ICOT Technical Report: TR-655

TR-655

Record Algebra Model for Feature Structures

by
K. Mukai

June, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Record Algebra Model for Feature Structures*

Kuniaki Mukai

Institute for New Generation Computer Technology (ICOT)
Mita Kokusai Bldg. 21F, 1-4-28 Mita, Minato-ku, Tokyo 108 JAPAN
Tel: +81-3-456-3069, E-mail: mukai@icot.or.jp

Abstract

A new class of algebras called record algebras is introduced as a mathematical model for feature structures. A record algebra is commutative and idempotent partial monoid R provided with an operator domain G. G is a monoid whose elements called features act on R from both left and right sides of R. R is called a G record algebra and G a feature monoid. The record algebra is an extension of the feature algebra in computational linguistics. A constraint theory $(R, \bowtie)$ is given for complete and standard record algebras R with a record compatibility relation $\bowtie$. The standard unification theory $(H, =)$ on Herbrand universe H with the identity relation $=$ is embedded into $(R, \bowtie)$. Also based on the theory $(R, \bowtie)$ a logic programming over $(R, \bowtie)$ is defined with both declarative and operational semantics, where the maximum semantics is used for the declarative semantics. It is shown that both semantics of the program are sound and complete including negation–as–failure rule. As an application of this logic programming language class the definite clause grammar over H is generalized to that over G–record algebras.

1 Introduction

In this paper[1], we propose a class of algebras called record algebras as a mathematical model of feature structures. In fact this structure was first introduced into the standard logic programming language Prolog as an extension of the first order term and it turned out to be very useful for natural language processing[20, 16]. We develop a unification theory using parametric records taking them as a partial description of pure records, i.e. parameter–free ones. We write $(R, \bowtie)$ informally for the proposed record unification theory, where R is the domain of pure records and $\bowtie$ a binary relation on R. We develop the theory $(R, \bowtie)$ so that it is a straightforward extension of the standard unification theory $(H, =)$ over Herbrand universe H, i.e. the domain of pure first order terms. In fact the former will be proved to be a conservative extension of the latter. Note that we treat only one relation $\bowtie$ as the first order unification theory does only the identity relation $=$.

Intuitively records are simple recursive structures of the form $\{(a_1, r_1), \cdots, (a_n, r_n)\}$, where a_i are distinct *atomic features* and r_i are atoms or possibly another records. r_i are

*To appear in J. Wedekind and C. Rohrer (eds.), Unification in Grammar, MIT Press, 1991.
[1]This work is a substantial revision of [18].

also called *features*. A formal definition will be given in section 2 to the record. We treat three familiar kinds of builtin operations on records: 'merge'(+), 'left'(·), and 'right' (//).

Example 1 a, b, c, d are features, r, s, t are atoms.

+ (merge)	: $\{(a,r),(b,s)\}+\{(b,s),(c,t)\}$	=	$\{(a,r),(b,s),(c,t)\}$.	
· (left)	: $a\cdot\{(b,r),(c,s)\}$	=	$\{(a,\{(b,r),(c,s)\})\}$.	(See figure 1.)
// (right)	: $\{(a,\{(b,r),(c,s)\}),(d,t)\}//a$	=	$\{(b,r),(c,s)\}$.	(See figure 2.)

□

Note that operations are partial in general. For instance $\{(a,p)\}+\{(a,q)\}$ is undefined when p and q are distinct 'atoms'. Using this algebra each record $\{(a_1,r_1),\cdots,(a_n,r_n)\}$ $(n\geq 1)$ is expressed as $a_1r_1+\cdots+a_nr_n$, where ax stands for $a\cdot x$.

With these simple examples in mind a *G-record algebra* will be formally defined to be 6-tuple $(R,G,+,\cdot,//,\epsilon)$, where R is a commutative and idempotent partial monoid under + with an operator domain G which acts on R from both left(·) and right (//) sides as a monoid. We call G a feature monoid and R a G-record algebra. An element of R is a *record*. An element of G is a *feature*. ϵ is the *unit* of R with respect to +. As a convention for the case $n=0$ above we identify the empty record $\emptyset$ with ϵ.

We have no type other than the record. Data such as numbers and strings which come at terminals of record structures are *atomic* records. A standard example of records are trees which have tags at only leaf nodes. Let R be the set of such trees. A feature is a path in the tree starting from the root node. The set of features forms a monoid under the concatenation. $x//\alpha$ denotes the subtree of the tree x which can be accessed along the feature α in x. αx denotes the tree which is obtained by putting the tree x at the end of the feature α. + is a tree merge. ϵ is a singleton tree which is not assigned a tag. Unlike the standard unification theory $(H,=)$ record algebras R involve partiality. In fact R is a partial semi-lattice structure w.r.t. (with regard to) $\leq$, where $a\leq b$ is defined by $a+b=b$.

Atomic constraints on records are of the form $p\bowtie q$, where p and q are record terms. The symbol $\bowtie$ means a binary relation on R representing compatibility of records, i.e. $u\bowtie v$ iff $u+v$ is defined in R, where $u,v\in R$. A *solution* of $p\bowtie q$ is an assignment f such that there is s a common 'instance' t in R of p and q. In other words $p\bowtie q$ is solvable iff there is a record $t\in R$ which is of parametric types both p and q. Also another informal reading of $p\bowtie q$ is that there is a record $t\in R$ such that p and q are a partial description of t. The concept of solutions in $(R,\bowtie)$ is a natural extension of that in the standard theory $(H,=)$. The notion of solutions in $(R,\bowtie)$ is fundamental and will be defined formally. The relation $\bowtie$ is not an equivalence relation. In fact there is no full transitive rule in the theory $(R,\bowtie)$ but only a restricted one, which only when x is a parameter the transitive rule

$$x\bowtie p\wedge x\bowtie q\Longrightarrow p\bowtie q$$

is applied in the unification process.

A *unification theory* of the record algebra can be seen as a closure operation on constraints. In fact a constraint is defined to be *unifiable* iff it has a consistent closure. The closure roughly is a generalization of the unifier. For example in $(R,\bowtie)$ the closure of the constraint $\{(a,y),(b,y)\}\bowtie\{(b,1),(c,y)\}$ contains $y\bowtie 1$, which means $y=1$ with 1 being atomic. We will show that for every constraint C on the record algebra R the constraint C is unifiable iff C is solvable in R. In fact our unification theory over records is satisfaction complete in the sense of the logic programming (CLP) scheme[12]. Thus the unification theory $(R,\bowtie)$ characterizes the set of solvable constraints in a decidable way. Also we show that the unification theory will be compact in the sense that a constraint is solvable iff so is its every finite subconstraint.

Now we turn to how to build the record algebra into logic programming. We view the logic programming over records as a form of inductive or coinductive definition for domains of records[1, 17]. So the semantics of the program is defined to be the maximum or minimum fixpoint of the program viewed as a monotone transformation on the domain R of records. In particular we are interested in nonwellfounded structures[1] such as streams for a variety of possible applications, e.g. type inference involving recursive type definitions among others. So unlike traditional semantics of logic programming we treat in this paper only the maximum semantics of programs[14].

Example 2 The program below consists of three Horn clauses for a recursive data type definition of list structures, where x and l are parameters, a, b, nil, $atom$, $list$ are atoms, $type$, car, cdr, and $form$ are features.

(1) $\{(type, atom), (form, a)\}$.

(2) $\{(type, atom), (form, b)\}$.

(3) $\{(type, list), (form, \{(car, x), (cdr, l)\})\}$:-
$\{(type, atom), (form, x)\}, \{(type, list), (form, l)\}$.

The minimum semantics of the program is the set

$$\{\{(type, atom), (form, a)\}, \{(type, atom), (form, b)\}\}.$$

On the other hand the maximum semantics of the program is the largest set M of records such that:

- $M = N \cup \{\{(type, atom), (form, a)\}, \{(type, atom), (form, b)\}\}$.
- N is the largest set such that for any $r \in N$, there are some $v \in M$ and $u \in \{a, b\}$ such that $r = \{(type, list), (form, \{(car, u), (cdr, v)\})\}$.

This kind of straightforward maximum semantics is given in [19] based on the hyperset theory [1]. This program example will be treated formally in example 11 and 12. □

We will show the soundness and completeness results and the display-theorem (theorem 17), which asserts that every solution is displayed by a computation for the given goal. Also we will show the soundness and completeness of the negation as failure rule. However our constraint language does not exactly fit to the CLP scheme. In fact as the atomic constraint $\epsilon \bowtie x$ holds for any record x the unit ϵ is not *constraint definable*, which means $(R, \bowtie)$ is not solution compact[12]. This aspect of $(R, \bowtie)$ may be easily modified by separating the role of the symbol $\bowtie$ into two, i.e. identity $(=)$ and subsumption $(\sqsubseteq)$ so that the modified language is solution compact[19]. This modification is, however, out of place.

There is a straightforward translation from Herbrand universe H into the record algebra R which maps, for instance, $f(a, b)$ to $\{(f_1, a), (f_2, b)\}$, where f_1 and f_2 are argument places of f. The first order term $f(a, g(x))$ for instance is translated to the record $\{(f_1, a), (f_2, \{(g_1, x)\})\}$. Partial descriptions $\{(f_1, a)\}$ and $\{(f_2, b)\}$ together mean the first order term $f(a, b)$, whereas there is no such term for $\{(f_1, a)\}$ and $\{(f_1, b)\}$ with a, b being distinct atoms. This kind of partiality is an essential aspect of records structure constraints which are not seen in the standard unification theory.

As applications of the unification theories $(R, \bowtie)$ we will show first that the standard domain of $(H, =)$ is embedded into a record algebra of $(R, \bowtie)$. Then we extend the DCG (definite clause grammar) over H to that over the record algebra R. Also we will see that DAGs used in unification grammar formalism are viewed constraints on records.

We will give the details of the unification theory $(R, \bowtie)$ over record algebras R which is a conservative extension of the standard unification theory $(H, =)$ over Herbrand universe

H. The standard term unification $f(x,x) = f(a,y)$ for instance in $(H,=)$ is translated to $\{(f_1,x),(f_2,x)\} \bowtie \{(f_1,a),(f_2,y)\}$ in $(R,\bowtie)$ preserving the solvability. The closure of the latter contains three atomic constraints: $x \bowtie a$, $x \bowtie y$, and $y \bowtie a$, which in fact means $x = y = a$. More formally speaking there is a translation $\tau: H \rightarrow R$ between the first order term unification (including infinite trees) $(H,=)$ and the proposed record unification theory $(R,\bowtie)$ such that $s = t$ is unifiable in $(H,=)$ iff $\tau(s) \bowtie \tau(t)$ is unifiable in $(R,\bowtie)$ and also that $s = t$ is solvable in $(H,=)$ iff $\tau(s) \bowtie \tau(t)$ is solvable in $(R,\bowtie)$. A non trivial thing is that the binary relation symbol $\bowtie$ between records is not interpreted as the record identity but as a kind of compatibility of two records[2]. In fact an extended notion of solutions for record constraints is needed to make the record unification complete as desired. Moreover there is a converse mapping from the record domain R to the standard term domain H in the sense that the record unification can be reduced to the standard term unification. For example take a compatibility constraint (1).

$$\{(a,y),(b,y)\} \bowtie \{(b,1),(c,y)\}. \quad (1)$$

$$\{(a,y),(b,y),(c,u)\} = \{(a,v),(b,1),(c,y)\}. \quad (2)$$

$$f(y,y,u) = f(v,1,y). \quad (3)$$

The compatibility constraint is reduced to the equational constraint (2) introducing new parameters u and v for 'hidden' features c and a. Clearly the two constraints (1) and (2) are equivalent in the sense that the equation is solvable iff so is the given constraint. Now the equation (2) is equivalent to the term equation (3), where f is a function symbol. The present work, however, concerns $\bowtie$–constraints themselves as a study of partiality of information without reducing them to standard constraints.

There are many related works on feature structures, which are still on going, so that we take only some of representative works among them related to this work and give brief notes on them from the view point of this work. First of all Pereira and Shieber[21] applies Scott's domain theory to give a denotational semantics to unification grammars viewing them as a computer program. Although it is not clear that the record algebra is an instance of the scheme both works share the basic idea in that grammars are computer programs. In fact the present work treats DCG as a form of coinductive definitions for a desired domain of records as legal feature structures.

