta(s)$ without instanciation of variables in $I(\eta)$ may be "projected" on an operational $\curvearrowright$-derivation issuing from s and any operational $\curvearrowright$-derivation issuing from s may be considered as the "projection" of a certain class of operational $\curvearrowright$-derivation ([16] p.322). The nondeterministic meta-unification algorithm *meta-unify* is said to be *projectable for* $\underline{\mathcal{R}}_d$ when it satisfies the following condition.

(a) Let s be a term, V be a finite set of variables containing $\mathcal{V}(s)$ and η be a $\underline{\mathcal{R}}$-normalized substitution with $D(\eta) \subseteq \mathcal{V}(s)$. Consider any operational $\curvearrowright$-derivation with operational degree less than or equal to d issuing from $\eta(s)$:

$$\eta(s) \equiv t_0 \curvearrowright_{[u_0, k_0, \nu_0 \circ \mu_0]} t_1 \curvearrowright_{[u_1, k_1, \nu_1 \circ \mu_1]} \cdots \curvearrowright_{[u_{n-1}, k_{n-1}, \nu_{n-1} \circ \mu_{n-1}]} t_n. \quad (1)$$

such that no variable in $I(\eta)$ is instanciated in the narrowings,i.e. $D(\nu_i \circ \mu_i) \bigcap I(\eta) = \emptyset$ for all $0 \leq i < n$. Then there exists an associated operational $\curvearrowright$-derivation with operational degree less than or equal to d issuing from s :

$$s \equiv s_0 \curvearrowright_{[u_0, k_0, \tau_0 \circ \sigma_0]} s_1 \curvearrowright_{[u_1, k_1, \tau_1 \circ \sigma_1]} \cdots \curvearrowright_{[u_{n-1}, k_{n-1}, \tau_{n-1} \circ \sigma_{n-1}]} s_n, \quad (2)$$

and for each $i, 0 \leq i \leq n$, a substitution η_i and a finite set of variables V_i such that :

(i) $D(\eta_i) \subseteq V_i$,
(ii) η_i is $\underline{\mathcal{R}}$-normalized,
(iii) $(\eta \mid V) = ((\eta_i \circ \theta_i) \mid V)$,
(iv) $\eta_i(s_i) \equiv t_i$

where $\theta_0 = <>$ and $\theta_{i+1} = (\tau_i \circ \sigma_i) \circ \theta_i$.

(b) Conversely, to each operational $\curvearrowright$-derivation (2) and every η such that $\theta_n \preceq \eta[\![V]\!]$, we can associate an operational $\rightarrow$-derivation (1).

Note that the operational $\curvearrowright$-derivation (1) is an operational $\rightarrow$-derivation treated in the Hullot's original Theorem 1, when $D(\eta) = \mathcal{V}(s)$ and $\mathcal{V}(\delta) \subseteq \mathcal{V}(\gamma)$ for all conditional rules used in the derivation. The first lemma is a generalization of the Theorem 1.

Lemma 1. When $\underline{\mathcal{R}}$ is confluent and terminating, the *meta-unify* is projectable for $\underline{\mathcal{R}}_d$ $(0 \leq d)$.

Proof. The proof is also a generalization of the Theorem 1 in Hullot [16] pp.323-324. We prove it by induction on operational degree.

Base Case : When $d = 0$,the proof is vacantly true.

Induction Step : We have to prove that the *meta-unify* is projectable for $\underline{\mathcal{R}}_{d+1}$ assuming the *meta-unify* is projectable for $\underline{\mathcal{R}}_d$.

The $\Rightarrow$-part (a) is by induction on i.

Base Case : For $i = 0$ it is obvious taking $\eta_0 = \eta$ and $V_0 = V \bigcup D(\eta)$.

Induction Step : Let us assume (i) to (iv) hold for i. Since $t_i \curvearrowright_{[u_i, k_i, \nu_i \circ \mu_i]} t_{i+1}$, we have

$$\mu_i(\gamma) \equiv \mu_i(t_i/u_i),$$

where "$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$" is the k_i-th rule and renamed away from V_i. From assumptions (ii),(iii),(iv) for i and the fact that variables in $I(\eta)$ are not instanciated, we get $u_i \in \overline{\mathcal{O}}(t_i)$ and therefore

$$\eta_i(s_i/u_i) \equiv \mu_i(\gamma).$$

Let us consider $\rho = \eta_i \bigcup \mu_i$.We have

$$\rho(s_i/u_i) \equiv \rho(\gamma).$$

Let σ_i be a most general unifier of s_i/u_i and γ. Then there exists a substitution ς' such that $\rho = \varsigma' \circ \sigma_i$. Therefore

$$\eta_i = ((\varsigma' \circ \sigma_i) | V_i).$$

and $\nu_i \circ \varsigma'$ is a $\underline{\mathcal{R}}_d$-unifier of $\sigma_i(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m))$ and $\sigma_i(h_m(\delta_1, \delta_2, \ldots, \delta_m))$. Now,let

$s' \equiv \sigma_i(h_2(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m), h_m(\delta_1, \delta_2, \ldots, \delta_m)))$,
$U = (V_i \bigcup \mathcal{V}(\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta) \bigcup I(\sigma_i)) - D(\sigma_i)$,
$\varsigma = \varsigma' | U$,

Then U is a finite set of variables containing $\mathcal{V}(s')$. Now,let us consider X in U.There are two cases :

(a) $X \in I(\sigma_i)$; then $\exists Y \in D(\sigma_i)$ such that $X \in \mathcal{V}(\sigma_i(Y))$,and $\eta_i(Y) \equiv \varsigma(\sigma_i(Y))$ normalized implies $\varsigma(X)$ normalized.

(b) otherwise $\sigma_i(X) \equiv X$ since $X \notin D(\sigma_i)$ and therefore $\varsigma(X) \equiv \eta_i(X)$ is normalized.

which proves ς is $\underline{R}$-normalized.

Consider the operational $\curlywedge$-derivation giving the meta-unifier ν_i with degree less than or equal to d issuing from $\mu_i(h_2(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m), h_m(\delta_1, \delta_2, \ldots, \delta_m)))$

$$\mu_i(h_2(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m), h_m(\delta_1, \delta_2, \ldots, \delta_m))) \equiv t'_0 \curlywedge t'_1 \curlywedge \cdots \curlywedge t'_m \qquad (2)'$$

Then $t'_0 \equiv \varsigma(s')$ and no variable in $I(\varsigma)$ is instanciated in the operational $\curlywedge$-derivation, since no variable in $I(\eta)$ is instanciated in $t_i \curlywedge_{[u_i, k_i, \nu_i \circ \mu_i]} t_{i+1}$. Because of the inductive assumption,we have a corresponding operational $\curlywedge$-derivation with degree less than or equal to d issuing from s'

$$s' \equiv s'_0 \curlywedge s'_1 \curlywedge \cdots \curlywedge s'_m \qquad (1)'$$

which gives τ_i such that

$s_i \curlywedge_{[u_i, k_i, \tau_i \circ \sigma_i]} s_{i+1}$,
$\tau_i \circ \sigma_i \preceq \rho[\![V_i]\!]$.

