Multiple-World Extension of Clausal Logical Structures

Kiyoshi Akama
Information Initiative Center
Hokkaido University
Sapporo, Hokkaido, Japan
Email: akama@iic.hokudai.ac.jp

Ekawit Nantajeewarawat
Computer Science Program
Sirindhorn International Institute of Technology
Thammasat University
Pathumthani, Thailand
Email: ekawit@siit.tu.ac.th

Tadayuki Yoshida
Faculty of Computer Science
Hokkaido University
Sapporo, Hokkaido, Japan
Email: tadayuki@sh.rim.or.jp

***Abstract*—In a simple clausal logical structure, a set of clauses determines models, each of which is a subset of some predetermined set $\mathcal{G}$ and represents a possible state of a world. A multiple-world clausal logical structure is an extension of a simple clausal logical structure, wherein multiple worlds are considered simultaneously. Each world model may be related not only to its corresponding set of clauses, but also to other world models. We define a method for multiple-world extension of a clausal logical structure. By applying this extension, we propose a new multiple-world clausal logical structure for S-expression atoms with definitions of several concrete constraints, which together provide rich expressive power including function variables and concepts such as argmax and negation.**

Keywords–Multiple-world logical structure; Referential constraint; Specialization system; Declarative description.

I. INTRODUCTION

An abstract notion of a logical structure has been introduced since 2006 [1], by which the general concepts of descriptions (formulas), interpretations, models, and their interrelations are formalized axiomatically. Let $\mathcal{G}$ denote the set of all ground first-order atoms. First-order formulas constitute a logical structure, where interpretations and models are subsets of $\mathcal{G}$. Similarly, clause sets form another logical structure, where interpretations and models are again subsets of $\mathcal{G}$. A logical structure defines a relation between a description k and a model x of k, i.e., $x \in \mathit{Models}(k)$, where *Models* is a mapping that determines the set of all models of a given description.

Basic methods for construction of logical structures have been proposed by Akama and Nantajeewarawat [2], including generation of a logical structure from a specialization system, construction of a conjunctive logical structure, and logical structure morphing. Such construction methods are essential for formulating and understanding new logical structures and extending existing logical structures to obtain richer expressive power, computation efficiency, and computation completeness [3].

This paper proposes a new method for constructing logical structures, by which a logical structure for dealing with multiple worlds is constructed from simple clausal logical structures. This is typically explained by the following system of membership constraints:

$$x_{\mathrm{J}} \in \mathit{Models}(k_{\mathrm{J}}, x_{\mathrm{J}}, x_{\mathrm{H}})$$
$$x_{\mathrm{H}} \in \mathit{Models}(k_{\mathrm{H}}, x_{\mathrm{J}}, x_{\mathrm{H}})$$

They together represent a situation in which there are two persons, say John and Henry, and their beliefs, say x_{J} and x_{H}, respectively, are described by k_{J} and k_{H} together with the beliefs themselves.

There are two main important features:

1) There may be multiple worlds; a pair of a description and a set of possible models is called a *world*.
2) The mapping *Models* is used and it takes models as arguments.

By applying this construction method to an S-expression clausal logical structure, we propose a multiple-world clausal logical structure for S-expression atoms. We show that this extension provides rich expressive power.

For example, when we want to find all persons who earn maximum salary in a department, we may write the following clause:

$$(h\ *\mathit{name}) \leftarrow (\mathit{argmax}\ *\mathit{name}\ (*n\ *s)\ ((\mathit{emp}\ *n\ *a\ *s\ *d))).$$

The predicate $argmax$ takes three arguments. The third one is in general a sequence of atoms. Its second and third arguments are used to represent a relation between names and salaries:

$$S = \{(n, s) | (\mathit{emp}\ n\ a\ s\ d)\}.$$

We need to maximize s by changing n in the set S. The predicate $argmax$ can not be defined by usual clauses. By the theory to be proposed in this paper, $argmax$ is defined as a new type of constraint. By adding a clause C, given by

$$C = ((p\ *n\ *s) \leftarrow (\mathit{emp}\ *n\ *a\ *s\ *d)),$$

to a set *Cs* of clauses, the new multiple-world semantics proposed in this paper allows us to construct a constraint that refers to a model of $\mathit{Cs} \cup \{C\}$, and thus, to refer to its subset S.

Multiple-world extension of clausal logical structures is also useful for representing other database constraints such as max, min, summation, and average. Moreover negation is also represented by using the same semantics. Together with function variables, the proposed logical structure has a very rich expressive power that is purely declarative and it is rigorously formulated based on an abstract definition of a logical structure.

 ISBN: 978-1-61208-354-4

The rest of the paper is organized as follows: Section II recalls an abstract notion of a logical structure and its related basic concepts, and identifies the main objective of the paper. Section III provides a general notion of a specialization system, and defines a specialization system for S-expressions and that for function variables and function constants. Section IV formalizes constraints, defines declarative descriptions along with their models, associates with a declarative description a system of membership constraints, and formulates a logical structure for declarative descriptions. Section V shows rich expressive power of an S-expression-based logical structure, which is directly created by instantiation of the general theory, by defining constraints and *func*-atoms, which cannot be supplied by the conventional first-order logic. Section VI provides conclusions.