As explained above by introducing new parameters for 'hidden' features the proposed record constraints can be translated into first order term equations or more generally those in the order sorted algebra (OSA)[10]. However it is the behavior of the relation $\bowtie$, i.e. a logic of 'compatibility' that the present work is interested in. In fact we give a record constraint for only $\bowtie$ relation and thereby we need no new parameters in unification procedures. Also the proposed record algebra has nonwellfounded structures in general and the semantics of programs is the maximum semantics. It is interesting but not so clear how the initial algebra semantics of OSA, for instance, can be applied to the record algebras.

PATR-II[24] is a standard computational framework for DAG–based unification grammars. Our unification over the record algebra is equivalent to that of PATR-II over feature structures, i.e. graph merging, though we includes infinite record structures. The proposed embedding of the standard domain $(H,=)$ into the record unification $(R,\bowtie)$ gives a mathematical model to a common sense view that Herbrand terms are a *degenerated form* of records and a DCG is a compiled form of unification grammar (Shieber). Also we will show a natural translation of DAGs into constraints in $(R,\bowtie)$ so that the unification theory over DAGs can be interpreted as a constraint theory over the record algebra in a

[2]Although the proposed theory of $\bowtie$ on records is not exactly an equational theory we still use the word 'unification' for such a theory based on a strong similarity to each other.

natural way. In other words records are considered to be denotations of directed graphs (DGs), which is an analogy to that hypersets are denotations of directed graphs[1].

Seeing records as partial functions the record algebra is related to situation semantics[6] and situation theory[3, 4]. That is, records can be used for representation of *state of affairs* which contains a partially specified list of arguments[16]. Pollard[22] proposed the notion of *anadic* relations for situation semantics. The record algebra seems to serve as a model for anadic relations.

In his thesis Aït-Kaci[2] criticizes first order terms from type theoretical point of view and proposes to see them as records. He uses semi–lattice for the framework. Aït-Kaci[2], however, gives no declarative semantics of the program over the semi lattice while we give a declarative semantics for constraint logic programming over record algebras.

Kasper and Rounds[13] proposes automata models of unification theory over feature structures. Also more recently Smolka[26] formalizes unification theories in feature logic with subsorts including negation and disjunction. He showed a linear time translation from constraint language over feature structures into a quantifier free sublanguage of first order predicate language.

Courcelle[9] treats infinite trees. Maher[15] gives an axiomatization of infinite trees. However, they treat only trees which have fixed arities. Also the same with case of Colmerauer[8] while the record has not a fixed arity.

This paper is organized as follows. Section 2 and section 3 are main part of the paper. In section 2 we introduce a class of record algebras and give a unification theory over them. The main goal is the equivalence theorem between the solvability and unifiability. In section 3 we give a class of unification grammars over record algebras being guided by the idea that grammars are computer programs[21]. Soundness and completeness results are obtained. DAGs and the notion of arity will be given a new interpretation respectively from the point of the record algebra. In section 4 we give concluding remarks.

2 Record Algebra and Unification

We introduce record algebras and describe a unification theory over them. As things in record algebras are partial we make a general convention for equations and evaluation in partial algebras used in the rest of the paper. By $e\downarrow$ we mean that the expression e is defined, i.e. has a value, where e is an expression in the constraint language. Details of 'logic of partial terms' will be given at appropriate places in the below. An equation $l \simeq r$ means that if either l or r is defined then both of them have the same value:

$$l \simeq r \iff [l\downarrow \vee r\downarrow \Longrightarrow l\downarrow \wedge r\downarrow \wedge l = r].$$

where l and r are expressions. The use of terminologies follows standard algebra text books such as [11, 28, 7].

2.1 Record Algebra

Definition 1 A *feature monoid* G is a monoid such that the following hold:

(1) Each $\alpha \in G$ has only a finite number of prefixes, where $\beta \in G$ *is a prefix of* $\alpha \in G$ if $\alpha = \beta\gamma$ for some $\gamma \in G$.

(2) $(G, \leq)$ is a partial order structure, where $\leq$ is a binary relation over G defined by $\alpha \leq \beta$ iff α is a prefix of β.

(3) $(G, \leq)$ forms a tree, i.e. the set of $\{\beta \in G \mid \alpha \leq \beta \leq \gamma\}$ is totally ordered by $\leq$ for any α, $\beta \in G$.

□

Elements of G are called a *feature*. Every free monoid is a feature monoid. We write $\alpha \not\sim \beta$ iff α and β are *incomparable* features, i.e. $\alpha \not\sim \beta \iff \alpha \not\leq \beta \wedge \beta \not\leq \alpha$ in G. α and β are *incompatible* iff there is no upper bound of α and β in G. As the unit $\varepsilon \in G$ is the minimum element of $(G, \leq)$ it follows from assumption (3) that α and β are incomparable in G iff they are incompatible in G.

The feature monoid is a generalization of monoids of strings over given letters with the string concatenation. Let L be a set of *atomic features*. A sequence of atomic features $a_1, \ldots, a_n$ is written $\langle a_1, \ldots, a_n \rangle$, where n is a non negative integer and is called the length of the sequence. As usual convention the sequence $\langle a_1, \cdots, a_n \rangle$ denotes the *empty string* $\langle\rangle$ when $n = 0$. Also ε denotes the empty string. The length of the empty string is 0. We write $\alpha\beta$ for the *concatenation* of two sequences α and β. The concatenation is defined by the following equations:

$$\begin{aligned} \varepsilon\alpha &= \alpha. \\ \alpha\varepsilon &= \alpha. \\ \langle a_1, \ldots, a_n \rangle \langle b_1, \ldots, b_m \rangle &= \langle a_1, \ldots, a_n, b_1, \ldots, b_m \rangle. \end{aligned}$$

The symbol X^* denotes the free monoid generated over a set X. For example with L being a set of letters L^* is the set of words of finite length over letters in L. For the convenience of notation the string $\langle a \rangle$ of the length one is written simply a. Clearly from the definition L^* is a feature monoid.

Remark Condition (3) above is narrow. It is desirable to find more general conditions for (3) in which a computational unification theory such as one developed in the present work is still effective. However this is an open problem.

Definition 2 Let G and M be sets and let $F \subseteq \{\cdot, +, /\!\!/\}$. $\mathcal{E}(G, M, F)$ is the least set E of *expressions* which satisfies the following.

(1) $M \subseteq E$.

(2) If $\cdot \in F, \alpha \in G, x \in E$ then $\cdot(\alpha, x) \in E$.

(3) If $/\!\!/ \in F, \alpha \in G$ and $x \in E$ then $/\!\!/(x, \alpha) \in E$.

(4) If $+ \in F$, $x, y \in E$ then $+(x, y) \in E$.

□

We use usual notations:

$$\begin{aligned} \alpha x &\stackrel{\text{def}}{=} \cdot(\alpha, x). \\ x /\!\!/ \alpha &\stackrel{\text{def}}{=} /\!\!/(x, \alpha). \\ x + y &\stackrel{\text{def}}{=} +(x, y). \end{aligned}$$

Given a record term p the set $\mathcal{V}(p)$ denotes the set of parameters appearing in p.

Definition 3 A *merge system* is a partial semi-group $(M, +)$, where M is an associative, commutative and idempotent under a partial binary operation $+ : M \times M \rightharpoonup M$. The axioms follows, where $a, b, c \in \mathcal{E}(\emptyset, M, \{+\})$.

Partiality	$a \in M$	$\Longrightarrow$	$a\downarrow$.
	$a + b\downarrow$	$\Longrightarrow$	$a\downarrow \wedge b\downarrow$.
Associative	$a + (b + c)$	$\simeq$	$(a + b) + c$.
Commutative	$a + b$	$\simeq$	$b + a$.
Idempotent	$a + a$	$\simeq$	a.

□

Note that a merge system may have not a unit.

Example 3 $(pow(P), \cup)$ is a merge system, where P is a set and $pow(P)$ denotes the set of subsets of P. □

$(M, +)$ is a *trivial merge system* if M is a set and $+$ is a binary operation on M such that $x + x = x$ but $x + y$ is undefined whenever $x \neq y$, where $x, y \in M$.

Definition 4 A *G-record algebra* is a 6-tuple $\mathcal{R} = (R, G, +, \cdot, /\!\!/, \epsilon)$, where

(1) $(R, +)$ is a merge system with the *unit* ϵ.

(2) G is a feature monoid which acts on R as an operator domain from both left and right:

$$\begin{aligned} \cdot &: G \times R \to R. \\ /\!\!/ &: R \times G \to R. \end{aligned}$$

(3) Partiality axioms for $\mathcal{R}$ follows, where $\alpha \in G$, and $a, b \in \mathcal{E}(G, R, \{\cdot, +, /\!\!/\})$:

$$\begin{aligned} &\Longrightarrow && \alpha\downarrow. \\ &\Longrightarrow && \epsilon\downarrow. \\ a + b\downarrow &\Longrightarrow && a\downarrow \wedge b\downarrow. \\ \alpha a\downarrow &\Longrightarrow && a\downarrow. \\ a /\!\!/ \alpha\downarrow &\Longrightarrow && a\downarrow. \end{aligned}$$

(4) The following equations hold, where $\alpha \in G$, $a, b, c \in \mathcal{E}(G, R, \{\cdot, +, /\!\!/\})$:

Associative	$a + (b + c)$	$\simeq$	$(a + b) + c.$	
Commutative	$a + b$	$\simeq$	$b + a.$	
Idempotent	$a + a$	$\simeq$	$a.$	
Unit	$a + \epsilon$	$\simeq$	$\epsilon + a$	$\simeq a.$
Left Distributive	$\alpha(a + b)$	$\simeq$	$\alpha a + \alpha b.$	
Right Distributive	$(a + b) /\!\!/ \alpha$	$\simeq$	$a /\!\!/ \alpha + b /\!\!/ \alpha.$	
Cancellation	$(\alpha a) /\!\!/ \alpha$	$\simeq$	$a.$	

□

Elements of R are called *records*. Note that actions by a feature are partial in general. We abuse R for $\mathcal{R}$. In the record algebra $x = y$ follows from $\alpha x = \alpha y$ by the cancellation axiom. For any $x \in R$ $\varepsilon x = x$ and in particular $\varepsilon\epsilon = \epsilon$ hold.

Definition 5 Records $x \in R$ are *atomic* if $x /\!\!/ \alpha$ is undefined for any $\alpha \in G \setminus \{\varepsilon\}$. □

The unit ϵ is not always atomic. Such an example will be found in example 4 and 5. We recall the definition of the feature algebra and show that the record algebra is an extension of the feature algebra. Roughly speaking a record algebra is a feature algebra which has an internal merge operation in addition to external ones. Moreover the operators operate on records from both sides not only as 'field selectors' but also 'record constructors' respectively, whereas operators in feature algebras work only as selectors.

Definition 6 Given a set F of *features* and C of *constants* a *feature algebra* is an ordered pair $(D, (\cdot)^{\mathcal{A}})$ which satisfies the following, where D is a set:

(1) $f^{\mathcal{A}}: D \to D$ is a partial function if $f \in F$.

(2) $c^{\mathcal{A}} \in D$ if $c \in C$.

(3) $a^{\mathcal{A}} \neq b^{\mathcal{A}}$ if $a, b \in C$, $a \neq b$.

(4) $a^{\mathcal{A}} \notin \mathrm{dom}(f) \in D$ if $f \in F$ and $a \in C$.

□

As distinct constants are interpreted to be distinct we can assume $C \subseteq D$. Assuming the hyperset theory [1] it is straightforward to show that every feature algebra is a record algebra.

Proposition 1 *Given a set F of features, C of constants, and a feature algebra $\mathcal{A} = (D, (\cdot)^{\mathcal{A}})$. Then there is a F^*–record algebra $\mathcal{R}_{\mathcal{A}} = (R, F^*, +, \cdot, /\!\!/, \epsilon)$ and an injection $\psi: D \to R$ such that the following hold.*

(1) $\psi(C) \subseteq R$.

(2) $\psi(f^{\mathcal{A}}(d)) \simeq \psi(d)/\!\!/ f \qquad (d \in D)$.