Thus there exists a substitution η' such that $\rho = \eta' \circ (\tau_i \circ \sigma_i)$. Therefore

$$\eta_i = (\eta' \circ (\tau_i \circ \sigma_i))|V_i,$$

Now,let

$V_{i+1} = (V_i \bigcup \mathcal{V}(\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta) \bigcup I(\tau_i \circ \sigma_i)) - D(\tau_i \circ \sigma_i)$,
$\eta_{i+1} = \eta' | V_{i+1}$,

We get (i) and

$$\eta_i = (\eta_{i+1} \circ (\tau_i \circ \sigma_i))|V_i. \qquad (*)$$

(We impose $D(\tau_i \circ \sigma_i) \bigcap I(\tau_i \circ \sigma_i) = \emptyset$).

Now similarly to variables in U, let us consider X in V_{i+1}.There are two cases :

(a) $X \in I(\tau_i \circ \sigma_i)$; then $\exists Y \in D(\eta_i)$ such that $X \in \mathcal{V}(\tau_i \circ \sigma_i(Y))$,and $\eta_i(Y) \equiv \eta_{i+1}(\tau_i \circ \sigma_i(Y))$ normalized implies $\eta_{i+1}(X)$ normalized.

(b) otherwise $\tau_i \circ \sigma_i(X) \equiv X$ since $X \notin D(\tau_i \circ \sigma_i)$ and therefore $\eta_{i+1}(X) \equiv \eta_i(X)$ is normalized.

which proves (ii). We now assume (iii) for i

$$\eta | V = (\eta_i \circ \theta_i)|V,$$

and show it for $i+1$.From (*) above,we get

$$(\eta_i \circ \theta_i)|V = (((\eta_{i+1} \circ (\tau_i \circ \sigma_i))|V) \circ \theta_i)|V.$$

From the definition of θ_i,we get $I(\theta_i) \subseteq V_i$ and $V \subseteq V_i \bigcup D(\theta_i)$. The above expression simplifies therefore to

$$(\eta_{i+1} \circ (\tau_i \circ \sigma_i) \circ \theta_i)|V = (\eta_{i+1} \circ \theta_{i+1})|V,$$

proving (iii).

Finally we get easily $\mathcal{V}(s_i) \subseteq V_i$ from which we get

$$\eta_{i+1}(s_{i+1}) \equiv \eta_{i+1} \circ (\tau_i \circ \sigma_i)(s_i[u_i \Leftarrow \delta]) \equiv \eta_i(s_i[u_i \Leftarrow \delta]) \equiv t_{i+1},$$

proving (iv). Note that because of (iii) every $\theta_i|V$ is normalized.

The $\Leftarrow$-part (b) is as follows. Let us consider any operational $\curlywedge$-derivation (2) and any substitution η such that $\theta_n \preceq \eta[\![V]\!]$ in the definition of "projectability" Let ρ be such that $\eta|V = (\rho \circ \theta_n)|V$. We define substitutions η_i for $0 \leq i \leq n-1$ by

$$\eta_i = \rho \circ (\nu_n \circ \mu_n) \circ (\nu_{n-1} \circ \mu_{n-1}) \circ \cdots \circ (\nu_i \circ \mu_i),$$

and substitution η_n as being ρ. With $t_i \equiv \eta_i(s_i)$, it is easy to show by induction on i,that

$\eta(s) \equiv t_0 \rightarrow_{[u_0,k_0,\nu_0\circ\mu_0]} t_1 \rightarrow_{[u_1,k_1,\nu_1\circ\mu_1]} \cdots \rightarrow_{[u_{n-1},k_{n-1},\nu_{n-1}\circ\mu_{n-1}]} t_n$.
Now $t_0 \equiv \eta_0(s_0) \equiv \eta_0(s) \equiv \eta_n \circ \theta_n(s) \equiv \eta(s)$, since $\mathcal{V}(s) \subseteq V$.

The second lemma guarantees consistency and completeness of the extended Fay-Hullot's algorithm.

Lemma 2. When $\underline{\mathcal{R}}$ is confluent and terminating,

(a) $\theta(s) \downarrow \theta(t)$ holds for $\underline{\mathcal{R}}$ when a substitution θ is generated by the *meta-unify*.
(b) The *meta-unify* can generates a substitution θ such that $\theta \preceq_{\underline{\mathcal{R}}} \rho[\![V]\!]$ for any substitution ρ if $\rho(s) \downarrow \rho(t)$ holds for $\underline{\mathcal{R}}$, where $V = \mathcal{V}(s,t)$.

Proof. The proof is a slight modification of the Lemma 1,Lemma 2 and Theorem 2 in Hullot [16] pp.324-325.

The proof of consistency (a) is as follows. Suppose *meta-unify*(s,t) returns θ. Let d be the the maximum operational degree used in the $\leftrightsquigarrow$-derivation issuing from $h_2(s,t)$:

$$h_2(s,t) \equiv s_0 \leftrightsquigarrow s_1 \leftrightsquigarrow s_2 \leftrightsquigarrow \cdots \leftrightsquigarrow s_n \equiv h_2(s',t'),$$

such that s' and t' are unifiable by a substitution θ' and let θ_n is the composition of substitutions along the derivation. Then using the condition (b) in the definition of "projectable for $\underline{\mathcal{R}}_d$" with $\eta = \theta_n$, we can associate to this $\leftrightsquigarrow$-derivation the follwoing operational $\rightarrow$-derivation with degree less than or equal to d.

$$h_2(\theta_n(s),\theta_n(t)) \equiv t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_n \equiv h_2(s'',t''),$$

and thus,we have

$$\theta_n(s) \rightarrow^* s'' \ \&\ \theta_n(t) \rightarrow^* t''.$$

Moreover,since $\eta_n = <>$ in this case,we have $s'' \equiv s' \ \&\ t'' \equiv t'$. Thus

$$\theta' \circ \theta_n(s) \downarrow \theta' \circ \theta_n(t),$$

in $\underline{\mathcal{R}}_d$, since two terms are operationally $\rightarrow$-reducible to the same term.