Preliminary Notation

The following notation will be used. For any sets A and B, *pow*(A) denotes the power set of A, *Map*(A, B) the set of all mappings from A to B, and *partialMap*(A) the set of all partial mappings on A (i.e., from A to A itself). Given a binary relation r, $dom(r)$ and $ran(r)$ denote the domain and the range of r, respectively. For any partial mappings f and g such that $dom(f) \supseteq ran(g)$, $f \circ g$ denotes the composition of f with g, i.e., for each $a \in dom(g)$, $(f \circ g)(a) = f(g(a))$. Let *Bool* $= \{$*true*, *false*$\}$.

II. Logical Structures and Basic Related Concepts

A. Logical Structures

An abstract notion of a logical structure [1] is recalled first.

Definition 1: Let $\mathcal{G}$ be a set. A *logical structure* $\mathcal{L}$ on $\mathcal{G}$ is a triple $\langle \mathcal{K}, \mathcal{I}, \nu \rangle$, where

1) $\mathcal{K}$ is a set,
2) $\mathcal{I} =$ *pow*$(\mathcal{G})$,
3) $\nu : \mathcal{K} \rightarrow$ *Map*$(\mathcal{I},$ *Bool*$)$.

An element of $\mathcal{K}$ is called a *description*. An element of $\mathcal{I}$ is called an *interpretation*. ■

Given an arbitrary set $\mathcal{G}$, a logical structure $\mathcal{L} = \langle \mathcal{K}, \mathcal{I}, \nu \rangle$ on $\mathcal{G}$ is a representation system. Intuitively, when we have a subset G of $\mathcal{G}$ in mind as a target atom set to be represented, we can provide information about the target atom set G using some description $k \in \mathcal{K}$, i.e., G satisfies the condition $\nu(k)(G) =$ *true*.

The concept of an interpretation in Definition 1 is very different from an interpretation in the usual logic [4][5]. The set $\mathcal{G}$ in Definition 1 is more similar to the Herbrand Base [6] for a given description, but they are not exactly the same. The Herbrand Base depends on a given description (logical formula). There is no common set like $\mathcal{G}$ in the definition of the Herbrand Base. However, in our theory, a unique set $\mathcal{G}$ is firstly given. The objective of defining a logical structure is to define a representation system in which a description determines a set of possible subsets of $\mathcal{G}$.

Despite its simplicity, the abstract notion given by Definition 1 provides a sufficient structure for defining the concepts of logical equivalence, models, satisfiability, and logical consequence, which will be given in Definition 2.

Definition 2: Let $\mathcal{L} = \langle \mathcal{K}, \mathcal{I}, \nu \rangle$ be a logical structure. Two descriptions $k_1, k_2 \in \mathcal{K}$ are *logically equivalent* in $\mathcal{L}$, denoted by $k_1 \equiv_{\mathcal{L}} k_2$, iff $\nu(k_1) = \nu(k_2)$. An interpretation $I \in \mathcal{I}$ is a *model* in $\mathcal{L}$ of a description $k \in \mathcal{K}$ iff $\nu(k)(I) =$ *true*. A description $k \in \mathcal{K}$ is *satisfiable* in $\mathcal{L}$ iff there exists a model in $\mathcal{L}$ of k. A description $k_1 \in \mathcal{K}$ is a *logical consequence* in $\mathcal{L}$ of a description $k_2 \in \mathcal{K}$, denoted by $k_2 \models_{\mathcal{L}} k_1$, iff every model in $\mathcal{L}$ of k_2 is a model in $\mathcal{L}$ of k_1. ■

For these concepts to have practical meanings, it is necessary to give appropriate structure to elements of $\mathcal{K}$ and $\mathcal{I}$ and also to the mapping ν. This process is called logical structure instantiation, which is illustrated below.

B. Logical Structure Instantiation: Examples

Instantiation of the abstract notion of a logical structure into first-order logic and into a clausal logical system is illustrated below.

1) First-Order Logic: Let $\mathcal{G}$ be the set of all variable-free atomic formulas (atoms) and $\mathcal{K}$ the set of all closed first-order formulas. Let ν be defined by: for any $k \in \mathcal{K}$ and any $G \subseteq \mathcal{G}$, $\nu(k)(G) =$ *true* iff k is true with respect to G. Then $\langle \mathcal{K},$ *pow*$(\mathcal{G}), \nu \rangle$ is a logical structure on $\mathcal{G}$.