Proof For $d \in D$ let $S_d = \{(f, f^{\mathcal{A}}(d)) \mid f \in F, d \in \mathrm{dom}(f^{\mathcal{A}})\}$ and $D' = \{d \in D \mid S_d \neq \emptyset\}$. Solve the system $\{d = S_d \mid d \in D'\}$ of equations, where by the solution lemma [1] there is a unique solution ψ to the system. Let $R_0 = \{\psi(d) \mid d \in D'\} \cup D \setminus D'$ and ϵ be some new atom not in R. Let $R = R_0 \cup \{\epsilon\}$. Define a merge system $(R, +)$ so that $(R_0, +)$ is a trivial merge system and $\epsilon + x = x + \epsilon = x$ for all $x \in R$. Then define $\psi(x)/\!\!/ f \stackrel{\mathrm{def}}{=} \psi(f^{\mathcal{A}}(x))$ and $f\psi(x) \stackrel{\mathrm{def}}{=} \psi(x_f)$, where x_f is some element in $f^{-1}(x) \subseteq D$, i.e. $x = f(x_f)$. It is clear that $(\alpha x)/\!\!/ \alpha = x$ and $x/\!\!/\alpha\downarrow \Longrightarrow \alpha(x/\!\!/\alpha)\downarrow$. Also define so that $\psi(c)/\!\!/ f$ is undefined for any $c \in C$ and $f \in F$. □

Definition 7 Let $(M, +)$ and $(M', +)$ be merge systems. A total function $\varphi: M \to M'$ is a *merge homomorphism* if for all $a, b \in M$

$$\varphi(a + b) \simeq \varphi(a) + \varphi(b).$$

□

Definition 8 Let R and R' be G–record algebras over M and M', respectively. A *homomorphism* from R into R' is a (total) function $h: R \to R'$ satisfying the following:

$$\begin{aligned} h(\epsilon) &= \epsilon. \\ h(u) &\in M' \qquad (u \in M). \\ h(\alpha u) &\simeq \alpha h(u). \\ h(u/\!\!/\alpha) &\simeq h(u)/\!\!/\alpha. \\ h(u + v) &\simeq h(u) + h(v). \end{aligned}$$

□

If the function h is a bijection and the inverse h^{-1} of h is a homomorphism from R' into R then R and R' are called *isomorphic* to each other.

We define a binary relation $\leq$ on R by

$$a \leq b \iff a + b = b.$$

Then it is proved that the binary relation $\leq$ is a partial order relation on R as follows: the reflective law $a \leq a$ follows the idempotent law $a + a = a$. The transitive law that $a \leq b \wedge b \leq c \Longrightarrow a \leq c$ is proved as follows: add c to both sides of the equation $b = a + b$ and then apply the equation $c = b + c$, then we obtain the equation $c = a + c$, which concludes the transitive law. The anti-symmetric law, i.e. $a \leq b \wedge b \leq a \Longrightarrow a = b$, follows directly from the commutative law of the record algebra. So we have the proposition:

Proposition 2 *Let R be a G–record algebra and $\leq$ the binary relation on R defined by $a \leq b \iff a + b = b$. Then $(R, \leq)$ is a partial order structure.*

Let S be a subset of a G–record algebra R. We write $\bigsqcup S$ for the supremum of S with respect to $\leq$. Also we call it the *sum* of S. A non empty subset S of R is *consistent* if there exists the sum for any two elements of S.

Definition 9 A *standard* G–record algebra R over M is a G–record algebra over M such that the following hold.

(1) $\alpha u + \beta v$ is defined for any $u, v \in R$ whenever $\alpha \not\sim \beta$.

(2) For each $u \in R$ there exist a set $H \subseteq G$ and a family $\{x_\alpha \in \{\epsilon\} \cup M \cup X\}_{\alpha \in H}$ such that $u = \bigsqcup\{\alpha x_\alpha \mid \alpha \in H\}$.

□

Remark The above H is not always an *anti–chain*, where an anti–chain is a subset H of G such that every distinct two elements in H are incomparable in G. For example define a (G, M)–PTT t by

$$t \stackrel{\text{def}}{=} \{a^n \epsilon \mid n \geq 0\}$$

where $a \in G$, $a \neq \varepsilon$. The PTT t satisfies the constraint $x \bowtie ax$ with $x = t$ and has an exactly one branch at every node. $\{a^n \mid n \geq 0\}$ is not an anti-chain. However it is clear that if $t = \bigsqcup\{\alpha x_\alpha \mid \alpha \in H\}$ for some $H \subseteq G$ and family $\{x_\alpha\}_{\alpha \in H}$ then $H \subseteq \{a^n \mid n \geq 0\}$ and hence H can not be an anti–chain.

A subset B of R is a G–*basis* if each $x \in R$ is the sum of some subset S_x of $G[B]$, i.e. $x = \bigsqcup S_x$, where $G[B] \stackrel{\text{def}}{=} \{gy \mid y \in B \cup \{\epsilon\}, g \in G\}$. Also we say R *is generated over* B when B is a G–basis of R.

Definition 10 Let M be a merge system. R is a *G–record algebra over (M, φ)* if the following hold.

(1) $\varphi\colon M \to R$ is a merge homomorphism.

(2) $\varphi(M)$ is a G–basis of R.

(3) Every element of $\varphi(M)$ is atomic.

□

It is clear that condition (3) is equivalent both to that if $\alpha \neq \varepsilon$ then $\alpha x \notin \varphi(M)$ for any $x \in R$ and to that $x /\!\!/ \alpha$ is not defined for any $x \in M$ and $\alpha \in G$. The injection φ may be implicit when the context is clear.

Definition 11 R is *complete* if every consistent subset of R has a sum and the sum operation is commutative with both the left and right action of G, i.e.:

$$\begin{array}{rcl} \bigsqcup S\downarrow & \iff & S \text{ is consistent.} \\ \alpha x\downarrow & \iff & x\downarrow. \\ \alpha(\bigsqcup S) & \simeq & \bigsqcup\{\alpha s \mid s \in S\}. \\ (\bigsqcup S)/\!\!/\alpha & \simeq & \bigsqcup\{s/\!\!/\alpha \mid s \in S, s/\!\!/\alpha\downarrow\}. \end{array}$$

□

We define $a \sqcup b \stackrel{\text{def}}{=} \bigsqcup\{a, b\}$. In the rest of the paper we assume every G–record algebra is complete.

Proposition 3 *If R is a G-record algebra then the following hold in R, where $a, b \in R$.*

(1) $a + b \simeq a \sqcup b$.

(2) $a \sqcup b\downarrow \Longrightarrow a\downarrow \wedge b\downarrow$.

Proof (1) Suppose first $a + b\downarrow$. We show $a \sqcup b\downarrow$ and $a + b = a \sqcup b$. As $a + b\downarrow$ it follows that $a \leq a + b$ and $b \leq a + b$. Suppose $x \in R$ and $a \leq x$ and $b \leq x$. Then by definition of $\leq$ it follows that $x = a + x$ and $x = b + x$, whence

$$x = x + x = (a + x) + (b + x) = a + b + x$$

i.e. $a + b \leq x$. Therefore by definition of $a \sqcup b$ it follows $a + b = a \sqcup b$.

In the second case suppose $a \sqcup b\downarrow$. By definition of $\sqcup$ it follows that $a \leq a \sqcup b$ and $b \leq a \sqcup b$. So by definition of $\leq$ it follows that $a + a' = a \sqcup b$ and $b + b' = a \sqcup b$ for some $a', b' \in R$. As $a \sqcup b\downarrow$ and $a \sqcup b = a \sqcup b + a \sqcup b$ it follows that $(a + a') + (b + b')\downarrow$. By the commutativity law it follows that $(a + b) + (a' + b')\downarrow$. Hence by the partiality axiom for $+$ it follows that $a + b\downarrow$. Therefore from the first case above we get $a + b = a \sqcup b$.

(2) Suppose $a \bigsqcup b$. Then it follows from (1) that $a + b\downarrow$. By the partiality axiom of $+$ we get $a\downarrow$ and $b\downarrow$. □

Proposition 4 $\alpha(a \sqcup b) \simeq \alpha a \sqcup \alpha b$.

Proof Applying the left action α on both sides of $a \sqcup b \simeq a + b$, $\alpha(a \sqcup b) \simeq \alpha a \sqcup \alpha b$ follows the distributive law between α and $+$. □

Proposition 5 *Let R be a complete G-record algebra. Then if $S, S' \subseteq R$ then the following holds:* $\bigsqcup S + \bigsqcup S' \simeq \bigsqcup S \cup S'$.

Proof Clearly $\bigsqcup S + \bigsqcup S'\downarrow \iff \bigsqcup S \cup S'\downarrow$ from the partiality hypothesis on $\bigsqcup$ above. So we can assume that both of them have values. As $S \subseteq S \cup S'$ we get $\bigsqcup S \leq \bigsqcup S \cup S'$. Similarly we get $\bigsqcup S' \leq \bigsqcup S \cup S'$. Hence we get $\bigsqcup S + \bigsqcup S' \leq \bigsqcup S \cup S'$. For the converse, as $u \leq \bigsqcup S + \bigsqcup S'$ for any element u in $S \cup S'$ we get $\bigsqcup S \cup S' \leq \bigsqcup S + \bigsqcup S'$. □

Proposition 6 *Let S be a consistent subset of a (complete) G-record algebra R. If $S = \bigcup\{S_\lambda \mid \lambda \in \Lambda\}$ then $\bigsqcup S = \bigsqcup\{\bigsqcup S_\lambda \mid \lambda \in \Lambda\}$.*

Proof As $S_\lambda \subseteq S$ we get $\bigsqcup S_\lambda \leq \bigsqcup S$. Hence $\bigsqcup\{\bigsqcup S_\lambda \mid \lambda \in \Lambda\} \leq \bigsqcup S$. For the converse, as every $x \in S$ is an element of S_{λ_0} for some λ_0, we get $x \leq \bigsqcup S_{\lambda_0}$. Hence we get $x \leq \bigsqcup\{\bigsqcup S_\lambda \mid \lambda \in \Lambda\}$. Therefore $\bigsqcup S \leq \bigsqcup\{\bigsqcup S_\lambda \mid \lambda \in \Lambda\}$. □

2.2 Partially Tagged Trees

In this section we introduce a domain of *partially tagged trees* (PTT) as a canonical record algebra. A PTT is a kind of unordered possibly nonwellfounded trees which is tagged only at some of leaf nodes. The set of PTTs will be characterized to be a complete free record algebra with some additional conditions.

Let M be a set of *tags*. Substructure of tags are left unanalysed. Let G be a feature monoid with the unit ε.

Definition 12 A *tree T over G* is a non empty subset of G which is closed under prefixes, i.e. if $\alpha\beta \in T$ then $\alpha \in T$. □

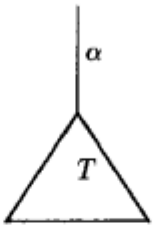

Figure 1: The left action by α on T: αT

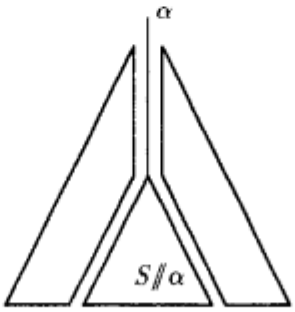

Figure 2: $S/\!\!/\alpha$.

The singleton $\{\varepsilon\}$ is the *unit tree*. By definition the unit tree is the minimum tree over G. As we have assumed that every feature monoid is a tree the set $\{y \in T \mid y \leq x\}$ is totally ordered with $\leq$ for any $x \in T$. Clearly if T_1 and T_2 are trees then $T_1 \cup T_2$ is a tree. Also $T_1 \cap T_2$ is a tree. Moreover the class of trees over G is closed under both of the set theoretical union and intersection of an arbitrary family of trees.

Let α and T be a feature and tree, respectively. By αT we mean the smallest tree which has all features $\alpha\alpha'$ for any $\alpha' \in T$. (See figure 1.) For instance let $G = L^*$ and $T = \{\varepsilon, \langle b\rangle, \langle c\rangle, \langle c, d\rangle\}$ with $L = \{a, b, c, d\}$. Then $aT = \{\varepsilon, \langle a\rangle, \langle a, b\rangle, \langle a, c\rangle, \langle a, c, d\rangle\}$.

Let S be a set of features. By $S/\!\!/\alpha$ we mean the maximum tree S' such that $\alpha S' \subseteq S$. By definition of an tree $S/\!\!/\alpha$ is a tree whenever it exists. For a tree S $S/\!\!/\alpha$ is a *subtree* of S. (See figure 2).

β is called a *direct successor* of α in T if $\alpha \leq \beta$, $\alpha \neq \beta$ and there is no $\gamma \in T$ such that $\alpha < \gamma < \beta$. A feature as a node of a tree may have infinite number of direct successors. A feature α of a tree T is a *leaf* if α has no successor in T. We write $leaf(T)$ for the set of leaves. A *tag function* is a *partial* function from $leaf(T)$ into M. In particular the empty function $\emptyset$ is a tag function.