The proof of completeness (b) is as follows. Suppose ρ is a $\underline{\mathcal{R}}$-unifier of s and t and η is a $\underline{\mathcal{R}}$-normalized substitution of ρ. (We rename some variables by η so that $\mathcal{D}(\eta) = \mathcal{V}(h_2(s,t))$.) Then by $\bigcup_{d=0}^{\infty} \underline{\mathcal{R}}_d = \underline{\mathcal{R}}$, there exists an operational degree d such that the derivation isuing from $h_2(\eta(s),\eta(t))$ to $h_2(r,r)$ is in $\underline{\mathcal{R}}_d$,i.e. we have an operational $\rightarrow$-derivation with degree less than or equal to d

$$h_2(\eta(s),\eta(t)) \equiv t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_n \equiv h_2(r,r),$$

By the condition (a) in the definition of "projectable for $\underline{\mathcal{R}}_d$", the corresponding operational $\leftrightsquigarrow$-derivation with degree less than or equal to d is such that

$$\eta_n(s_n) \equiv h_2(\eta_n(s'),\eta_n(t')) \equiv t_n \equiv h_2(r,r).$$

Thus η_n is a unifier of s_n and t_n. Let θ' be the most general unifier. Then there exists ρ such that $\eta_n = \rho \circ \theta$,therefore

$$(\rho \circ \theta) \circ \theta_n | V = (\eta_n \circ \theta_n | V) = (\eta | V) =_{\underline{\mathcal{R}}} (\rho | V),$$

that is, $\theta' \circ \theta_n \preceq_{\underline{\mathcal{R}}} \rho[\![V]\!]$.

The third lemma says that the logical reduction relation $\overline{\mathcal{R}}$ is identical to the operational reduction relation $\underline{\mathcal{R}}$ under the condition $\underline{\mathcal{R}}$ be confluent and terminating. Then $\underline{\mathcal{R}}$ and $\overline{\mathcal{R}}$ define a same congruence relation. We denote it simply by $=_{\mathcal{E}}$ and define $\preceq_{\mathcal{E}}$ similarly to one in 2.2.1.

Lemma 3. When $\underline{\mathcal{R}}$ is confluent and terminating, $s \rightarrow t \in \underline{\mathcal{R}}$ iff $s \rightarrow t \in \overline{\mathcal{R}}$, that is, $\underline{\mathcal{R}} = \overline{\mathcal{R}}$.

Proof. Both proofs are by induction on degree.

$\underline{\mathcal{R}} \subseteq \overline{\mathcal{R}}$ holds without the condition that $\underline{\mathcal{R}}$ be confluent and terminating.

Base Case : Let $s \rightarrow t$ be in $\underline{\mathcal{R}}_1$. Then $\underline{\mathcal{R}}_1 \subseteq \overline{\mathcal{R}}$ is trivial,because $\underline{\mathcal{R}}_1 = \overline{\mathcal{R}}_1$.

Induction Step : Assume $\underline{\mathcal{R}}_d \subseteq \overline{\mathcal{R}}$ and let $s \to t$ be in $\underline{\mathcal{R}}_{d+1}$. Then s/u is an instance by some substitution μ of the left hand side of the head of a rule $\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \to \delta$ and $\mu(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m))$ and $\mu(h_m(\delta_1, \delta_2, \ldots, \delta_m))$ are meta-unifiable by some substitution ν in $\underline{\mathcal{R}}_d$. Because of the inductive assumption, $\nu \circ \mu(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m))$ and $\nu \circ \mu(h_m(\delta_1, \delta_2, \ldots, \delta_m))$ are reducible to a same term in $\overline{\mathcal{R}}$. Then $s \to t$ is in $\overline{\mathcal{R}}$ by the inductive definition of $\overline{\mathcal{R}}_{d+1}$. Therefore $\underline{\mathcal{R}}_{d+1}$ is included in $\overline{\mathcal{R}}$.

$\overline{\mathcal{R}} \subseteq \underline{\mathcal{R}}$ needs the condition that $\underline{\mathcal{R}}$ be confluent and terminating.

Base Case : Let $s \to t$ be in $\overline{\mathcal{R}}_1$. Then $\overline{\mathcal{R}}_1 \subseteq \underline{\mathcal{R}}$ is trivial,because $\overline{\mathcal{R}}_1 = \underline{\mathcal{R}}_1$.

Induction Step : Assume $\overline{\mathcal{R}}_d \subseteq \underline{\mathcal{R}}$ and let $s \to t$ be in $\overline{\mathcal{R}}_{d+1}$. Then s/u is an instance by some substitution θ of the left-hand side of the head of a rule $\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \to \delta$ and $\theta(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m)) \downarrow \theta(h_m(\delta_1, \delta_2, \ldots, \delta_m))$ holds for $\overline{\mathcal{R}}_d$. Because of the inductive assumption, they are also reducible to a same term in $\underline{\mathcal{R}}$. Let $\mu = \theta | \mathcal{V}(\gamma \to \delta), \theta = \rho \circ \mu$ and $V = \mathcal{V}(\mu(h_2(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m), h_m(\delta_1, \delta_2, \ldots, \delta_m))))$. Now,since $\underline{\mathcal{R}}$ is confluent and terminating, we can use Lemma 2 and $\mu(h_m(\gamma_1, \gamma_2, \ldots, \gamma_m))$ and $\mu(h_m(\delta_1, \delta_2, \ldots, \delta_m))$ are meta-unifiable in $\underline{\mathcal{R}}$ by a substitution ν such that $\nu \preceq_{\underline{\mathcal{R}}} \rho[\![V]\!]$ and ν instanciates no variables in $\mathcal{V}(\mu(\gamma \to \delta))$. Then $s \to t$ is in $\underline{\mathcal{R}}$ by the inductive definition of $\underline{\mathcal{R}}$. Therefore $\overline{\mathcal{R}}_{d+1}$ is included in $\underline{\mathcal{R}}$.

Now we have almost finished the proof of that the extended Fay-Hullot's algorithm is consistent and complete.

Theorem 1 (Consitency and Completeness) When $\underline{\mathcal{R}}$ is confluent and terminating,
$\mathcal{U}(s, t, W_0) = \{\theta | V \mid$ *meta-unify*$(\mathcal{E}_0)$ with initialization $W := W_0$ stops with answer $\theta\}$ is a complete set of $\mathcal{E}$-unifiers of $\mathcal{E}_0$ away from W_0, where $W_0 \supseteq V = \mathcal{V}(s) \bigcup \mathcal{V}(t)$.

Proof. The theorem is,so to say, the consistency and completeness w.r.t. logical reduction,that is,

(a) $\theta(s) \downarrow \theta(t)$ holds for $\overline{\mathcal{R}}$ when a substitution θ is generated by the *meta-unify*.
(b) The *meta-unify* can generates a substitution θ such that $\theta \preceq_t \rho[\![V]\!]$ for any substitution ρ if $\rho(s) \downarrow \rho(t)$ holds for $\overline{\mathcal{R}}$.

But it is trivial by Lemma 3 and Lemma 2 when $\underline{\mathcal{R}}$ is confluent and terminating.