2) Clausal Logical Systems: Let $\mathcal{G}$ be the set of all variable-free atoms and $\mathcal{K}$ the set of all sets of clauses. Let ν be defined by: for any $k \in \mathcal{K}$ and any $G \subseteq \mathcal{G}$, $\nu(k)(G) =$ *true* iff all clauses in k are true with respect to G. Then $\langle \mathcal{K},$ *pow*$(\mathcal{G}), \nu \rangle$ is a logical structure on $\mathcal{G}$.

C. The Primary Objective of the Paper

The objective of the paper is twofold:

1) To develop a method for multiple-world extension of logical structures (Section IV).
2) To introduce a concrete logical structure in the S-expression space by the instantiation of the proposed method (Section V).

The resulting concrete logical structure provides multiple-world knowledge representation based on an extended space with (i) clauses and iff-formulas, and with (ii) function variables and function constants.

III. Specialization Systems

A. Specialization Systems

The notion of a specialization system [7] is recalled below.

Definition 3: A *specialization system* Γ is a quadruple $\langle \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu \rangle$ of three sets $\mathcal{A}$, $\mathcal{G}$, and $\mathcal{S}$, and a mapping μ from $\mathcal{S}$ to *partialMap*$(\mathcal{A})$ that satisfies the following conditions:

1) $(\forall s', s'' \in \mathcal{S})(\exists s \in \mathcal{S}) : \mu(s) = \mu(s') \circ \mu(s'')$.
2) $(\exists s \in \mathcal{S})(\forall a \in \mathcal{A}) : \mu(s)(a) = a$.
3) $\mathcal{G} \subseteq \mathcal{A}$.

Elements of $\mathcal{A}$, $\mathcal{G}$, and $\mathcal{S}$ are called *atoms*, *ground atoms*, and *specializations*, respectively. The mapping μ is called the

specialization operator of Γ. A specialization $s \in \mathcal{S}$ is said to be *applicable* to $a \in \mathcal{A}$ iff $a \in dom(\mu(s))$. ■

In the sequel, assume that a specialization system $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu \rangle$ is given. A specialization in $\mathcal{S}$ will often be denoted by a Greek letter such as θ. A specialization $\theta \in \mathcal{S}$ will be identified with the partial mapping $\mu(\theta)$ and used as a postfix unary (partial) operator on $\mathcal{A}$ (e.g., $\mu(\theta)(a) = a\theta$), provided that no confusion is caused. Let ϵ denote the identity specialization in $\mathcal{S}$, i.e., $a\epsilon = a$ for any $a \in \mathcal{A}$. For any $\theta, \sigma \in \mathcal{S}$, let $\theta \circ \sigma$ denote a specialization $\rho \in \mathcal{S}$ such that $\mu(\rho) = \mu(\sigma) \circ \mu(\theta)$, i.e., $a(\theta \circ \sigma) = (a\theta)\sigma$ for any $a \in \mathcal{A}$. A subset A' of $\mathcal{A}$ is said to be *closed* iff $a\theta \in A'$ for any $a \in A'$ and any $\theta \in \mathcal{S}$.

B. A Specialization System for S-Expressions

Next, a specialization system for S-expressions is introduced. It will be used for providing concrete examples of constraints and declarative descriptions in Section V.

An alphabet $\Delta = \langle \mathbf{K}, \mathbf{V} \rangle$ is assumed, where $\mathbf{K}$ is a countably infinite set of constants, $\mathbf{V}$ is a countably infinite set of variables, $\mathbf{K}$ and $\mathbf{V}$ are disjoint, and *nil* $\in \mathbf{K}$. Assume that $\mathbf{K}$ includes the set of all numbers. As notational conventions, each variable begins with an asterisk (e.g., $*x$), while constants do not.

An *S-expression* (*symbolic expression*) on Δ is defined inductively as follows:

1) A constant in $\mathbf{K}$ is an S-expression on Δ.
2) A variable in $\mathbf{V}$ is an S-expression on Δ.
3) If a and a' are S-expressions on Δ, then $(a|a')$ is an S-expression on Δ.

An S-expression $(a_1|(a_2|\cdots|(a_n|\mathit{nil})\cdots))$ is often written as $(a_1\ a_2\ \cdots\ a_n)$. The S-expression *nil* is often written as $()$.

A *substitution* on Δ is a finite set of bindings $\{v_1/a_1, \ldots, v_n/a_n\}$ such that for each $i \in \{1, \ldots, n\}$, $v_i \in \mathbf{V}$ and a_i is an S-expression on Δ such that $v_i \neq a_i$.

A specialization system $\Gamma_{\mathrm{S}} = \langle \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu \rangle$ for S-expressions is defined by:

1) $\mathcal{A}$ is the set of all S-expressions on Δ.
2) $\mathcal{G}$ is the set of all variable-free S-expressions on Δ.
3) $\mathcal{S}$ is the set of all substitutions on Δ.
4) $\mu : \mathcal{S} \to \mathit{partialMap}(\mathcal{A})$ such that for any $a \in \mathcal{A}$ and any $s \in \mathcal{S}$, $\mu(s)(a)$ is the S-expression obtained from a by simultaneously applying all bindings in s to a.