Definition 13 A *partially tagged tree* (PTT) is an ordered pair (T, f) of a tree T and a tag function of T. (See Figure 3.) □

The *unit* PTT, denoted by ϵ, is the ordered pair of the unit tree and the empty function $\emptyset$:

$$\epsilon = (\{\varepsilon\}, \emptyset).$$

Given a PTT $t = (T, f)$ $v \in M$ is the *tag at* α in t if $f(\alpha) = v$. We call PTTs a (G, M)-PTT when the feature monoid G and the set M of tags should be explicit. Now for a feature α we define left actions αt $(= \alpha \cdot t)$ and right actions $t/\!\!/\alpha$ also on PTTs. Let

White circles are nodes without tags, while black ones with tags.

Figure 3: A partially tagged tree.

$t = (T, f)$ be a PTT. For a feature α we define $\alpha t \stackrel{\text{def}}{=} (\alpha T, f^\alpha)$, where f^α is a tag function of αT such that $\mathrm{dom}(f^\alpha) = \{\alpha\gamma \mid \gamma \in \mathrm{dom}(f)\}$ and $f^\alpha(\alpha\gamma) = f(\gamma)$ for $\gamma \in \mathrm{dom}(f)$. For features α in T the right (partial) action is defined by the following equations:

$$t/\!/\alpha \stackrel{\text{def}}{=} (T/\!/\alpha, f_\alpha)$$

where f_α is a tag function of $T/\!/\alpha$ defined by $f_\alpha(\beta) = f(\alpha\beta)$.

Let $t_1 = (T_1, f_1)$ and $t_2 = (T_2, f_2)$ be two PTTs. $t = (T_1 \cup T_2, f_1 \cup f_2)$ is the *merge* of t_1 and t_2, written $t_1 + t_2$, iff $f_1 \cup f_2$ is a tag function of the tree $T_1 \cup T_2$. By this definition $t_1 + t_2$ is undefined if there is some $\alpha \in T_1 \cap T_2$ such that $f_1(\alpha)$ and $f_2(\alpha)$ are defined but are not the same. Also $t_1 + t_2$ is undefined if there is some β such that $f_1(\beta)$ is defined but β is not a leaf of T_2.

A set of PTTs is *consistent* if any two PTTs in the set have the merge. Let $t_1 = (T_1, f_1)$ and $t_2 = (T_2, f_2)$ be PTTs. By $t_1 \leq t_2$ we means that $T_1 \subseteq T_2$ and $f_1 \subseteq f_2$. Let R be the set of (G, M)–PTTs. As the set union is complete it is easily checked that the partial order structure $(R, \leq)$ is complete. Also it is proved without difficulty that the left and right actions are commutative with the operation of taking the supremum. M can be embedded into R by identifying each element a of M with the singleton PTT whose unique tag is a. We denote this embedding by $\varphi: M \to R$. We show that the set of (G, M)–PTTs is characterized as the most universal complete G–record algebra over M.

Proposition 7 *Let M and G be a merge system and a feature monoid respectively. Then there exists a unique (R, φ) such that the following hold.*

(1) R is a complete and standard G–record algebra over (M, φ).

(2) For any record algebra R' and a homomorphism $\varphi' : M \to R'$, there exists a complete G–homomorphism f from R into R' such that $f \circ \varphi = \varphi'$, where $\circ$ is the function composition operator.

Proof Let M be a merge system. Let R be the set of (G, M)–PTTs and φ an injection from M into R such that $\varphi(x)$ is the *singleton* PTT whose tag is x, i.e. $\varphi(x) = \{(\{\varepsilon\}, \{(\varepsilon, x)\})\}$. It is clear that R is a complete and standard G–record algebra over (M, φ). As R is standard the function f' from $G[\varphi(M)]$ into $G[\varphi'(M)]$ which assigns $\alpha\varphi'(x)$ to $\alpha\varphi(x)$ maps consistent subsets of R to those of R'. Hence f' determines a complete G–record homomorphism f from R into R'. It is easy to see that $f \circ \varphi = \varphi'$.

The uniqueness is a routine. □

(R, φ) in the above proposition is called a *free* complete G-record record algebra over M.

Corollary 8 *Let R be the set of (G, M)–PTTs and let φ be the embedding injection $\varphi: M \to R$. Then the 6-tuple $(R, G, +, \cdot, /\!/, \epsilon)$ is a complete and standard G–record algebra over M with the injection $\varphi: M \to R$.*

Example 4 We show a non trivial record algebra on which features operate totally from both sides. Let G be a feature monoid and A a non empty set which has sufficient number of elements. A^G is the set of functions from G into A. For each $f \in A^G$ $f/\!/\alpha$ is the function f' in A^G defined by $f'(\beta) \stackrel{\text{def}}{=} f(\alpha\beta)$ for each $\beta \in G$. We define $t/\!/\alpha \stackrel{\text{def}}{=} \{f/\!/\alpha \mid f \in t\}$ for each $t \subseteq A^G$. Also for each $t \subseteq A^G$ we define $\alpha \cdot t$ $(= \alpha t)$ to be the largest set $t' \subseteq A^G$ such that $t'/\!/\alpha \subseteq t$.

Let $\mathcal{R} = (pow(A^G), G, \cap, \cdot, /\!/, A^G)$. Then it is clear that $\mathcal{R}$ is a complete G–record algebra. Note that $t/\!/\alpha$, αt, and $t \cup t'$ are defined for all $t, t' \subseteq A^G$, $\alpha \in G$, i.e. $\mathcal{R}$ is a totally defined algebra. □

Example 5 Let $G, A, \mathcal{R}$ be the same as in example 4. For some indexing set Π we construct a family $\{\mathcal{R}_M\}_{M \in \Pi}$ of complete G–record *sub*algebras $\mathcal{R}_M$ of $\mathcal{R}$ *over* M. In fact Π is the set of sets $M \subseteq pow(A^G)$ such that

(1) $\emptyset \notin M$.

(2) If $x \in \bigcup M$, $\alpha \in G$, and $\alpha \neq \varepsilon$ then $x/\!/\alpha \notin \bigcup M$.

(3) If $t, t' \in M$ and $t \neq t'$ then $t \cap t' = \emptyset$.

We show that $\Pi \neq \emptyset$. Let S be the set of $f \in A^G$ such that the image of f is $\{a, b\} \subseteq A$ for some distinct $a, b \in A$ and $f^{-1}(a) = \{\varepsilon\}$. Let $M_S = \{\{f\} \mid f \in S\}$. As A has at least two elements it follows that $S \neq \emptyset$ and hence $M_S \in \Pi$. Therefore $\Pi \neq \emptyset$. For any $M \in \Pi$ let $\mathcal{R}_M = (R_M, G, \cap, \cdot, /\!/, \epsilon_M)$ be the least complete G–record subalgebra of $\mathcal{R}$ such that $M \subseteq R_M$. It is clear that $\mathcal{R}_M$ exists and is a totally defined record algebra. As ϵ_M is the unit of $\mathcal{R}_M$ it follows that $\epsilon_M \cap t = t$ for all $t \in R_M$ and hence $\epsilon_M = \bigcup R_M$. It is clear from the construction of $\mathcal{R}_M$ that in $\mathcal{R}_M$ the following hold.

(1) If $t, t' \in M$ and $t \neq t'$ then $t \cap t' = \emptyset$.

(2) If $\alpha t = \emptyset$ then $t = \emptyset$.

(3) If $t/\!/\alpha = \emptyset$ then $t = \emptyset$.

(4) If $t \in M$, $u \in R_M$, and $\alpha \in G \setminus \{\varepsilon\}$ then $t \cap \alpha u = \emptyset$.

(5) If $\{\alpha_j\}_{j \in J}$ is an anti–chain in G, and $\{t_j \mid j \in J\}$ is a family of elements of R_M then $\bigcap\{\alpha_j t_j \mid j \in J\} \neq \emptyset$.

For $M \in \Pi$ it follows from these properties that by removing $\emptyset$ from R_M we get a desired partial G–record algebra $\mathcal{R}'_M$ over M, where $\mathcal{R}'_M = (R_M \setminus \{\emptyset\}, G, \cap, \cdot, /\!/, \epsilon_M)$.

□

2.3 The Unification Theory

In this subsection we fix M, G, R, X as follows unless mentioned otherwise: R is a complete and standard G–record algebra over a merge system $(M, +)$ and X is a set of parameters. Elements of $\mathcal{E}(G, M \cup X, \{\cdot, +, /\!/\})$ are called a *record term*. It is also called a *parametric record*[3]. Elements of $\mathcal{E}(G, M \cup X, \{\cdot\})$ are called a *basic record term*. It is convenient to have the left action built into the record terms. So we identify elements of $\mathcal{E}(G, M, F)$ up to the following equations:

$$\begin{aligned} (\alpha\beta)x &= \alpha(\beta x). \\ \varepsilon x &= x. \end{aligned}$$

[3]Record terms are called a *partially specified term* (PST) in [16].

In fact we abuse the symbol $\mathcal{E}(G, M, F)$ for the quotient set of $\mathcal{E}(G, M, F)$ divided by the least congruence relation generated by the above three equations. This convention will be used without mentioning.

Note that we have no syntactical counter part for ϵ. Hence ϵ does not appear in any record term. In particular ϵ is not a record term. An *atomic constraint* is an ordered pair of record terms, written

$$p \bowtie q$$

where p, q are record terms.

Remark As $u /\!\!/ \alpha \bowtie v$ is semantically reduced to $u \bowtie \alpha w$ plus $w \bowtie v$ with w being a new parameter we assume $p, q \in \mathcal{E}(G, M \cup X, \{\cdot, +\})$ without loss of generality.

A *constraint* is a possibly infinite set of atomic constraints. We give a set of *constraint rules* in table 1. Each rule there means a condition on constraints.

Definition 14 A constraint C is *closed* if C satisfies all clauses in the constraint rule table 1. □

Definition 15 Recalling that R is a complete and standard G–record algebra over M a pair of u and v in $\mathcal{E}(G, M \cup X, \{\cdot, +, /\!\!/\})$ is a *conflict* if one of the following hold.

(1) $u \in M$ and $v \in M$ but $u + v$ is undefined in $(M, +)$.

(2) $u \in M$ and $v = \alpha w$ for some record term w and $\alpha \in G \setminus \{\varepsilon\}$.

□

Clearly if u and v are a conflict pair then there is no assignment such that $f(u + v)$ is defined. The set of constraint rules is designed so that every constraint C is *solvable* iff it is unifiable, i.e. there is a closed and consistent extension C' of C.

In table 1 C is a constraint, $x \in X$, and $\alpha, \beta \in G$ are incomparable features appearing in C, and u, v, w are record terms appearing in C. Note that x appearing in the restricted transitive rule is a *parameter* in X.

Table 1: Constraint Rules

Merge (0)	$u \bowtie v \in C \wedge u, v \in M$	$\Longrightarrow$	$u + v\downarrow$.
Base	$\alpha u \bowtie \beta v \in C, \alpha \not\sim \beta$	$\Longrightarrow$	$u \bowtie u \in C, v \bowtie v \in C$.
Reflexive	$u \in X \cup M$	$\Longrightarrow$	$u \bowtie u \in C$.
Symmetric	$u \bowtie v \in C$	$\Longrightarrow$	$v \bowtie u \in C$.
Restricted Transitive	$x \bowtie u \in C, x \bowtie v \in C$	$\Longrightarrow$	$u \bowtie v \in C$.
Merge (1)	$(u + v) \bowtie w \in C$	$\Longrightarrow$	$u \bowtie w \in C$.
Merge (2)	$(u + v) \bowtie w \in C$	$\Longrightarrow$	$u \bowtie v \in C$.
Cancellation	$\alpha u \bowtie \alpha v \in C$	$\Longrightarrow$	$u \bowtie v \in C$.

Definition 16 A *closed constraint* over R is a set of atomic constraints satisfying the rules on table 1. □

From the reflexive, symmetric, and restricted transitive rules on the table every closed constraint contains an equivalence relation between parameters. However as there is not a full transitive law the closed constraint gives no equivalence relation on $\mathcal{E}(G, M \cup X, \{\cdot, +, /\!\!/\})$ in general. The minimum closed extension is called the *closure* of S. Clearly

the closure of a finite constraint S is computed effectively. A *unification problem* is to find the closure. The input of the unification algorithm is a finite set C of atomic constraints:

$$C = \{p_1 \bowtie q_1, \ldots, q_n \bowtie q_n\}.$$

The output is the consistent closure $\overline{C}$ or 'conflict' when $\overline{C}$ has a conflict.