Remark. The converse of Lemma 3 does not hold,i.e. even if $\overline{\mathcal{R}}$ is confluent, $\underline{\mathcal{R}}$ is not necessarily confluent. For example,let $\mathcal{E}$ be a conditional equational theory

a = b.
a = c.
f(b) = g(c).
f(Y) = g(Y) ⊃b = 0.
f(Y) = g(Y) ⊃c = 0.
f(Y) = g(Y) ⊃f(X) = suc(X).
f(Y) = g(Y) ⊃g(X) = suc(X).

Though $\overline{\mathcal{R}}$ is confluent and terminating, $\underline{\mathcal{R}}$ is a strict subset of $\overline{\mathcal{R}}$ and not confluent. ($f(0) \to suc(0)$ is not included in $\underline{\mathcal{R}}$.) Hence,the completeness does not hold even if $\overline{\mathcal{R}}$ is confluent and terminating. For example,suppose we meta-unify $f(A)$ and $suc(A)$. Then,though $< A \Leftarrow 0 >$ is a meta-unifier, we can't compute any meta-unifier subsuming it by the extended Fay-Hullot' meta-unification algorithm.

3. Syntax of Talos

3.1. Definition of Data Types

Definition of data types is similar to the algebraic specification of abstract data types except the separation of constructors. Constructors are operators from which every instance of the type is freely and uniquely constructed. For example, a data type *list* has two constructors *nil* ([ ]) and *cons* ([|]). (We follow the DEC-10 Prolog-like syntax [18].) The choice of constructors is left to programmers.

Example 3.1. A data type *number* is defined as follows.

```
data number = new.
  constructor.
    zero.
    suc(N:number).
  operator.
    add(M,N:number):number.
      M+0=M.
      M+(N1+1)=(M+N1)+1.
    less-than(M,N:number):boole.
      0<N+1=true.
      M<0=false.
      M+1<N+1=M<N.
end.
```

We assume a data type *boole* is already defined. $0, +, 1, <$ are built-in symbols and $suc^i(N)$ is represented by $N + i$.

3.2. Definition of Functions

Functions are defined by a set of conditional equations of the form

$\gamma = \delta$ **where** $\gamma_1 = \delta_1, \gamma_2 = \delta_2, \ldots, \gamma_m = \delta_m$.

When $m = 0$,the condition part (including **where**) is omitted.

Example 3.2. A function inserting an elemnt into a binary tree labelled with numbers is defined as follows.

```
function insert(X:number,T:tree):tree.
  insert(X,∅)=tree(∅,X,∅).
  insert(X,tree(L,Y,R))=tree(L,Y,R) where X=Y.
  insert(X tree(L,Y,R))=tree(insert(X,L),Y,R) where X<Y.
  insert(X,tree(L,Y,R))=tree(L,Y,insert(X,R)) where Y<X.
end.
```

The comparison of the element being inserted and the root element is done in the condition parts. We have added syntactic sugar for boolean-valued function p to denote $p(t_1, t_2, \ldots, t_n)$ in place of $p(t_1, t_2, \ldots, t_n) = true$.

3.3. Query

A query is a conditional term of the form

?- t **when** $s_1 = t_1, s_2 = t_2, \ldots, s_m = t_m$.

Example 3.3. A query to request searching an instance of C satisfying $insert(A, insert(B, tree(\emptyset, 1, S))) = tree(tree(\emptyset, C, \emptyset), 1, T)$ is given as follows.

```
?- C when insert(A,insert(B,tree(∅,1,S))) =tree(tree(∅,C,∅),1,T).
```

4. Meta-Unification for Conditional Equational Theories with Constructors

We present a nondeterministic equational algorithm for meta-unification with constructors and its property.

4.1. Conditional Equational Theories with Constructors

By separating constructors in the definition of data types, we have the signature Σ partitioned into $C \uplus D$. We call operators in C the *constructors*. (We assume there are at least one constant constructors.) A *constructor term* is a term on C. The set of all constructor terms is denoted by $\mathcal{T}_C$ and the set of all ground constructor terms is denoted by $\mathcal{G}_C$. A *semi-constructor term* is either a variable or a term whose root function symbol is a constructor.

The *conditional equational theory* $\mathcal{E}$ *corresponding to* a Talos program P is a conditional equational theory with proper axioms as follows.

$\gamma = \delta$ for all "$\gamma = \delta$" in the definition of data types
$\gamma_1 = \delta_1 \wedge \gamma_2 = \delta_2 \wedge \cdots \wedge \gamma_m = \delta_m \supset \gamma = \delta$
for all "$\gamma = \delta$ **where** $\gamma_1 = \delta_1, \gamma_2 = \delta_2, \ldots, \gamma_m = \delta_m$" in the definition of functions

The *conditional term rewriting system* $\mathcal{R}$ *corresponding to* a Talos program P is a conditional term rewriting system as follows.

$\gamma \to \delta$ for all "$\gamma = \delta$" in the definition of data types
$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \to \delta$
for all "$\gamma = \delta$ **where** $\gamma_1 = \delta_1, \gamma_2 = \delta_2, \ldots, \gamma_m = \delta_m$" in the definition of functions

Example 4.1. The conditional euqational theory and the conditional term rewriting system corresponding to the definitions of the data types *boole*,*number*,*tree* and the function *insert* have four constructors zero 0,successor function *suc*, empty tree $\emptyset$ and tree constructor *tree*.

We assume $\mathcal{E}$ and $\underline{\mathcal{R}}$ satisfy the following three conditions.

(A) $\underline{\mathcal{R}}$ is confluent and terminating.
(B) Every left hand side of the head equation in $\mathcal{E}$ (or the head rewriting rule in $\mathcal{R}$) is not a semi-constructor term.
(C) For any ground term $s \in \mathcal{G}$, there exists a ground constructor term $t \in \mathcal{G}_C$ such that $s \to^* t$.

The condition (B) implies for every ground constructor terms $s, t \in \mathcal{G}_C$ we have $s =_{\mathcal{E}} t$ only if $s \equiv t$. The condition (C) with the condition (B) guarantees that the initial algebra of $\mathcal{E}$ is isomorphic to $\mathcal{G}_C$.

Remark. Several sufficient syntactical conditions of (A) are investigated. But the sufficient condition "left-linear and nonoverlapping" for the usual term rewriting systems ([11],[12]) is no longer sufficient and we need more explorations. A sufficient syntactical condition of (C) for usual term rewriting systems is investigated in [13].