C. A Specialization System for Function Variables and Function Constants

The conventional Skolemization is not a meaning-preserving transformation [6]. Meaning-preserving Skolemization was recently developed in [3], in which function variables are used in place of Skolem functions. For example,

$$(\mathit{motherOf}\ *x\ *y) \leftarrow \langle f_{\mathit{func}}, h_1, *y, *x \rangle$$

is used for representing the statement "every person has his/her mother", where h_1 is a function variable, which determines a mother $*x$ of $*y$ by $*x = f_1(*y)$ using some unknown function f_1 obtained by instantiation of h_1. The specialization system Γ_{F} is introduced below for function variables and their instantiations.

Let $\mathcal{FV}$ be the set of all function variables, and let

$$\mathcal{FC} = \bigcup_{n \in \mathbb{N}} \mathit{Map}(\mathcal{G}^n, \mathcal{G}),$$

where $\mathbb{N}$ is the set of all nonnegative integers. Elements of $\mathcal{FC}$ are called *function constants*. Let Γ_{F} be defined as a specialization system $\langle \mathcal{A}_{\mathrm{F}}, \mathcal{G}_{\mathrm{F}}, \mathcal{S}_{\mathrm{F}}, \mu_{\mathrm{F}} \rangle$ as follows:

1) $\mathcal{A}_{\mathrm{F}} = \mathcal{FV} \cup \mathcal{FC}$,
2) $\mathcal{G}_{\mathrm{F}} = \mathcal{FC}$,
3) $\mathcal{S}_{\mathrm{F}}$ the set of all substitutions on $\langle \mathcal{FV}, \mathcal{FV} \cup \mathcal{FC} \rangle$,
4) $\mu_{\mathrm{F}} : \mathcal{S}_{\mathrm{F}} \to \mathit{partialMap}(\mathcal{A}_{\mathrm{F}})$ such that for any $f \in \mathcal{A}_{\mathrm{F}}$ and any $s \in \mathcal{S}_{\mathrm{F}}$, $\mu(s)(f)$ is obtained from f by applying the substitution s to it.

IV. Declarative Descriptions and Their Models

A description, called a *declarative description*, is introduced in this section. Declarative descriptions form a knowledge representation scheme with the following characteristics: (i) multiple-world representation, (ii) clauses and iff-formulas, and (iii) function variables and function constants.

A. Constraints

Assume that $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu \rangle$ and $\Gamma_{\mathrm{c}} = \langle \mathcal{A}_{\mathrm{c}}, \mathcal{G}_{\mathrm{c}}, \mathcal{S}, \mu_{\mathrm{c}} \rangle$ are specialization systems with the set $\mathcal{S}$ of specializations in common. They are used below, along with the specialization system Γ_{F} (Section III-C), for defining constraints, clauses, and declarative descriptions.

A constraint is an expression such that the truth value of its ground instantiation is predetermined. In this paper, constraint is represented by using a tuple enclosed by a pair of angle brackets. A constraint of the simplest type uses only terms as its arguments, e.g., an equality constraint $\langle f_{eq}, t_1, t_2 \rangle$. We also use function variables/constants as arguments. For example $\langle f_{\mathit{func}}, h_0, t, t' \rangle$ is a constraint with a unary function variable h_0, and it is true iff $h_0(t) = t'$. The most important type of constraints for multiple-world semantics is the one that refers to models of some sets of clauses. For example, $\langle f_{\mathit{not}}, (p\ *x), l_2 \rangle$ is a *not*-constraint, which has a world label l_2 that refers to a model of a set of clauses. Intuitively, $\langle f_{\mathit{not}}, (p\ *x), l_2 \rangle$ means that $(p\ *x) \notin \mathit{model}(l_2)$, where $\mathit{model}(l_2)$ is a model of the world l_2.

Definition 4: Let L be a set of labels. A *constraint* on $\langle \Gamma, \Gamma_{\mathrm{c}}, L, \Gamma_{\mathrm{F}} \rangle$ is an $(m+1)$-tuple $\langle \phi, d_1, d_2, \ldots, d_m \rangle$, where

1) $m \geq 0$,
2) $\phi : (\mathcal{G}_{\mathrm{c}} \cup \mathcal{G} \cup \mathit{pow}(\mathcal{G}) \cup \mathcal{G}_{\mathrm{F}})^m \to \mathit{Bool}$, and
3) for each $i \in \{1, \ldots, m\}$, $d_i \in \mathcal{A}_{\mathrm{c}} \cup \mathcal{A} \cup L \cup \mathcal{A}_{\mathrm{F}}$.