In the following example $G = \{a, b\}^*$ $(a \neq b)$ and $M = \{1, 2\}$ is a trivial merge system. That is, 1+2 is undefined and $1 + 1 = 1$, $2 + 2 = 2$. Let R be a G–record algebra over M. Let $X = \{x, y\}$ be a set of parameters. For saving the space we omit obvious atomic constraints obtained by, for instance, the reflexive rule or the base rule. Also we make use of the symmetric rule implicitly.

Example 6 Let $C_1 \stackrel{\text{def}}{=} \{ax + bx \bowtie by + a1\}$. By merge rule (1) we get $ax \bowtie a1$ and $bx \bowtie by$. Applying the cancellation rule to the two we get $x \bowtie 1$ and $x \bowtie y$. Applying the restricted transitive rule we get $y \bowtie 1$. Thus the closure of C_1 is

$$\{ax + bx \bowtie by + a1, ax \bowtie a1, bx \bowtie by, x \bowtie 1, x \bowtie y, y \bowtie 1\}.$$

There is no conflict in the closure. □

Example 7 The example gives a cyclic graph.

$$C_2 \stackrel{\text{def}}{=} \{x \bowtie ay + by, y \bowtie ax + bx, x \bowtie y\}.$$

Applying the restricted transitive law to parameter x we get $y \bowtie ay + by$. With this and applying the restricted transitive law to $y \bowtie ax + bx$ w.r.t. y we get $ax + bx \bowtie ay + by$. By repeating merge law (1) we get $ax \bowtie ay$ and $bx \bowtie by$. Applying the cancellation law to each of them we get $x \bowtie y$. Now no rule is applicable. The closure of C_2 is:

$$\{x \bowtie ay + by, y \bowtie ax + bx, x \bowtie y, ax \bowtie ay, bx \bowtie by, x \bowtie ax + bx, y \bowtie ay + by\}.$$

The output of this unification means a singleton graph which has two self–loops with features a and b. □

2.4 Satisfiability of Record Constraints

M, X, G, R are the same in the previous subsection. Recall the definition of $\mathcal{E}$ (definition 2).

Definition 17 We define a function $\pi: \mathcal{E}(G, X \cup M, \{\cdot, +\}) \rightarrow pow(\mathcal{E}(G, X \cup M, \{\cdot\}))$ inductively by the following equations:

$$\begin{array}{rcll} \pi(x) & = & \{x\} & \text{if } x \in M \cup X. \\ \pi(x + y) & = & \pi(x) \cup \pi(y). & \\ \pi(\alpha x) & = & \{\alpha y \mid y \in \pi(x)\}. & \end{array}$$

□

Example 8 If $a, b, c \in G$ and $x, y, z \in X$ then $\pi(a(bx + c(y + z))) = \{abx, acy, acz\}$. □

We abuse the π as $\pi(S) \stackrel{\text{def}}{=} \bigcup\{\pi(x) \mid x \in S\}$. An *assignment* is a partial function from X into R.

Proposition 9 *If f is an assignment then there is the largest partial function $h: \mathcal{E}(G, M \cup X, \{\cdot, +, /\!\!/\}) \to R$ which satisfies the following.*

$$\begin{array}{rcrcl} x \in X, h(x)\downarrow & \Longrightarrow & x & \in & dom(f). \\ x \in dom(f) & \Longrightarrow & h(x) & = & f(x). \\ x \in M & \Longrightarrow & h(x) & = & x. \\ & \Longrightarrow & h(x+y) & \simeq & h(x) + h(y). \\ & \Longrightarrow & h(\alpha x) & \simeq & \alpha h(x). \\ & \Longrightarrow & h(x/\!\!/\alpha) & \simeq & h(x)/\!\!/\alpha. \end{array}$$

Proof A proof is done by structural induction on record terms. □

We use the assignment f also for the extension h.

Definition 18 Let p be a record term in $\mathcal{E}(G, M \cup X, \{\cdot, +, /\!\!/\})$, f an assignment and t a record in R. Then t *is an instance of* p *with* f, written

$$t :_f p$$

if for any $\alpha x \in \pi(p)$ the following hold, where $x \in X \cup M$, $\alpha \in G$:

(1) $t/\!\!/\alpha\downarrow$.

(2) $x \leq t/\!\!/\alpha$ if $x \in M$.

(3) $f(x) = t/\!\!/\alpha$ if $x \in X$.

□

Clearly $t :_f p$ implies $f(p)\downarrow$.

Example 9 $ac+ad+bc+bd :_f ax+bx$ is a valid assertion, where $a, b \in G, c, d \in M, c+d\downarrow$, and $f(x) = c + d$. □

Definition 19 f is a *solution* of $p \bowtie q$ (in R) iff $t :_f p$ and $t :_f q$ for some record $t \in R$. Given a constraint C f is a *solution* of C (in R) if f is a solution of each atomic constraint in C. □

f *solves* a constraint if f is solution of the constraint. Recall that R is standard.

Lemma 1 *Let $x \in X$, $\alpha, \beta \in G$, $u, v, w \in \mathcal{E}(G, M \cup X, \{\cdot, +\})$, and f an assignment. Then the following hold.*

(1) If $u, v \in M$ and f solves $u \bowtie v$ then $u + v\downarrow$.

(2) If $u \in X$ and f solves $x \bowtie \alpha u$ then $f(u) = f(x)/\!\!/\alpha$.

(3) If $u \in M$ and f solves $x \bowtie \alpha u$ then $u \leq f(x)/\!\!/\alpha$.

(4) If f solves $u \bowtie v$ then f solves $v \bowtie u$.

(5) If f solves $x \bowtie u$ and $x \bowtie v$ then f solves $u \bowtie v$.

(6) If $\alpha \not\sim \beta$ and f solves $\alpha u \bowtie \beta v$ then f solves both $u \bowtie u$ and $v \bowtie v$.

(7) If f solves $(u + v) \bowtie w$ then f solves $u \bowtie w$.

(8) If f solves $(u + v) \bowtie w$ then f solves $u \bowtie v$.

(9) If f solves $\alpha u \bowtie \alpha v$ then f solves $u \bowtie v$.

(10) If $u, v \notin M \cup X$ and f solves $u \bowtie v$ then there is an extension f' of f such that f' solves both $x \bowtie u$ and $x \bowtie v$ for some $x \in dom(f') \setminus dom(f)$.

Proof

(1) As f solves $u \bowtie v$ there is some $t \in R$ such that $u :_f t$ and $v :_f t$. Hence as $u, v \in M$ we get $u \leq t$ and $v \leq t$. Hence $u + v\downarrow$.

(2) Suppose that $u \in X$ and f solves $x \bowtie \alpha u$. Then there is some $t \in R$ such that $f(x) = t/\!/\varepsilon = t$ and $u = t/\!/\alpha$. Hence $f(u) = f(x)/\!/\alpha$.

(3) Suppose that $u \in M$ and f solves $x \bowtie \alpha u$. Then there is some $t \in R$ such that $f(x) = t/\!/\varepsilon = t$ and $u \leq t/\!/\alpha$. Hence $u \leq f(x)/\!/\alpha$.

(4) It is obvious by definition that f solves $u \bowtie v$ iff f solves $v \bowtie u$. The case follows from this equivalence.

(5) Suppose that f solves $x \bowtie u$ and $x \bowtie v$. Then $t :_f x$, $t :_f u$, $t' :_f x$, $t' :_f v$ for some $t, t' \in R$. Hence we get $t = f(x) = t'$, whence $t :_f u$ and $t :_f v$. Hence by definition f solves $u \bowtie v$.

(6) Suppose $\alpha \not\sim \beta$ and f solves $\alpha u \bowtie \beta v$. Then $t :_f \alpha u$ and $t :_f \beta v$ for some $t \in R$. So $t/\!/\alpha :_f u$ and $t/\!/\alpha :_f v$. Hence f solves $u \bowtie u$ and $v \bowtie v$.

(7) Suppose f solves $(u + v) \bowtie w$. Then $t :_f u + v$ and $t :_f w$ for some $t \in R$. As $t :_f u + v$ implies $t :_f u$ f solves $u \bowtie w$.

(8) Suppose f solves $(u + v) \bowtie w$. Then $t :_f u + v$ for some t. As $t :_f u + v$ implies $t :_f u$ and $t :_f v$ f solves $u \bowtie v$.

(9) Suppose f solves $\alpha u \bowtie \alpha v$. Then $t :_f \alpha u$ and $t :_f \alpha v$ for some $t \in R$, whence $t/\!/\alpha :_f u$ and $t/\!/\alpha :_f v$. Hence f solves $u \bowtie v$.

(10) Suppose $u, v \notin M \cup X$ and f solves $u \bowtie v$. Let x be a new parameter not in $\mathrm{dom}(f) \cup \mathcal{V}(u) \cup \mathcal{V}(v)$. Let f' be the extension of f such that $\mathrm{dom}(f') = \{x\} \cup \mathrm{dom}(f)$ and $f'(x) = f(u) + f(v)$. It is clear that f' is well-defined, f' solves both $x \bowtie u$ and $x \bowtie v$. Hence f' satisfies condition (10) in the proposition.

□

Definition 20 $\models$ is the *largest* ternary relation which satisfies the following clauses, where $x \in X$, $u, v, w \in \mathcal{E}(G, M \cup X, \{\cdot, +\})$, $\alpha, \beta \in G$:

$$
\begin{array}{lrcl}
(1) & u, v \in M \wedge R, f \models u \bowtie v & \Longrightarrow & u + v\downarrow. \\
(2) & u \in X \wedge R, f \models x \bowtie \alpha u & \Longrightarrow & f(u) = f(x)/\!/\alpha. \\
(3) & u \in M \wedge R, f \models x \bowtie \alpha u & \Longrightarrow & u \leq f(x)/\!/\alpha. \\
(4) & R, f \models u \bowtie v & \Longrightarrow & R, f \models v \bowtie u. \\
(5) & R, f \models x \bowtie u \wedge R, f \models x \bowtie v & \Longrightarrow & R, f \models u \bowtie v. \\
(6) & \alpha \not\sim \beta \wedge R, f \models \alpha u \bowtie \beta v & \Longrightarrow & R, f \models u \bowtie u \wedge R, f \models v \bowtie v. \\
(7) & R, f \models (u + v) \bowtie w & \Longrightarrow & R, f \models u \bowtie w. \\
(8) & R, f \models (u + v) \bowtie w & \Longrightarrow & R, f \models u \bowtie v. \\
(9) & R, f \models \alpha u \bowtie \alpha v & \Longrightarrow & R, f \models u \bowtie v. \\
(10) & R, f \models u \bowtie v & \Longrightarrow & R, f \models x \bowtie u \wedge\ R, f \models x \bowtie v \quad (\exists x \in \mathrm{dom}(f)).
\end{array}
$$

□

Proposition 10 *The relation* $\models$ *exists.*

Proof Clearly $\emptyset$ satisfies all clauses in definition 20. Take the union of all such relations. Then it is clear that the union also satisfies the clauses. □

Remark As the class of complete and standard G–record algebras R can be a proper

class even if G is fixed the relation $\models$ is a proper class relation in general.

Proposition 11 *Let $p, q \in \mathcal{E}(G, M \cup X, \{\cdot, +\})$ and f an assignment such that $\mathcal{V}(p) \cup \mathcal{V}(q) \subseteq dom(f)$. Then the following are equivalent.*

(1) f is a solution of $p \bowtie q$ in R.

(2) $R, f' \models p \bowtie q$ for some extension f' of f.

Proof
(1)$\Longrightarrow$(2): Suppose f is a solution of $p \bowtie q$ in R. Then there is the consistent closure C of $p \bowtie q$. Let $C' = \{u \bowtie v \in C \mid u, v \notin X \cup M\}$ and x_c be a new and unique parameter for each $c \in C'$. Let f' be the extension of f such that $f(x_{u \bowtie v}) = f(u) + f(v)$ for $u \bowtie v \in C'$. Let C'' be the consistent closure of $C \cup \bigcup\{\{x_c \bowtie u, x_c \bowtie v\} \mid c = (u \bowtie v), c \in C'\}$ and finally let $D = \{(R, f', c) \mid c \in C''\}$. From lemma 1 D as a ternary relation satisfies all defining clauses for $\models$ in definition 20. As $\models$ is the largest such relation we get $D \subseteq \models$.