4.2. Meta-Unification with Constructors

Now let us consider a conditional equational theory $\mathcal{E}$ with constructors and $\mathcal{E}$-unification of a set of equations $\mathcal{E}_0$. The logical and operational narrowing and meta-unifiability are defined similarly to those in 2.2 and we use the same notation. By combining the well-known equational unification algorithm and the Fay-Hullot's meta-unification algorithm, we obtain

a nondeterministic equational algorithm for metaunifications with constructors as follows.

meta-unify$_c$($\mathcal{E}_0$:set of equations) : substitution;
$\theta := <>$;
while $\mathcal{E}_0 \neq \emptyset$ **delete** one of the equations in $\mathcal{E}_0$
  when the equation is of the form $X = X$
    do nothing.
  when the equation is of the form $X = t$ or $t = X$ (X does not occur in t)
    apply variable-elimination to X and t.
  when the equation is of the form $s = t$ (either s or t is a non-variable term)
    if root function symbols are different constructors
    then stop with failure
    else apply term-reduction to s and t
endwhile
return θ.

variable-elimination(X:variable,t:term);
  let σ be a renaming of variables in t away from W and τ be $< X \Leftarrow \sigma(t) >$;
  apply $\tau \circ \sigma$ to $\mathcal{E}_0$; $\theta := (\tau \circ \sigma) \circ \theta$; $W := W + I(\tau)$
term-reduction(s,t:term);
  when s and t are of the form $f(s_1, s_2, \ldots, s_m)$ and $f(t_1, t_2, \ldots, t_m)$ (f is a constructor);
    add $s_1 = t_1, s_2 = t_2, \ldots s_m = t_m$ to $\mathcal{E}_0$
  when $NS_{pre}(s, W) \neq \emptyset$
    select $\sigma \in NS_{pre}(s, W)$ and let the corresponding conditional rule be
    "$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$" ($\sigma(\gamma) \equiv \sigma(s/u)$);
    $W := W + I(\sigma)$; let τ be meta-unify$_c$($\sigma(\{\gamma_1 = \delta_1, \gamma_2 = \delta_2, \ldots, \gamma_m = \delta_m\})$);
    if there exists a variable $X \in W$ for which $\tau \circ \sigma(X)$ is not in $\underline{R}$-normal form
    then stop with failure
    else add $s[u \Leftarrow \delta] = t$ to $\mathcal{E}_0$; apply $\tau \circ \sigma$ to $\mathcal{E}_0$; $\theta := (\tau \circ \sigma) \circ \theta$;
  when $NS_{pre}(t, W) \neq \emptyset$
    select $\sigma \in NS_{pre}(t, W)$ and let the corresponding conditional rule be
    "$\gamma_1 \downarrow \delta_1 \wedge \gamma_2 \downarrow \delta_2 \wedge \cdots \wedge \gamma_m \downarrow \delta_m \supset \gamma \rightarrow \delta$" ($\sigma(\gamma) \equiv \sigma(t/v)$);
    $W := W + I(\sigma)$; let τ be meta-unify$_c$($\sigma(\{\gamma_1 = \delta_1, \gamma_2 = \delta_2, \ldots, \gamma_m = \delta_m\})$);
    if there exists a variable $X \in W$ for which $\tau \circ \sigma(X)$ is not in $\underline{R}$-normal form
    then stop with failure
    else add $s = t[v \Leftarrow \delta]$ to $\mathcal{E}_0$; apply $\tau \circ \sigma$ to $\mathcal{E}_0$; $\theta := (\tau \circ \sigma) \circ \theta$;
  otherwise
    stop with failure

Figure 4.2. Equational Meta-Unification with Constructors

Example 4.2. Let s be $insert(A, insert(B, tree(\emptyset, 1, S)))$ and t be $tree(tree(\emptyset, C, \emptyset), 1, T)$. The meta-unification process proceeds similarly to Example 3.2.2 except peeling off root constructors and generating simultaneous equations. For example,if the narrowing to $insert(A, tree(insert(0, \emptyset), 1, S_1))$ in the second repetition is applied at the root using the fourth rule, we have three equations

$insert(0, \emptyset) = tree(\emptyset, C, \emptyset)$, $1 = 1$, $insert(suc(suc(A_2)), S_2) = T$.

Remark : Note that the "occur check" defers some binding. For example,when the equation selected from $\mathcal{E}_0$ is $X = [car([A|X])|Y]$ ($[A|X]$ is a list with head A and tail X), this equation is forced to be transformed to $X = [A|Y]$ in the third **when** once, becaues X occurs in $[car([A|X])|Y]$.

4.3. Consistency and Ground Completeness

The algorithm in 4.2 is a specialization of the Fay-Hullot's algorithm. But it is too special to keep its general completeness. For example,consider the meta-unification of $\{insert(A, S) = insert(A, T)\}$. Our algorithm won't generate the meta-unifier $< S \Leftarrow W, T \Leftarrow W >$. Nevertheless it still keeps enough completeness to guarantee the equivalence of the operational semantics and the model theoretic semantics of Talos. Before explaining it,we prepare three lemmas.

The first lemma says that once a prefix (occurrences near the root) of a term is filled with constructors in any $\rightarrow$-derivation issuing from a term t_0 to a term t_n in our conditional rewriting system, the fuction symbols at these occurrences in t_n are determined. (This lemma justifies the peeling off of constructor symbols at root in the equational meta-unification algorithm with constructors in 4.2.)

Lemma 4. Let $\mathcal{E}$ and $\mathcal{R}$ be a conditional equational theory and a conditional term rewriting system satisfying the three conditions in 4.1,

$t_0 \rightarrow_{[u_0, k_0, \nu_0 \circ \mu_0]} t_1 \rightarrow_{[u_1, k_1, \nu_1 \circ \mu_1]} \cdots \rightarrow_{[u_{n-1}, k_{n-1}, \nu_{n-1} \circ \mu_{n-1}]} t_n$

be any $\rightarrow$-derivation issuing from t_0 and ending with t_n. If root function symbols of t_i/v are constructor symbols for all $v \prec v_0$, then $u_j \not\prec v_0$ for all $i \geq j$ and the root function symbols of t_i/v and t_n/v are identical for all $v \prec v_0$.

Proof. We prove the lemma by structural induction on t_n. Let $t_i \equiv f(s_1, s_2, \ldots, s_m)$ be the first term in the $\rightsquigarrow$-derivation whose root symbol f is a constructor. Because of the condition (B) in 4.1, the succeeding narrowings never occur at the root. Hence $t_n \equiv f(r_1, r_2, \ldots, r_m)$, $s_1 \rightarrow^* r_1$, $s_2 \rightarrow^* r_2$, $\ldots$, $s_m \rightarrow^* r_m$, and $r_1, r_2, \ldots, r_m$ are all smaller than t_n. Hence from induction hypothesis,the lemma holds.

We specialize the concept "projectable" in 2.3 to ground cases. The *meta-unify*$_c$ is said to be *ground projectable* when all terms $t_0, t_1, \ldots, t_n$ in the $\rightsquigarrow$-derivation (1) in 2.3 in the definition of "projectability" are ground. (Hence it is a $\rightarrow$-derivation.)