A constraint $\langle \phi, d_1, d_2, \ldots, d_m \rangle$ on $\langle \Gamma, \Gamma_{\mathrm{c}}, L, \Gamma_{\mathrm{F}} \rangle$ is called a *referential constraint* if d_i is a label in L for some $i \in \{1, \ldots, m\}$. It is called a *simple constraint* otherwise. It is called a *ground constraint* iff for any $i \in \{1, \ldots, m\}$, $d_i \in \mathcal{G}_{\mathrm{c}} \cup \mathcal{G} \cup L \cup \mathcal{G}_{\mathrm{F}}$. Let $\textsc{Con}(\Gamma, \Gamma_{\mathrm{c}}, L, \Gamma_{\mathrm{F}})$ and $\textsc{GCon}(\Gamma, \Gamma_{\mathrm{c}}, L, \Gamma_{\mathrm{F}})$ denote the set of all constraints and that of all ground constraints, respectively, on $\langle \Gamma, \Gamma_{\mathrm{c}}, L, \Gamma_{\mathrm{F}} \rangle$. ■

 ISBN: 978-1-61208-354-4

A specialization $\theta \in \mathcal{S}$ is always applicable to labels and elements in $\mathcal{A}_{\rm F}$ and its application does not change them, i.e., for any label l and any $a \in \mathcal{A}_{\rm F}$, the results of applying θ to l and a, denoted by $l\theta$ and $a\theta$, are l and a, respectively, themselves. Given a constraint $c = \langle \phi, d_1, d_2, \ldots, d_m \rangle$ and a specialization $\theta \in \mathcal{S}$, (i) θ is applicable to c iff it is applicable to each of $d_1, d_2, \ldots, d_m$, and (ii) when applicable, the result of applying θ to c, denoted by $c\theta$, is defined by $c\theta = \langle \phi, d_1\theta, d_2\theta, \ldots, d_m\theta \rangle$. A specialization $\theta \in \mathcal{S}_{\rm F}$ is always applicable to elements of $\mathcal{A}_{\rm c} \cup \mathcal{A} \cup L$, and its application does not change them. A specialization $\theta \in \mathcal{S}_{\rm F}$ is always applicable to an element f in $\mathcal{A}_F$, and its application changes it into $f\theta$.

B. Clauses and Iff-Formulas

A declarative description proposed in this paper consists of clauses and iff-formulas. Some set of clauses can be equivalently and more concisely represented by an iff-formula with higher chance of transformation. For example, the set of three clauses

$$(q *A *B), (r *B) \leftarrow (p *A *B),$$
$$(p *A *B) \leftarrow (q *A *B),$$
$$(p *A *B) \leftarrow (r *B)$$

can be represented by the iff-formula

$$(p * A * B) \leftrightarrow (\{(q *A *B)\} \vee \{(r *B)\}).$$

Definition 5: Let L be a set of labels. A *clause* C on $\langle \Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F} \rangle$ is an expression of the form

$$a_1, a_2, \ldots, a_m \leftarrow b_1, b_2, \ldots, b_n,$$

where (i) $m, n \geq 0$, (ii) for each $i \in \{1, \ldots, m\}$, a_i is an atom in $\mathcal{A}$, and (iii) for each $j \in \{1, \ldots, n\}$, b_j is an atom in $\mathcal{A}$ or a constraint on $\langle \Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F} \rangle$ (i.e., $b_j \in \mathcal{A} \cup \textsc{Con}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$). The set $\{a_1, a_2, \ldots, a_m\}$ is called the *left-hand side* of C, denoted by *lhs*(C), and the set $\{b_1, b_2, \ldots, b_n\}$ is called the *right-hand side* of C, denoted by *rhs*(C). The set *rhs*$(C) - \mathcal{A}$ is denoted by *con*(C). C is called a *referential clause* if there exists $i \in \{1, \ldots, n\}$ such that b_i is a referential constraint. It is called a *simple clause* otherwise. It is called a *ground clause* iff (i) for each $i \in \{1, \ldots, m\}$, $a_i \in \mathcal{G}$, and (ii) for each $j \in \{1, \ldots, n\}$, $b_j \in \mathcal{G} \cup \textsc{GCon}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$. Let $\textsc{CL}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$ and $\textsc{GCL}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$ denote the set of all clauses and that of all ground clauses, respectively, on $\langle \Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F} \rangle$. Let $\textsc{GCL}(\mathcal{G})$ denote the set of all ground clauses that consist only of atoms in $\mathcal{G}$. ■

Given a clause $C = (a_1, a_2, \ldots, a_m \leftarrow b_1, b_2, \ldots, b_n)$ and a specialization $\theta \in \mathcal{S} \cup \mathcal{S}_{\rm F}$, (i) θ is applicable to C iff it is applicable to each of $a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_n$, and (ii) when applicable, the result of applying θ to C, denoted by $C\theta$, is defined by $C\theta = (a_1\theta, a_2\theta, \ldots, a_m\theta \leftarrow b_1\theta, b_2\theta, \ldots, b_n\theta)$.