(2)$\Longrightarrow$(1): Suppose $R, g \models p \bowtie q$. If $p, q \in M \cup X$ then it is clear by definition of a solution and $\models$ that f solves $p \bowtie q$. Otherwise by condition (10) in definition 20 there is a parameter $x \in \mathrm{dom}(g)$ such that $R, g \models x \bowtie p$ and $R, g \models x \bowtie q$. Then it follows from conditions in definition 20 that for any $u \in \pi(p) \cup \pi(q)$ $R, g \models x \bowtie u$. As x is a parameter and u is a basic record term it follows from the definition of $\models$ that $g(x) :_g u$. Hence g solves $p \bowtie q$ in R. □

Example 10 Let $x \in X$ be a parameter. We show that the constraint

$$ad1 + bd2 \bowtie ax + bx$$

has no solution, where 1 and 2 are used as distinct atoms. To see this suppose that a solution f exists. Then by definition of a solution there exists $t \in R$ such that $t :_f \{ax + bx\}$ and $t :_f \{ad1 + ad2\}$. Then by definition of an instance we have $t/\!/a = f(x)$ and $t/\!/b = f(y)$. Also $ad1 \leq t$ follows from $t :_f \{ad1 + ad2\}$. Hence $d1 \leq t/\!/a = f(x)$. So we get $1 \leq f(x)/\!/d$. Similarly we get $2 \leq f(x)/\!/d$. This is impossible because the sum $1 + 2$ of distinct tags 1 and 2 was assumed to be undefined.

However note that the constraint $ad1 + bd2 \bowtie ac3 + bc3$, which is obtained by applying substitution $x \mapsto c3$, is true in the record algebra R. So this example explains why a restricted notion of a solution is necessary for an equivalence between satisfiability and unifiability in our constraint language. □

Definition 21 A record $r \in R$ is an *initial segment* of a basic record term u if one of the following hold.

- $r = \alpha c$ and $u = \alpha v$, where $v \in \mathcal{E}(G, M \cup X, \{\cdot\})$, $\alpha \in G$.
- $r = u = \alpha c$ where $\alpha \in G$, $c \in M$.

□

Theorem 12 (Record Solution Theorem) *Every consistent closed constraint has a solution.*

Proof Given a consistent closed constraint S let S' be a minimal constraint such that the following hold.

- $S \subseteq S'$.
- S' is closed under the constraint rules on table 1.

- If $x \bowtie \alpha y \in S'$ and $y \bowtie \beta z \in S'$ then $x \bowtie \alpha\beta z \in S'$, where $\alpha, \beta \in G$, $x, y, z \in M \cup X$.

Choose a unique new parameter, say $x_{p \bowtie q}$, for each $p \bowtie q \in S'$ such that $p, q \notin M \cup X$. Let C be the reflexive closure of $S' \cup \bigcup\{\{x_{p\bowtie q} \bowtie p, x_{p\bowtie q} \bowtie q\} \mid p \bowtie q \in S', p, q \notin M \cup X\}$. It is clear that C has the same set of solutions as S. C may be infinite even when S is finite. Anyway it suffices to show that C is satisfiable. Let $B = \{u \bowtie v \mid p \bowtie q \in C, u, v \in \pi(u) \cup \pi(v)\}$. Clearly $B \subseteq C$. We first construct a solution of B. For $x \in M \cup X$ let D_x be the set of initial segments of some basic record term u such that $x \bowtie u \in C$. As C is consistent and closed D_x has no conflict. We make a convention that $D_x = \{\epsilon\}$ if D_x is empty.

We consider atomic constraints in C of the form $\alpha x \bowtie y$, where $x, y \in M \cup X$. If $x, y \in M$ then α must be ε and $x + y\downarrow$ because C is consistent. If $x \in X$ and $y \in M$ then also α must be ε. However this is a special case of the following by changing the role of x and y.

So finally we suppose $y \bowtie \alpha x$ with $y \in X$, $x \in M \cup X$. Suppose first $x \in X$. Now we show that $D_x = D_y /\!\!/ \alpha$, where $D /\!\!/ \alpha$ denotes the set $\{w /\!\!/ \alpha \mid w \in D, w /\!\!/ \alpha\downarrow\}$. Suppose $u \in D_x$. By definition of D_x there is some u' such that $x \bowtie u' \in C$ and u is an initial segment of u'. As $y \bowtie \alpha x \in C$, $x \bowtie u' \in C$ and C is closed under 'unfolding' $y \bowtie \alpha u'$ must be in C. Hence again by definition of the D_x we get $u \in D_y /\!\!/ \alpha$. For the converse, suppose $u \in D_y /\!\!/ \alpha$. Then $y \bowtie \alpha u' \in C$ for some u' such that αu is an initial segment of $\alpha u'$. So u is an initial segment of u'. As $y \bowtie \alpha x \in C$ after a sequence of several steps of applying constraint rules we have $x \bowtie u' \in C$. Thus $u \in D_x$. Hence we get $D_x = D_y /\!\!/ \alpha$. Let f be the assignment defined by putting $f(z) = \bigsqcup D_z$ for any parameters z of S. As R is a complete G record algebra over M we get $f(x) = f(y) /\!\!/ \alpha$.

In the second case suppose $x \in M$. As $\alpha x \in D_y$ we get $\alpha x \leq \bigsqcup D_y = f(y)$ and hence $x \leq f(y) /\!\!/ \alpha$. Therefore f is a solution of B.

We show that f is a solution of C. Using the fact that f is a solution of B it is a routine to check that the set $\{(R, f, c) \mid c \in C\}$ satisfies all defining clauses of $\models$ in definition 20. As $\models$ is the largest such ternary relation we get $\{(R, f, c) \mid c \in C\} \subseteq \models$. Hence by proposition 11 f is a solution of C in R. □

As an obvious corollary of this theorem we get the equivalence between the unifiability and the satisfiability.

Theorem 13 (Unification Theorem) *Let $p, q \in \mathcal{E}(G, M \cup X, \{\cdot, +\})$. Then the following are equivalent.*

(1) $p \bowtie q$ is unifiable, i.e. has a consistent closure.

(2) $p \bowtie q$ is satisfiable in R.

Let S be a constraint. It is easy to see that the closure of S is the union of the closures of all finite sets of S. The *compactness theorem* follows directly from this:

Theorem 14 (Compactness Theorem) *Let C be a set of constraints. Then C has a solution iff every finite subset of C has a solution.*

3 Unification Grammar over Records

In this section we use R, X, $(M, +)$ for a complete and standard G–record algebra over M, a set of parameters, and a merge system, respectively.

3.1 Semantics of the Program

A *program clause* over $\mathcal{E}(G, M \cup X, \{\cdot, +\})$ is an ordered pair (p, B) of a record term p and a finite set B of record terms. A *program* P is a finite set of program clauses. A *goal* is a non–empty finite set of record terms. A program clause $(p, \{p_1, ..., p_n\})$ $(n \geq 0)$ is written

$$p \leftarrow p_1, ..., p_n.$$

In the case $n = 0$ we write

$$p.$$

for the program clause $(p, \emptyset)$ as usual. A program P is fixed throughout this section.

Example 11 The program P_0 below consists of three Horn clauses for a recursive data type definition for list structures. L is a set of atomic features. Let $\mathcal{R} = (R, G, +, \cdot, /\!/, \epsilon)$ be a record algebra over M, where $G = L^*$ and M is a trivial merge system. In the program we assume that $x, l \in X$, $a, b, nil, atom, list \in M$, and $type, car, cdr, form \in L$.

(1) $type\ atom + form\ a.$

(2) $type\ atom + form\ b.$

(3) $type\ list + form\ (car\ x + cdr\ l) \leftarrow$
$type\ atom + form\ x,\ type\ list + form\ l.$

□

Definition 22 [**Model**] A subset M of R is a *model* of the given program P if for each $t \in M$ there exists some program clause $p \leftarrow p_1, \ldots, p_n$ and assignment f such that the following hold:

(1) $t :_f p.$

(2) For each $1 \leq i \leq n$ there exists $t_i \in M$ such that $t_i :_f p_i$.

□

Clearly $\emptyset$ is a model of any program. Also M and M' are models then $M \cup M'$ is a model. Hence there exists the maximum model of P.

Definition 23 The *semantics* $\mathcal{M}_P$ *of the program* P is the maximum model of the program P. □

Definition 24 We define a transformation $\Phi_P : pow(R) \rightarrow pow(R)$. Given a $Q \subseteq R$ $\Phi_P(Q)$ is the set of records $t \in R$ such that $t :_f p$ for some program clause $(p, B) \in P$ and assignment f such that for any $q \in B$ there is $t' \in Q$ such that $t' :_f q$. □

Example 12 $\Phi_{P_0}(S) = S_1 \cup S_2$ where $S_1 = \{type\ atom + form\ a, type\ atom + form\ b\}$. and S_2 is the set of records $t \in R$ such that the following hold for some $s \in S$ and an assignment f.

(1) $t :_f type\ list + form\ car\ x + form\ cdr\ l.$

(2) $s :_f type\ atom + form\ x.$

(3) $s :_f type\ list + form\ l.$

$\mathcal{M}_{P_0}$ is the largest fixpoint of Φ_{P_0}. □

As $(pow(R), \subseteq)$ is a complete partial ordered structure and Φ_P is a monotone function w.r.t. the order it follows from the standard theorem that there is a maximum fixpoint of Φ_P. Also it is a routine to show that the maximum fixpoint is the maximum semantics $\mathcal{M}_P$. For more details the reader is referred to Aczel[1], in which the existence theorem of the minimum and maximum fixpoints of continuous class functors is given in more general setting.

Definition 25 An assignment f is an *answer solution* of a goal G in $\mathcal{M}_P$ if for each $q \in G$ there exists $t_q \in \mathcal{M}_P$ such that $t_q :_f q$. □

We make a convention. Let $(D, \leq)$ be a partial order structure and let f and f' be partial functions from some set into D. f' is an *extension* of f if $\mathrm{dom}(f) \subseteq \mathrm{dom}(f')$ and $f(x) \leq f'(x)$ for any $x \in \mathrm{dom}(f)$.

Definition 26 A *support* is a consistent and closed constraint. □

Definition 27 A *computation state* (state for short) is a pair (Q, E) of a goal Q and support E. □

Definition 28 A *resolution step* is an ordered pair (s, s') written as

$$s \rightarrow s'$$

of two states $s = (Q, E)$ and $s' = (Q', E')$, where $Q = \{p_1, ..., p_n\}$, satisfying the following:

(1) There exist copies $q_i \leftarrow q_1^i, ..., q_{k_i}^i$ $(1 \leq i \leq n)$ of program clauses such that

$$Q' = \{q_1^1, ..., q_{k_1}^1, ..., q_1^n, ..., q_{k_n}^n\}.$$

(2) E' is the consistent closure of $\{p_1 \bowtie q_1, ..., p_n \bowtie q_n\} \cup E$. □

The constraint $\{p_1 \bowtie q_1, ..., p_n \bowtie q_n\}$ above is called the *constraint associated with* the resolution step. A *computation* is a finite or countable sequence of states such that if s is the successor of s' then $s \rightarrow s'$. A *success computation* is a computation Γ such that Γ is a countably infinite one or the goal component of the last state of Γ is empty. A *failure computation* is a computation which is not a success one.

3.2 Soundness and Completeness

Given a computation Γ the supports appearing in Γ form a monotone increasing sequence. So the union of these supports is a support, which we call *the support of* Γ. The support of a finite computation Γ is the support at the last state of the computation Γ. A *computation for a goal* Q is a computation which starts from the state $(Q, \emptyset)$.