Lemma 5. When $\underline{\mathcal{R}}$ and $\mathcal{E}$ satisfy the conditions in 4.1,

(a) $\theta(s_1) \downarrow \theta(t_1), \theta(s_2) \downarrow \theta(t_2), \ldots, \theta(s_m) \downarrow \theta(t_m)$ hold for $\underline{\mathcal{R}}$ when a substitution θ is generated by the *meta-unify*$_c$ for $\mathcal{E}_0 = \{s_1 = t_1, s_2 = t_2, \ldots, s_m = t_m\}$.

(b) The *meta-unify*$_c$ can generates a substitution θ such that $\theta \preceq_{\underline{\mathcal{R}}} \rho[\![V]\!]$ for any substitution ρ if $\rho(s_1) \downarrow \rho(t_1), \rho(s_2) \downarrow \rho(t_2), \ldots, \rho(s_m) \downarrow \rho(t_m)$ hold for $\underline{\mathcal{R}}$ and ρ instanciates $\mathcal{E}_0 = \{s_1 = t_1, s_2 = t_2, \ldots, s_m = t_m\}$ to a set of ground equations, where $V = \mathcal{V}(\mathcal{E}_0)$.

Proof. (a) is trivial. As to (b),the $\Rightarrow$-part (a) in lemma 1 must holds when the first term is ground. We only need to consider the ground projectability due to the following three facts.

(a) Let η be a $\underline{\mathcal{R}}$-normalized meta-unifier of s and t such that $\eta(s)$ and $\eta(t)$ are ground. Because of the condition (C) in 4.1. there is a $\rightarrow$-derivation (1) in 2.3 which issues from a ground term $h_2(\eta(s), \eta(t))$ and ends with a ground constructor term $h_2(r, r)$. When $\overline{\eta}$ is a substitution which instanciates all variables in the $\rightarrow$-derivation to any $\underline{\mathcal{R}}$-normal ground

terms. Then $\overline{\eta} \circ \eta$ is a $\underline{R}$-normalized meta-unifier of s and t and $\overline{\eta} \circ \eta =_{\underline{R}} \eta[\![\mathcal{V}(s) \bigcup \mathcal{V}(t)]\!]$.

(b) Consider a $\rightsquigarrow$-derivation which is for operational meta-unification of the condition part of a ground reduction and gives a meta-unifier ν. When it starts from $h_2(s,t)$ and $\overline{\nu}$ is a substitution which instanciates all variables in the first term $\eta(h_2(s,t))$ to $\underline{R}$-normal ground terms, $\overline{\nu} \circ \nu$ is also a meta-unifier of the condition part. Because it is for the condition part of the ground reduction, $\overline{\nu}$ has no effect to the original ground reduction.

(c) Two ground terms s and t are reduced to a same ground constructor term if $s =_{\underline{R}} t$ because of the condition (C) in 4.1. In such a case, the $\emptyset$-unifier in the last step of the Fay-Hullot's algorithm can be computed equationally in our algorithm. (Note that our algorithm can only compute most general $\emptyset$-unifiers of two terms when root function symbols of corresponding nonvariable subterms are identical constructors. We can't compute $\emptyset$-unifiers in general, while the Fay-Hullot's algorithm does it directly in the first **when**.)

The proof goes similarly to those of Lemma 1 and Lemma 2 except the peeling off of constructor symbols in term-reduction. This manipulation is justified by lemma 4, i.e. once root symbols of $t_i/1v$ and $t_i/2v$ are identical constructor for all $v \prec v_0$ in an operational $\rightarrow$-derivation, they are determined and thereafter there occurs no reduction inside v_0,i.e., $u_j \not\prec v_0$ if $i \leq j$. Hence,the correspondence between ground operational $\rightarrow$-derivations and operational $\rightsquigarrow$-derivations is not lost even with the peeling off. We omit the details due to space limit.

Lemma 6. When $\underline{R}$ is confluent and terminating, there exists a ground operational reduction $s \rightarrow t$ iff there exists a ground logical reduction $s \rightarrow t$.

Proof. The proof goes similarly to that of Lemma 3 except the peeling off of constructor symbols in term-reduction. The correspondence between ground operational reduction and ground logical reduction is not lost even with the peeling off. Due to space limit,we also omit other details.

A set of $\mathcal{E}$-unifier U is called a ground complete set of $\mathcal{E}$-unifiers of a set of equations $\mathcal{E}_0$ away from W iff it satisfies the condition (a) and (b) and a modification of (c) in 3.1.3.

(a) $\forall \theta \in U \ (\mathcal{D}(\theta) \subseteq V \ \& \ \theta$ is away from $W)$.

(b) $U \subseteq U_{\mathcal{E}}(\mathcal{E}_0)$.

(c) $\forall \sigma \in U_{\mathcal{E}}(\mathcal{E}_0)\,(\forall s = t \in \mathcal{E}_0(\sigma(s) \in \mathcal{G} \wedge \sigma(t) \in \mathcal{G}) \supset \exists \theta \in U \ \theta \preceq_{\mathcal{E}} \sigma[\![V]\!])$

Theorem 2 (Consistency and Ground Completeness) When $\underline{R}$ is confluent and terminating,

$U(\mathcal{E}_0, W_0) = \{\theta|V \mid$ *meta-unify*$_c(\mathcal{E}_0)$ with initialization $W := W_0$ stops with answer $\theta\}$

is a ground complete set of $\mathcal{E}$-unifiers of $\mathcal{E}_0$ away from W_0, where $W_0 \supseteq V = \mathcal{V}(s) \bigcup \mathcal{V}(t)$.

Proof. The theorem is,so to say,the consistency and ground completeness w.r.t. logical reduction. The proof goes in the completely same way as Theorem 1.

Remark. Again the *meta-unify*$_c$ is not complete even if $\overline{R}$ is confluent and terminating. The example in the remark of 2.3 is with two constructor zero 0 and successor function *suc*.

5. Semantics of Talos

5.1. Operational Semantics

A query of the form

?-t **when** $s_1 = t_1, s_2 = t_2, \ldots, s_m = t_m$.

is a request to prove

$\exists X_1, X_2, \ldots, X_n\ (s{=}t \wedge s_1 {=} t_1 \wedge s_2 {=} t_2 \wedge \cdots \wedge s_m {=} t_m)$

for some constructor term s, where $X_1, X_2, \ldots, X_n$ are all variables in the conjunction of equations.