Definition 6: An *if-and-only-if formula* (for short, *iff-formula*) I on $\langle \Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F} \rangle$ is a formula of the form

$$a \leftrightarrow (conj_1 \vee conj_2 \vee \cdots \vee conj_n),$$

where (i) $n \geq 0$, (ii) $a \in \mathcal{A}$, and (iii) each of the $conj_i$ is a finite subset of $\mathcal{A} \cup \textsc{Con}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$. For each $i \in \{1, \ldots, n\}$, $conj_i$ corresponds to the conjunction $\bigwedge\{b \mid b \in conj_i\}$ of atoms and constraints in $conj_i$, and this conjunction is denoted by $\mathbf{F}(conj_i)$. The atom a is called the *head* of the iff-formula I, denoted by *head*(I); n is called the *number of conjunctions* in I, denoted by $\#$*Conj*(I); and for each $i \in \{1, \ldots, n\}$, $conj_i$ is called the *ith conjunction in the right-hand side* of I, denoted by *rhs*$_i(I)$. The set of all constraints occurring in I is denoted by *con*(I). I is called a *ground* iff-formula if a, $conj_1, conj_2, \ldots, conj_n$ are all ground. When emphasis is given to its head, an iff-formula whose head is an atom a is often referred to as *iff*(a). An iff-formula $I = (a \leftrightarrow (conj_1 \vee \cdots \vee conj_n))$ corresponds to the universally quantified formula $\forall(a \leftrightarrow (\mathbf{F}(conj_1) \vee \cdots \vee \mathbf{F}(conj_n)))$, which is denoted by $\mathbf{F}(I)$. Let $\textsc{IFF}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$ and $\textsc{GIFF}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$ be the set of all iff-formulas and that of all ground iff-formulas, respectively, on $\langle \Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F} \rangle$. Let $\textsc{GIFF}(\mathcal{G})$ denote the set of all ground iff-formulas that consist only of atoms in $\mathcal{G}$. ■

If I is an iff-formula $a \leftrightarrow (conj_1 \vee conj_2 \vee \cdots \vee conj_n)$ on $\langle \Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F} \rangle$, then:

- For any $\theta \in \mathcal{S} \cup \mathcal{S}_{\rm F}$ and any $i \in \{1, \ldots, n\}$, if $conj_i = \{b_1, b_2, \ldots, b_k\}$, then (i) θ is applicable to $conj_i$ iff it is applicable to each of $b_1, b_2, \ldots, b_k$, and (ii) when applicable, the result of applying θ to $conj_i$, denoted by $conj_i\theta$, is defined by $conj_i\theta = \{b_1\theta, b_2\theta, \ldots, b_k\theta\}$.
- For any $\theta \in \mathcal{S} \cup \mathcal{S}_{\rm F}$, (i) θ is applicable to I iff it is applicable to a and each of $conj_1, conj_2, \ldots, conj_n$, and (ii) when applicable, the result of applying θ to I, denoted by $I\theta$, is defined by $I\theta = (a\theta \leftrightarrow (conj_1\theta \vee conj_2\theta \vee \cdots \vee conj_n\theta)$.

C. Declarative Descriptions

A declarative description consists of clauses and iff-formulas. They may have labels inside referential constraints, by which we can have richer expressive power. For example, the constraint in the right-hand side of the iff-formula

$$l_3 : (odd *x) \leftrightarrow \{\langle \phi_{not}, (even *x), l_3 \rangle, (int *x)\}$$

means that $*x$ is an odd number iff $*x$ is an integer that is not even.

Definition 7: Let $\mathbb{L}$ be a set of labels. A *declarative description* D on $\langle \Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F} \rangle$ is a triple $\langle l, L, f \rangle$, where

1) $l \in L$,
2) $L \subseteq \mathbb{L}$, and
3) $f : L \rightarrow pow(\textsc{CL}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F}) \cup \textsc{IFF}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F}))$.

D is called a *referential declarative description* if there exists a referential clause in the set $\bigcup\{f(l) \mid l \in L\}$. It is called a *simple declarative description* otherwise. The set of all declarative descriptions on $\langle \Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F} \rangle$ is denoted by $\textsc{DD}(\Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F})$. ■

 ISBN: 978-1-61208-354-4

D. Evaluation of Constraints

Let $D = \langle l_0, L, f\rangle$ be a declarative description on $\langle \Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F}\rangle$ and $l_0, l_1, \ldots, l_k$ are all the labels in L, listed according to some predetermined order. Mappings

- $val : \mathrm{GCON}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F}) \times pow(\mathcal{G})^{k+1} \to Bool$,
- $\mathrm{TCON} : pow(\mathcal{G})^{k+1} \to pow(\mathrm{GCON}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F}))$

are introduced for evaluation of constraints and determination of the sets of all true constraints with respect to ground atom sets associated with $l_0, l_1, \ldots, l_k$. They are defined as follows:

1) For any constraint

$$c = \langle \phi, d_1, d_2, \ldots, d_m\rangle \in \mathrm{GCON}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})$$

and any $G_0, G_1, \ldots, G_k \subseteq \mathcal{G}$,

$$val(c, G_0, G_1, \ldots, G_k) = \phi(g_1, g_2, \ldots, g_m),$$

where for each $i \in \{0, 1, \ldots, m\}$,

$$g_i = \begin{cases} d_i & \text{if } d_i \in \mathcal{G} \cup \mathcal{G}_{\rm c}, \\ G_j & \text{if } d_i = l_j \text{ for some } j \in \{0, 1, \ldots, k\}. \end{cases}$$

2) For any $G_0, G_1, \ldots, G_k \subseteq \mathcal{G}$, $\mathrm{TCON}(G_0, G_1, \ldots, G_k)$ is the set

$$\{c \in \mathrm{GCON}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F}) \mid val(c, G_0, G_1, \ldots, G_k) = true\}.$$

E. Evaluation of Clauses and Iff-Formulas

Using the mapping TCON, ground clause sets and ground iff-formula sets, respectively, are associated with the labels of D by the mappings

- $gclSet : L \times pow(\mathcal{G})^{k+1} \to pow(\mathrm{GCL}(\mathcal{G}))$,
- $giffSet : L \times pow(\mathcal{G})^{k+1} \to pow(\mathrm{GIFF}(\mathcal{G}))$,

which are defined as follows:

1) For any $j \in \{0, 1, \ldots, k\}$ and any $G_0, G_1, \ldots, G_k \subseteq \mathcal{G}$, $gclSet(l_j, G_0, G_1, \ldots, G_k)$ is the set

$$\{C' \in \mathrm{GCL}(\mathcal{G}) \mid (\exists C \in f(l_j) \cap \mathrm{CL}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})) (\exists \theta \in \mathcal{S})(\exists \sigma \in \mathcal{S}_{\rm F}) : (C\theta\sigma \in \mathrm{GCL}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})) \;\&\; (con(C\theta\sigma) \subseteq \mathrm{TCON}(G_0, G_1, \ldots, G_k)) \;\&\; (lhs(C') = lhs(C\theta\sigma)) \;\&\; (rhs(C') = rhs(C\theta\sigma) - con(C\theta\sigma))\}.$$

2) For any $j \in \{0, 1, \ldots, k\}$ and any $G_0, G_1, \ldots, G_k \subseteq \mathcal{G}$, $giffSet(l_j, G_0, G_1, \ldots, G_k)$ is the set

$$\{I' \in \mathrm{GIFF}(\mathcal{G}) \mid (\exists I \in f(l_j) \cap \mathrm{IFF}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})) (\exists \theta \in \mathcal{S})(\exists \sigma \in \mathcal{S}_{\rm F}) : (I\theta\sigma \in \mathrm{GIFF}(\Gamma, \Gamma_{\rm c}, L, \Gamma_{\rm F})) \;\&\; (con(I\theta\sigma) \subseteq \mathrm{TCON}(G_0, G_1, \ldots, G_k)) \;\&\; (head(I') = head(I\theta\sigma)) \;\&\; (\#Conj(I') = \#Conj(I)) \;\&\; (\forall i \in \{1, 2, \ldots, \#Conj(I)\} : rhs_i(I') = rhs_i(I\theta\sigma) - con(I\theta\sigma))\}.$$

F. Models of Declarative Descriptions

Truth values of ground clauses in $\mathrm{GCL}(\mathcal{G})$ and ground iff-formulas in $\mathrm{GIFF}(\mathcal{G})$ are defined below. Let G be an arbitrary subset of $\mathcal{G}$. Then:

- A ground clause $C \in \mathrm{GCL}(\mathcal{G})$ is true with respect to G iff $lhs(C) \cap G \neq \emptyset$ or $rhs(C) \not\subseteq G$.
- A ground iff-formula $I \in \mathrm{GIFF}(\mathcal{G})$ is true with respect to G iff one of the following conditions is satisfied:
  1) $head(I) \in G$ and for some $i \in \{1, 2, \ldots, \#Conj(I)\}$, $rhs_i(I) \subseteq G$.
  2) $head(I) \notin G$ and for any $i \in \{1, 2, \ldots, \#Conj(I)\}$, $rhs_i(I) \not\subseteq G$.

Given these definitions, a mapping

$$Models : pow(\mathrm{GCL}(\mathcal{G}) \cup \mathrm{GIFF}(\mathcal{G})) \to pow(pow(\mathcal{G}))$$

is then defined by: For any $K \subseteq \mathrm{GCL}(\mathcal{G}) \cup \mathrm{GIFF}(\mathcal{G})$ and any $G \subseteq pow(\mathcal{G})$, $G \in Models(K)$ iff for any $k \in K$, k is true with respect to G.