Theorem 15 (Soundness Theorem) *Let Γ be a success computation for Q and E the support of Γ. Then E is solvable and every solution of E is an answer solution of Q.*

Proof Suppose Γ is a success computation for Q and let E be the support of Γ. As E is consistent and closed by the unification theorem 13 there exists a solution f of E. For every element p of Q it follows from the definitions of $\mathcal{M}_P$ and the computation that $f(p) \in \mathcal{M}_P$. Hence as $p \in Q$ is arbitrary f is an answer solution of Q. □

Lemma 2 (Resolution Step Lemma) *If an assignment f is a solution of E and an answer solution of Q ($\neq \emptyset$) in $\mathcal{M}_P$ then there exist an extension f' of f and a resolution step $(Q, E) \rightarrow (Q', E')$ such that f' is a solution of E' and an answer solution of Q'.*

Proof Let $p \in Q$. As f is an answer solution of p in $\mathcal{M}_P$ by definition of $\mathcal{M}_P$ there exists an extension f'_p of f and a fresh copy (h_p, D_p) of a program clause such that f'_p is an solution of $p \bowtie h_p$ and an answer solution of D_p. Let $Q' = \bigcup\{D_p \mid p \in Q\}$ and let E' be the closure of $C \cup E$, where $C = \{p \bowtie h_p \mid p \in Q\}$. As (h_p, D_p) is a fresh copy the family $\{f'_p\}_{p \in Q}$ is compatible as functions. Hence there is an extension of f which satisfies both C and E. Therefore by the unification theorem 13 E' is consistent and closed. By definition 38 of the resolution step we get finally $(Q, E) \rightarrow (Q', E')$. □

The following completeness theorem is obtained by repeating the resolution step lemma 2.

Theorem 16 (Completeness Theorem) *If f is an answer solution of a goal Q then f extends to a solution of the support of some success computation for Q.*

In the rest of this section we assume that $\mathrm{dom}(f)$ of assignments f is large enough to include all necessary parameters for evaluating expressions in the context. By $sol(D)$ we mean the set of answer solutions of D.

Lemma 3 (Lifting Lemma) *Let $(Q, F) \rightarrow (Q', E')$ be a resolution step. Let F be a support such that $sol(E) \subseteq sol(F)$. Then there exist a support F' and a resolution step $(Q, F) \rightarrow (Q', F')$ such that $sol(E') \subset sol(F')$.*

Proof Let C be the constraint associated with the resolution step $(Q, E) \rightarrow (Q', E')$. By definition of an resolution step E' is the consistent closure of $C \cup E$. Let F' be the closure of $C \cup F$. Clearly $sol(E') \subseteq sol(F')$ follows from $sol(E) \subseteq sol(F)$. Hence as $sol(E')$ is not empty $sol(F')$ is not empty. Therefore F' is a support. As (Q, F) and (Q', F') satisfy all defining clauses in definition 38 of the resolution step we get $(Q, F) \rightarrow (Q', F')$. □

Let Γ be a success computation from the state (Q, E). By repeating application of the lifting lemma we have a success computation Γ' starting from the state $(Q, \emptyset)$. Let (H, D) and (H, D') be corresponding states on the two computations Γ and Γ'. By induction on the number of resolution steps from the initial state it is proved that D is the closure of $E \cup D'$. We call Γ' the *lifting* of Γ.

Definition 29 A parameter x is *free* in a support E if $x \bowtie u \in E$ implies $u = x$. □

For example in the constraint $\{x \bowtie x, z \bowtie ay, u \bowtie v\}$ x, y are free but z, u, v are not free. The following theorem is a counter part of the theorem in Lloyd[14] having the same title.

Theorem 17 (Display Theorem) *Let E be a support such that every solution of E is an answer solution of a goal Q. Let $\mathcal{F}(E)$ be the set of free parameters appearing in E. Then there exists a success computation $\Gamma_\emptyset$ from $(Q, \emptyset)$ such that the restriction of each solution of E to $\mathcal{F}(E)$ can extends to a solution of the support of the computation.*

Proof We assume for the sake of simplicity that there are sufficiently many constants. For each $x \in \mathcal{F}(E)$ let c_x be a new constant such that $c_x \neq c_y (x \neq y)$. Let C be the reflexive closure of $\{x \bowtie c_x \mid x \in M\}$. From the assumption $E' = C \cup E$ is a support. Hence by the completeness theorem there exists a success computation $\Gamma_{E'}$ from (Q, E'). By the lifting lemma there exists a success computation Γ_E from (Q, E). Applying the lifting lemma again to $\Gamma_{E'}$ we obtain a success computation $\Gamma_\emptyset$ from $(Q, \emptyset)$. We use Σ_Δ

for the support of a computation Δ. By the remark above we can construct Γ_E and $\Gamma_\emptyset$ so that Σ_{Γ_E} is the closure of $E \cup \Sigma_{\Gamma_\emptyset}$ and $\Sigma_{\Gamma_{E'}}$ is the closure of $C \cup \Sigma_{\Gamma_E}$.

$$\begin{array}{lll} \Gamma_\emptyset : & (Q,\emptyset) & \rightarrow \cdots \\ \Gamma_E : & (Q,E) & \rightarrow \cdots \\ \Gamma_{E'} : & (Q,E') & \rightarrow \cdots \end{array}$$

As $E \cup \Sigma_{\Gamma_\emptyset}$ has the closure Σ_{Γ_E} for the proof of the theorem it suffices to show that each parameter in $\mathcal{F}(E)$ is free in $\Sigma_{\Gamma_\emptyset}$. Hence it is sufficient to show that the closure of $C \cup \Sigma_{\Gamma_\emptyset}$ is a support. In fact the closure of $C \cup \Sigma_{\Gamma_\emptyset}$ must be a support because the closure of $C \cup E \cup \Sigma_{\Gamma_\emptyset}$ is the support $E_{\Gamma_{E'}}$. □

Theorem 18 (Soundness of NAF) *If there is no success computation from $(Q,\emptyset)$ then Q has no answer solution.*

Proof This is the contraposition of the soundness theorem 15 □

Theorem 19 (Completeness of NAF) *If Q has no answer solution then there is no success computation from $(Q,\emptyset)$.*

Proof This is the contraposition of the completeness theorem 16. □

Due to the maximum semantics the proof of soundness and completeness of negation-as-failure rule has become almost obvious. Infinite computations is always meaningful in $\mathcal{M}_P$, while in the least Herbrand model infinite computations are meaningless.

3.3 DAGs as Constraints on Records

In this subsection we show a relationship between DAGs (directed acyclic graphs) used in unification grammars[25] and record structures. This is done by giving a simple translation from DAGs into constraints on records. In stead of DAGs we treat directed graphs (DGs) as a more general class than the DAG class.

Let X, L, M be a set of *nodes*, *atomic features*, *tags*, respectively. Let R be a free complete and standard G-record algebra over M, where $G \stackrel{\text{def}}{=} L^*$. We assume without loss of generality that R and X are disjoint to each other.

Definition 30 A directed graph (DG) D is a 5–tuple $D = (N, A, f, g, s)$, where $N \subseteq X$, $A \subseteq N \times N$, f is a partial function from N into M, g is a function from A to L, and s is the *root node*. □

Given a DG $D = (N, A, f, g, s)$ let C_D be the least set C such that the following hold:

(1) If $(x,y) \in A$ and $g((x,y)) = a$ then $x \bowtie ay \in C$.

(2) If $f(x) = c$ then $x \bowtie c \in C$.

Viewing C_D as a binary relation on $\mathcal{E}(G, M \cup X, \{\cdot, +\})$ we take C_D as a constraint on G-record algebra R over M. Thus the DG D is a constraint over the record algebra R. Let $D_i = (N_i, A_i, f_i, g_i, s_i)$ $(i = 1,2)$ be two DGs. The graph merge of D_1 and D_2 is the union of the constraint $\{s_1 \bowtie s_2\} \cup C_{D_1} \cup C_{D_2}$. Thus through this translation it is straightforward to give the proposed record algebra semantics to the unification grammar. Our semantics covers some basic part of DAG–based unification grammar theory in Shieber[24].
The notion of *structure sharing* in DAG–based theory corresponds to that of sharing parameters in constraint language $(R, \bowtie)$. From the view point of this translation the

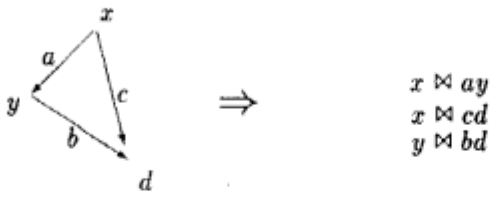

Figure 4: A DAG as a constraint on records

notion of structure sharing belongs only to the constraint language. The structure sharing is not a property of objects but just occurrences of the same parameters.

As records can be infinite or nonwellfounded they can represent more complex structures than (finite) DAGs can do. In particular the record domain seems to be suitable for representing and processing circular situations proposed by Barwise and Etchemendy[5] in addition to the ordinary linguistic information processing.

3.4 Arity in Record Algebra

In this subsection we show an embedding of a (complete) Herbrand domain into a record algebra. This embedding allows us to use nested structures consisting of record terms and standard terms in a uniform way. Let G be a feature monoid. A *sort* (of G) is a prefix closed subset S of G, i.e. if $\alpha\beta \in S$ then $\alpha \in S$. The *arity* of the sort S is defined to be the set of minimal elements of $S \setminus \{\varepsilon\}$. G is *sorted* if there is a family $\{S_j\}_{j\in J}$ of sorts of G such that $G = \bigcup\{S_j \mid j \in J\}$ and $S_j \cap S_{j'} = \{\varepsilon\}$ for $j \neq j'$.

Let F be a set of function symbols. We assume each $f \in F$ is assigned a set $\mathrm{arg}(f)$ of *argument places*. Moreover we assume $\mathrm{arg}(f) \cap \mathrm{arg}(f') = \emptyset$ for $f \neq f'$. G_F is a free monoid over $\bigcup\{\mathrm{arg}(f) \mid f \in F\}$. For $f \in F$ let $S_f = \{\varepsilon\} \cup \{a\alpha \mid a \in \mathrm{arg}(f), \alpha \in G_F\}$. Clearly G_F is sorted with $\{S_f\}_{f\in F}$. Let F_0 be the set of symbols in F which has no argument place.

Definition 31 Given a set F of function symbol $\hat{R}_F$ denotes the free complete G_F record algebra over F_0, where $(F_0, +)$ is a trivial merge system. □

Note that it follows from proposition 7 and corollary 8 that $\hat{R}_F$ is standard.

Definition 32 Given a record $x \in R$ $[x]$ denotes the set of records $y \in R$ such that $y \leq x$. □

Definition 33 A G–record algebra R is *sorted* if there is a family $\{R_i\}_{i\in I}$ of G–record algebras such that

(1) $R = \bigcup\{R_i \mid i \in I\}$.

(2) If $x \in R_i \cap R_j$ and $x \neq \epsilon$ then x is atomic, where $i \neq j$.

(3) For $x, y \in R$ if $x + y\downarrow$ then there is some $i \in I$ such that $x, y \in R_i$.

(4) $\forall x \in R \forall \alpha \in G \exists i \in I\ [x]/\!\!/\alpha\downarrow \Longrightarrow [x]/\!\!/\alpha \subseteq R_i$.

□

In harmony with the introduction of sorts we add the following clause to the definition 15 of the conflict for the record unification theory.

Definition 34 (In addition to definition 15.) Any basic constraint of the form $\alpha u \bowtie \beta v$ is a *conflict*, where α and β belong to distinct sorts of the feature monoid G, $\alpha \neq \varepsilon$, $\beta \neq \varepsilon$ and u, v are record terms. □

For example $f_1x \bowtie g_2y$ is a conflict, whereas $f_1x \bowtie f_2y$ is not a conflict, where f_1 and f_2 are distinct argument places of f and g_2 is an argument place of g, provided that $f \neq g$.

Let F be a set of function symbols and $F_+ = F \setminus F_0$ as above. Let $\mathcal{Z}$ be the set of ordered pairs $(\{S_f\}_{f\in F_+}, S_F)$ such that $S_f \subseteq \hat{R}_F$ and $S_F \subseteq \hat{R}_F$. We define an order relation $\leq$ on $\mathcal{Z}$ by $(\{S_f\}_{f\in F_+}, S_F) \leq (\{S'_f\}_{f\in F_+}, S'_F)$ iff $S_f \subseteq S'_f$ for each $f \in F_+$ and $S_F \subseteq S'_F$.

Definition 35 $\{R_f\}_{f\in F_+}$ and R_F are a family of sets and a set such that the ordered pair $(\{R_f\}_{f\in F_+}, R_F)$ is the largest element in $(\mathcal{Z}, \leq)$ which satisfies the following condition:

(1) If $x \in R_f$ then $x = \{\epsilon\} \cup F_0$ or $x = a_1x_1 + \cdots + a_nx_n$, where $\{a_1, \cdots, a_n\} \subseteq \arg(f)$ and $\{x_1, \cdots, x_n\} \subseteq R_F$.

(2) $R_F = \bigcup\{R_f \mid f \in F_+\}$.