When the Talos interpreter receives a query "?- t **when** $s_1 = t_1, s_2 = t_2, \ldots, s_m = t_m$.", it generate a set of equations $\mathcal{E}_0 = \{!Value = t, s_1 = t_1, s_2 = t_2, \ldots, s_m = t_m\}$, where $!Value$ is a special variable $Value$ annotated by "!". (Variables annotated by "!" are called *eager variable*, while those without it are called *lazy variable*.) Then it compute a meta-unifier θ of $\mathcal{E}_0$ away from $\mathcal{V}(\mathcal{E}_0)$ nondeterministically and returns $\theta|\mathcal{V}(\mathcal{E}_0)$ as the result. The meta-unifier is extended to treat the distinction of lazy and eager variables. It behaves in a manner similar to one in 4.2 except variable-elimination as follows.

execute($\mathcal{E}_0$:set of equations) : substitution;
$\theta := <>$;
while $\mathcal{E}_0 \neq \emptyset$ **delete** one of the equations in $\mathcal{E}_0$
  when the equation is of the form $X = X$ or $!X = !X$
    do nothing.
  when the equation is of the form $X = t$ or $t = X$ (X does not occur in t)
    apply lazy-variable-elimination to X and t
  when the equation is of the form $!X = t$ or $t = !X$ (t is a semi-constructor term)
    apply eager-variable-elimination to $!X$ and t
  when the equation is of the form $s = t$ (either s or t is a non-variable term)
    if root function symbols are different constructors
    then stop with failure
    else apply term-reduction to s and t
endwhile
return θ.

lazy-variable-elimination(X:lazy variable,t:term);
  let σ be a renaming of variables in t away from W and τ be $< X \Leftarrow \sigma(t) >$;
  apply $\tau \circ \sigma$ to all equations in $\mathcal{E}_0$; $\theta := (\tau \circ \sigma) \circ \theta$; $W := W + I(\tau \circ \sigma)$
eager-variable-elimination(!X:eager variable,t:term);
  when t is a variable Y (either lazy or eager)
    let $< Y \Leftarrow !Z >$ be a renaming of the variable Y away from W
    and $< !X \Leftarrow !Z >$ be a renaming of the variable $!X$ away from W;
    apply $< !X \Leftarrow !Z, Y \Leftarrow !Z >$ to all equations in $\mathcal{E}_0$;
    $\sigma := < !X \Leftarrow !Z, Y \Leftarrow !Z > \circ \sigma$; $W := W + \{!Z\}$
  when t is $f(t_1, t_2, \ldots, t_m)$ (f is a constructor)
    add $!X_1 = t_1, !X_2 = t_2, \ldots, !X_m = t_m$ to $\mathcal{E}_0$;
    apply $< X \Leftarrow f(!X_1, !X_2, \ldots, !X_m) >$ to all equations in $\mathcal{E}_0$;
    $\sigma := < !X \Leftarrow f(!X_1, !X_2, \ldots, !X_m) > \circ \sigma$; $W := W + \{!X_1, !X_2, \ldots, !X_m\}$
    ($!X_1, !X_2, \ldots, !X_m$ are fresh eager variables)

Figure 5.1. Talos Interpreter

Example 5.1. When the Talos interpreter receives a query

?- C **when** $insert(A, insert(B, tree(\emptyset, 1, S))) = tree(tree(\emptyset, C, \emptyset), 1, T)$.

one of the answers is

!Value=0,
A=A_4+2,
B=0,
C=0,
S=S_4,
T=insert(A_4+2,S_4);

by meta-unifying $\{!Value = C, insert(A, insert(B, tree(\emptyset, 1, S))) = tree(tree(\emptyset, C, \emptyset), 1, T)\}$, while the corresponding answer is

!Value=[0,tree(∅,2,∅)],
A=2,
B=0,
C=0,
S=∅,
T=tree(∅,2,∅);

when the query is

?- $[C, T]$ **when** $insert(A, insert(B, tree(\emptyset, 1, S))) = tree(tree(\emptyset, C, \emptyset), 1, T)$.

because it meta-unifies $\{!Value = [C, T], insert(A, insert(B, tree(\emptyset, 1, S))) = tree(tree(\emptyset, C, \emptyset), 1, T)\}$ and "!" is propagated to force evaluation of T. Note that the Talos interpreter needs nondeterministic search, though data types and functions themselves are deterministic.

Remark. So far we have described a nondeterministic algotithm. In its sequential implementation, we make the meta-unification a self-recursive program and search meta-unifiers using Prolog-like backtracking. We need to choose an appropriate equation in the meta-unification. We keep a set of simultaneous equations in a stack. The equation at the top is popped at each meta-unification call and processed. It is either simply erased (possibly with some application of substitution) or it generates several new equations and the generated equations are pushed at the top. Then new meta-unification is called with the new stack. We also need to choose an appropriate pre-narrowing substitution σ in term-reduction. Before choosing it, we set the backtracking point there. Depending on the choice of the occurrence at which narrowing is applied, we can integrate various evaluation strategies into Talos such as call-by-value,call-by-name,call-by-need,lazy evaluation and eager evaluation. Once some occurrence is chosen, the rewriting rules in the definition are tried from top to bottom following the control of Prolog. When we encountered "stop with failure", we backtrack to the latest backtracking point and try another alternatives.

5.2. Model Theoretic Semantics

The model theoretic semantics of Talos is defined by the set M_0 of all ground equations valid in all models of $\mathcal{E}$ (cf.[6]). This formulation is located between the initial algebra for algebraic data type specifications (cf. Goguen and Meseguer [8]) and the minimum Herbrand model for Prolog (van Emden and Kowalski [4]).

The fixpoint semantics of Talos is defined similarly to that of Prolog. Let T be a transformation of sets of ground equations associated with the set $\mathcal{E}$ of quational definite clauses defined by

$$T(I) = \{s = t \mid s \text{ and } t \text{ is a ground term } \& \text{ there is some ground instance } s_1 = t_1 \wedge s_2 = t_2 \wedge \cdots \wedge s_m = t_m \supset s = t \text{ such that } s_1 = t_1, s_2 = t_2, \ldots, s_m = t_m \text{ are all in } I\}$$

Then it is obvious that M_0 is the least fixpoint of T and $\bigcup_{i=0}^{\infty} T^i(\emptyset) = M_0$ (cf.[6]).

Example 5.2. A Talos program P consisting of the definitions of the data types *boole,number,tree* and the function *insert* defines all the set of equations M_0 holding between ground terms denoting boole,number and tree in our common sense. The quotient $\mathcal{G}/M_0$ is isomorphic to the set of ground constructor terms denoting *boole,number* and *tree*.

5.3. Equivalence of Two Semantics

Now we prove the most important theorem for the semantics of Talos.

Equivalence Theorem
When $\mathcal{E}$ and $\mathcal{R}$ satisfy the conditions in 3.1, ?-t stops and returns a ground constructor term s satisfying $s =_{\mathcal{E}} t$ for any ground term t.

Proof. Because the "!" annotation is propagated only to force instanciation, it answers correctly when it stops. Moreover it is ovbious that this annotation does not prevent any computation of ground meta-unifiers from termination, because any ground term is reducible to a ground constructor term.