□

Proposition 20 *Given a set of function symbols there is a complete, standard and sorted G_F-record algebra $(R_F, G_F, +, \cdot, /\!\!/, \epsilon)$ over F_0 w.r.t. $\{R_f \mid f \in F_+\}$ satisfying definition 35.*

Proof It is clear by definition. □

Let L be a set of atomic features, F a set of function symbols, $F_0 = \{f \in F \mid \arg(f) = \emptyset\}$, $L_F = \bigcup\{\arg(f) \mid f \in F\}$, and X a set of parameters. To introduce DCG over the record algebras in the below we define terms over X, L, F expressing records and an embedding translation τ between standard terms and record terms.

Definition 36 Let L and F be as the above and let P be a set. Then $\mathcal{T}(L, F, P)$ is the *least* set T such that the following hold:

(1) $P \subseteq T$.

(2) If $a_1, \ldots, a_n$ are features and $p_1, \ldots, p_n \in T$ then the set $\{(a_1, p_1), \ldots, (a_n, p_n)\}$ is in T.

(3) If $f \in F$ is a function symbol of arity $n \leq$ and $p_1, \ldots, p_n \in T$ then the form $f(p_1, \ldots, p_n)$ is in T.

□

An element of $\mathcal{T}(L, F, X)$ is called a *term*. Let $G = (L \cup L_F)^*$.

Definition 37 A *translation* τ is a partial function from $\mathcal{T}(L, F, X)$ into $\mathcal{E}(G, F_0 \cup X, \{\cdot, +\})$ such that the following hold:

(1) If $u \in F_0 \cup X$ then $\tau(u) = u$.

(2) $\tau(\{(a_1, p_1), \ldots, (a_n, p_n)\}) = a_1\tau(p_1) + \cdots + a_n\tau(p_n)$, where $a_i \in L$.

(3) $\tau(f(p_1, \ldots, p_n)) = b_1\tau(p_1) + \cdots + b_n\tau(p_n)$, where $\arg(f) = \{b_1, \cdots, b_n\}$.

□

H_F denotes the Herbrand universe over F. By $(H_F, =)$ we mean the standard unification theory over H_F. We take the theory $(H_F, =)$ as a familiar congruence closure operation on sets of standard term equations. By $(R_F, \bowtie)$ we mean the theory of sorted record constraints taken as the closure operation given by table 1. Now we are at the place to state and prove that $(R_F, \bowtie)$ is a 'conservative extension' of $(H_F, =)$.

Theorem 21 *Let s and t be first order terms then the following are equivalent.*

(1) $s = t$ is solvable in $(H_F, =)$.

(2) $\tau(s) \bowtie \tau(t)$ *is solvable in* $(R_F, \bowtie)$.

Proof Let C be a constraint in $(H_F, =)$. As the unifiability and satisfiability is equivalent in $(H, =)$ ([14]) and $(R, \bowtie)$ (theorem 13), respectively, it is straightforward to show that for any standard terms s, t the following are equivalent.

(1) C is the consistent closure of $s = t$ in $(H, =)$

(2) $\tau(C)$ is the consistent closure of $\tau(s) \bowtie \tau(t)$ in $(R, \bowtie)$.

□

3.5 Definite Clause Grammar over Records

As an application of record algebras we extend definite clause grammar (DCG) over a Herbrand universe to that over a record algebra. Let R be a G–record algebra over a merge system M, where G is a feature monoid. A *DCG* is a finite set of *rules* of the following form:

$$p_0 \leftarrow q_1 \bowtie r_1, \ldots, q_m \bowtie r_m \mid p_1, \ldots, p_n$$

where p_i, q_j, $r_j \in \mathcal{E}(G, M \cup X, \{\cdot, +\})$, $n \geq 0$, $m \geq 0$, and X is a set of parameters. In the same way as for programs over the record algebra R the *semantics* of a DCG, say D, over R is defined to be the largest subset $\mathcal{M}_D$ of R such that the following hold: Any $t \in \mathcal{M}_D$ there exists some DCG rule $p \leftarrow C \mid Q$ in D and assignment f such that the following hold:

(1) t is an instance of p with f.

(2) f satisfies the constraint C.

(3) f has some extension f' such that every element of Q has an instance in $\mathcal{M}_D$ with f'.

Also an operational semantics of a DCG is defined in the same way for the program semantics except a slight modification of definition 38 of the constraint associated with resolution steps as follows:

Definition 38 An ordered pair (s, s') of two states $s = (Q, E)$ and $s' = (Q', E')$ is a *resolution step*, written $s \rightarrow s'$, if the following hold, where $Q = \{p_1, \ldots, p_n\}$:

(1) $Q' = \{q_1^1, \ldots, q_{k_1}^1, \ldots, q_1^n, \ldots, q_{k_n}^n\}$ for some fresh copies $q_i \leftarrow C_i \mid q_1^i, \ldots, q_{k_i}^i$ $(1 \leq i \leq n)$ of rules in D.

(2) E' is the consistent closure of $\{p_1 \bowtie q_1, \ldots, p_n \bowtie q_n\} \cup E \cup C_1 \cup \cdots \cup C_n$.

□

The set $\{p_1 \bowtie q_1, \ldots, p_n \bowtie q_n\} \cup C_1 \cup \cdots \cup C_n$ is called the constraint *associated with* the resolution step. The same results about soundness and completeness are obtained in almost the same way as in the case of program semantics.

For an illustration we show a simplified interpreter for DCG and a sample DCG. Let L be a set of atomic features, F a set of function symbols, $F_0 = \{f \in F \mid \arg(f) = \emptyset\}$, and X a set of parameters. Define $F' = F \cup \{\#\}$ for some new function symbol $\# \notin F$ so that $\arg(\#) = L$. So by proposition 20 we have the sorted complete and standard $G_{F'}$ record algebra $R_{F'}$ over F_0' defined for F'.

Thus we can precisely say that the example below is a DCG over the record algebra $R_{F'}$ and is written in $\mathcal{E}(G_{F'}, F_0' \cup X, \{\cdot, +\})$. Infix notations are used freely as in the standard Prolog. Unit clauses (2) and (3) below are for lexical items. The equality $=$ in the body of (1) means the builtin constraint $\bowtie$. a/b is used for the ordered pair (a, b).

Each clause r there means a grammar rule $\Theta(r)$ defined by the following equations, where τ is the translation in definition 3.4.

$$\begin{aligned}\Theta((p,C,B)) &\stackrel{\text{def}}{=} (\tau(p),\tau'(C),\tau''(B)).\\ \tau'(C) &\stackrel{\text{def}}{=} \{\tau(q)\bowtie\tau(r)\mid q=r\in C\}.\\ \tau''(B) &\stackrel{\text{def}}{=} \{\tau(q)\mid q\in B\}.\end{aligned}$$

```
(1) {cat/s, head/H}<-  H={subject/H1} |
        {cat/np, head/H1},
        {cat/vp, head/H}.
(2) lex(jack, {cat/np, head/jack}).
(3) lex(runs, {cat/vp, head/{subject/X,
                                pred/run(X)}}).
```

The clauses from (4) to (8) describe a simplified interpreter for the DCG grammars.

```
(4) parse([X|Y]-Y, F)<- lex(X, F).
(5) parse(X-Y, (A, B))<-
        parse(X-Z, A),
        parse(Z-Y, B).
(6) parse(X-Y, F)<-
        (F<-B),
        parse(X-Y, B).
```

The execution of the grammar looks like this:

```
?-parse([jack, runs]-□, F).

F={cat/s,{head/{subject/jack, pred/run(jack)}}}.
```

4 Concluding Remarks

Several important relevant issues on feature structure such as complement and disjunction feature constructors are out of place. Also set values as feature values[23] and unification under inheritance hierarchies[27] are not considered in this paper. In the programming language CIL[16], from which the record algebra came out, however, full first order terms possibly with parameters are allowed to be features like $brother(1)$ and $brother(2)$ in

$$\{(brother(1),John),(brother(2),Jack)\}.$$

Also this aspect is not treated in this paper.

We have put a condition onto the structure of feature monoids so that they are essentially the same as free monoids. It is an open problem to extend the notion of feature monoids as wide as possible so that the intuitive notion of feature structures is still preserved.

A hyperset theoretical approach to feature structures is discussed to some extent in [19], which treats the bisimulation and subsumption relations with disjunction and negation. A unification of this approach and the present work should be studied in the near future.

Acknowledgments

I would like to thank anonymous referee of an earlier version of this paper for giving useful comments. Also I would like to thank Prof. Hozumi Tanaka of TIT for his continuous encouragement to this work.

References

[1] P. Aczel. *Non-well-founded Sets.* CSLI lecture notes Number 14. CSLI Publications, Stanford University, 1988.

[2] H. Aït-Kaci. *A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures.* PhD thesis, Computer and Information Science, University of Pennsylvania, 1984.

[3] J. Barwise. The situation in logic- III: Situations, sets and the axiom of foundation. In Wilkiw et al, editor, *Proceedings of the 1984 Logic Summer School (Studies in Logic).* Amsterdam: North-Holland, 1986. Also Report No. CSLI-85-26, Stanford: CSLI Publications.

[4] J. Barwise. *The Situation in Logic.* CSLI Lecture Notes 17. Stanford: CSLI Publications, 1989.

[5] J. Barwise and J. Etchemendy. *The Liar: An Essay on Truth and Circular Propositions.* Oxford Univ. Press, 1987.

[6] J. Barwise and J. Perry. *Situations and Attitudes.* MIT Press, 1983.

[7] C. Chevalley. *Fundamental Concepts of Algebra.* Academic Press, 1956.

[8] A. Colmerauer. Prolog II: Reference manual and theoretical model. Technical report, Groupe Intelligence Artificielle, Universite d'Aix-Marseille II, 1982.

[9] B. Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25:95–169, 1983.

[10] J.A. Goguen and J. Meseguer. Order-sorted algebra I: Partial and overloaded operators, errors and inheritance. Technical report, SRI International and CSLI, 1985.

[11] S. Iyanaga and K. Kodaira. *Introduction to Modern Mathematics (I).* Tokyo: Iwanami Shoten, 1961. (in Japanese).

[12] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, 1987.

[13] R.T. Kasper and W. Rounds. A logical semantics for feature structures. In *Proceedings of the 24th Annual Meeting of the ACL, Columbia University.* ACL, 1986. New York, N.Y.

[14] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1984.

[15] M. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees, 1988. draft.

[16] K. Mukai. Partially specified term in logic programming for linguistic analysis. In *Proceedings of International Conference on the Fifth Generation Computer Systems.* Institute for New Generation Computer Technology, 1988. (Also appears as ICOT-TM 568, 1988).

[17] K. Mukai. Coinductive semantics of Horn clause with compact constraint. Technical Report TR-562, ICOT, 1990. Presented at the second conference on Situation Theory and its Application at Scotland, September 1990.

[18] K. Mukai. Merge structure with an operator domain and its unification theory. *JSSST*, 7(2), 1990. (in Japanese).

[19] K. Mukai. A system of logic programming for linguistic analysis. Technical Report TR-540, ICOT, 1990. To appear also from SRI Tokyo series.

[20] K. Mukai and H. Yasukawa. Complex indeterminates in Prolog and its application to discourse models. *New Generation Computing*, 3(4), 1985.

[21] F.C.N. Pereira and S.M. Shieber. The semantics of grammar formalisms seen as computer languages. In *Proceedings of the Tenth International Conference on Computational Linguistics*, Stanford University, 1984.

[22] C.J. Pollard. Toward anadic situation semantics. Manuscript, 1985.

[23] W. Rounds. Set values for unification–based grammar formalisms and logic programming. Technical Report Report No. CSLI-88-129, CSLI, 1988.

[24] S.M. Shieber. *An Introduction to Unification–based Approaches to Grammar.* CSLI Lecture Notes No.4. CSLI publications, Stanford University, 1986.

[25] S.M. Shieber, F.C.N. Pereira, L. Karttunen, and M. Kay. A compilation of papers on unification-based grammar formalisms, Parts I and II. Technical Report CSLI-86-48, CSLI, April 1986.

[26] G. Smolka. Feature logic with subsorts. Technical Report LILOG Report 33, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, W. Germany, May 1989.

[27] G. Smolka and H. Aït-Kaci. Inheritance hierarchies: Semantic and unification. *Journal of Symbolic Computation*, 7, 1989.

[28] B.L. van der Waerden. *Algebra I, II.* Springer Verlag, 1955.