6. Discussions

Several attempts have been done to amalgamate relational programming languages and functional programming languages. Bellia [2] introduced Horn clauses with equalty into relational program, but their language is substantially completely deterministic functional programming language. Fribourg [6] used equational Horn clause and clarified its semantics based on the paramodulation, which is very similar to the general narrowing. But because he did not impose any conditions (like confluence and termination), his completeness theorem needed superposition between programs and additional functional reflexive axioms. Moreover the narrowings was not restricted to those at occurrences of non-variable terms. Tamaki [20] introduced a reducibility predicate into Prolog and defined its semantics based on source-level expansion of nested terms to conjunction of atoms. Because he did not impose the termination condition, he had to add the reflexivity of $\rightarrow^*$ to the expanded programs, which plays a very important role. Goguen and Meseguer [8] suggested the use of narrowing in computation in their functional-relational language Eqlog based on rigouros logical basis of many sorted logic [9]. They allowed general algebraic specification of abstract data types and used the general narrowing.

We claim our contributions in this paper are the following two. First, our equivalence theorem is a one-step advance towards the completeness of more general languages using the narrowing such as Eqlog [8] and SLOG [6]. (The completeness of "SLD-resolution + meta-unification" in Eqlog is left open. See comment in [8] pp.206-207 and [6] p.173). Secondly,our framework makes the programming reasonably easy as well as the interpreter reasonably efficient. Eqlog's general data type specification indeed gives high expressive power. Though Talos lacks such generality, existence of constructors is helpfull not only for programmers but also for the meta-unification process. To programmers who uses such languages as *programming* languages, it gives concrete symbolic objects to manipulate and conceive easily in mind. From the meta-unification process, it alleviates the too frequent check of unifiability and enables us to compare corresponding terms only when they are semi-constructor term. (Note that we always have to compare corresponding terms at the first when in the Fay-Hullot's algorithm. cf. comment in [8] p.206). The constructor terms

in Talos exactly do play the same role as general terms in Prolog do.

We have shown only one of the feature of Talos,i.e. conditional computation. Talos has another three prominent features, nondeterministic computation, "call by need" computation [14],[7],[10] and computation with stream [1]. The first version of Talos was implemented in MACLISP from April in 1982 to March in 1983. The language features and implementation details are explained in another paper [17].

7. Conclusions

We have presented a computation mechanism "computation by meta-unification with constructors" stemed from the behavior of an interpreter of an equational programming language Talos.

Acknowledgements

The author would like to express deep gratitude to Dr.Joseph.A.Goguen (SRI International and Center for the Study of Language and Information,Stanford University) for kindly informing of Eqlog and Dr.Kō Sakai (ICOT 2nd Laboratory) for pointing out errors in the previos equational metaunification algorithm.

This work is based on the activity of TRS WG (Term Rewriting Systems Working Group) for the FGCS(Fifth Generation Computer System) project. The author would like to thank to Dr.K.Fuchi (ICOT Director) and Dr.S.Yokoi (Chief of ICOT 2nd Laboratory) for the opportunity of doing this research.

References

[1] Ashcroft,E.A. and W.W.Wadge, "Lucid,a Nonprocedural Language with Iteration", C.ACM, Vol.20, No.7, pp.519-526, 1977.

[2] Bellia,M.,P.Degano and G.Levi, "The Call-by-Name Semantics of a Clause Language with Functions", in Logic Programming (K.L.Clark and S-Å.Tärnlund Eds), pp.281-295, 1982.

[3] Burstall,R.M.,D.B.MacQueen and D.T.Sannela, "HOPE:An Experimental Applicative Language", Proc.of 1980 LISP Conference, pp.136-143, 1980.

[4] van Emden,M.H. and R.A.Kowalski, "The Semantics of Predicate Logic as Programing Language", J.ACM, Vol.23, pp.733-742, 1976.

[5] Fay,M., "First-order Unification in Equational Theory", 4th Workshop on Automated Deduction, pp.161-167, 1979.

[6] Fribourg,L., "Oriented Equational Clauses as A Programming Language", J.Logic Programming, Vol.1, No.2, pp.165-177, 1984.

[7] Friedman,D., and Wise,D., "CONS Should Not Evaluate Its Arguments", Automata,Language and Programming, Edinburgh University Press, pp.257-284, 1976.

[8] Goguen,J.A.and J.Meseguer, "Equality,Types,Modules and (Why Not?) Generics for Logic Programming", J.Logic Programming, Vol.1, No.2, pp.179-210, 1984.

[9] Goguen,J.A.and J.Meseguer, "Completeness of Many-Sorted Equational Logic", SIGPLAN Notices 16, 7, pp.24-32, 1981. Also apperaed in SIGPLAN Notices 17, 1, pp.9-17; extended version SRI Technical Report, 1982, and to be published in Houston Journal of Mathematics.

[10] Henderson,P. and J.H.Morris, "A Lazy Evaluator", Conference Record of 3rd ACM Symposium on Principles of Programming Languages, pp.95-103, 1976.

[11] Hoffman,M. and M.J.O'Donnell, "Programming with Equations", ACM TOPLAS, Vol.4,

No.1, pp.83-112, 1982.
[12] Huet,G., "Confluent Reduction : Abstract Properties and Applications to Term Rewriting System", J.ACM, Vol.27, No.4, pp.797-821, 1980.
[13] Huet,G. and J-M.Hullot, "Proofs by Inductions in Equational Theories with Constructors", J.Computer and System Science, 25, pp.239-266, 1982.
[14] Huet,G. and J-J.Levy, "Call by Need Computations in Non-Ambiguous Linear Term Rewriting Systems", INRIA Research Report 359, August, 1979.
[15] Huet,G. and D.C.Oppen, "Equations and Rewrite Rules : a Survey", in Formal Language Theory : Perspectives and Open Problems (R.V.Book Ed), pp.349-405, Academic Press, 1980.
[16] Hullot,J-M., "Canonical Forms and Unification", in 5th Conference on Automated Deduction (W.Bibel and R.A.Kowalski Eds), pp.318-334, 1980.
[17] Kanamori,T., "Language Features of A Meta-Unification Based Language Talos", ICOT Technical Report, 1985.
[18] Pereira,L.M.,F.C.N.Pereira and D.H.D.Warren, "User's Guide to DECsystem-10 Prolog", Occasional Paper 15,Dept.of Artificial Intelligence, Edinburgh, 1979.
[19] Sato,M.and T.Sakurai, "Qute : A Functional Language Based on Unification", Proc.of International Conference on Fifth Generation Computer System 1984, pp.157-165, 1984.
[20] Tamaki,H., "Semantics of A Logic Programming Language with Reducibility Predicate", Proc.of 1984 International Symposium on Logic Programming, pp.259-264,1984.
[21] Turner,D.A., "The Semantic Elegance of Applicative Languages", Proc. of 1981 Conference on Functional Programming Languages and Computer Architectures, pp.85-92, 1982.