Now let $D = \langle l_0, L, f\rangle$ be a declarative description on $\langle \Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F}\rangle$ and $l_0, l_1, \ldots, l_k$ are all the labels in L, listed according to some predetermined order. For any $j \in \{0, 1, \ldots, k\}$, a mapping

$$model_j : pow(\mathcal{G})^{k+1} \to pow(pow(\mathcal{G}))$$

is defined as follows: For any $G_0, G_1, \ldots, G_k \subseteq \mathcal{G}$,

$$model_j(G_0, G_1, \ldots, G_k) = Models(K_j),$$

where K_j is the union of the following two sets:

- $gclSet(l_j, G_0, G_1, \ldots, G_k) \subseteq \mathrm{GCL}(\mathcal{G})$
- $giffSet(l_j, G_0, G_1, \ldots, G_k) \subseteq \mathrm{GIFF}(\mathcal{G})$

Using the mappings $model_0, model_1, \ldots, model_k$, the declarative description D determines a collection of the following membership constraints:

$$\begin{aligned} x_0 &\in model_0(x_0, x_1, \ldots, x_k) \\ x_1 &\in model_1(x_0, x_1, \ldots, x_k) \\ &\cdots \\ x_k &\in model_k(x_0, x_1, \ldots, x_k) \end{aligned}$$

This collection is called the *system of membership constraints* for D, denoted by $\mathrm{SMC}(D)$.

Definition 8: $[G, G_1, G_2, \ldots, G_k]$ is an extended model of D iff the set of equations $\{x_0 = G, x_1 = G_1, x_2 = G_2, \ldots, x_k = G_k\}$ satisfies $\mathrm{SMC}(D)$. ■

Definition 9: G is a model of D iff there exist $G_1, G_2, \ldots, G_k$ such that $[G, G_1, G_2, \ldots, G_k]$ is an extended model of D. ■

 ISBN: 978-1-61208-354-4

G. A Logical Structure for Declarative Descriptions

Using the class of declarative descriptions proposed so far, a logical structure $\mathcal{L} = \langle \mathcal{K}, \mathcal{I}, \nu \rangle$ for declarative descriptions on $\langle \Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F} \rangle$ is defined as follows:

1) $\mathcal{K} = {\rm DD}(\Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F})$.
2) $\mathcal{I} = \mathit{pow}(\mathcal{G})$.
3) $\nu : \mathcal{K} \to \mathit{Map}(\mathcal{I}, \mathit{Bool})$ is defined as follows: Let $L \subseteq \mathbb{L}$. Let $D = \langle l_0, L, f \rangle$ be a declarative description on $\langle \Gamma, \Gamma_{\rm c}, \mathbb{L}, \Gamma_{\rm F} \rangle$ and $l_0, l_1, \ldots, l_k$ are all the labels in L, where $l_1, \ldots, l_k$ are listed according to some predetermined order of labels in $\mathbb{L}$. Then for any $G \subseteq \mathcal{G}$, $\nu(D)(G) = \mathit{true}$ iff there exist $G_1, \ldots, G_k \subseteq \mathcal{G}$ such that $(x_0 = G; x_1 = G_1; \ldots; x_k = G_k)$ satisfies ${\rm SMC}(D)$.

V. Constraints in the S-Expression Space

Referring to the specialization system $\Gamma_{\rm S} = \langle \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu \rangle$ given in Section III-B, we define important constraints in the specific domain of S-expressions. We also show that *func*-atoms can be realized in the same class of constraints. In the rest of this paper,

- we assume that $\Gamma = \Gamma_{\rm c} = \Gamma_{\rm S}$, and we consider ${\rm DD}(\Gamma_{\rm S}, \Gamma_{\rm S}, \mathbb{L}, \Gamma_{\rm F})$;
- let $H = \mathcal{G}_{\rm c} \cup \mathcal{G} \cup \mathit{pow}(\mathcal{G}) \cup \mathcal{G}_{\rm F}$.

A. Reference to a Database

The iff-formula in Fig. 1 represents an employee table consisting of four columns, i.e., name, age, salary, and department, where *eq*-atoms are defined by the set of iff-formulas

$$\{((\mathit{eq}\ \bar{a}\ \bar{b}) \leftrightarrow (\mathit{true})) \mid (\bar{a} \text{ and } \bar{b} \text{ are variable-free S-expressions in } \mathcal{G})\ \&\ (\bar{a} = \bar{b})\}.$$

Reference from a world to a table in another world is realized by the use of a referential constraint. For example, for referring to the 4th argument (i.e., the department column) of employee table above, the constraint $\langle \phi_{\rm refer}, l_0, \mathit{emp}, 4, *a4 \rangle$ can be used by assuming a mapping $\phi_{\rm refer} : H^4 \to \mathit{Bool}$ given by: $\phi_{\rm refer}(G, t_1, t_2, t_3) = \mathit{true}$ iff $G \subseteq \mathcal{G}$, $t_1, t_2, t_3 \in \mathcal{G}_{\rm c}$, and $(\mathit{refer}\ t_1\ t_2\ t_3) \in G$, where *refer* is defined by the clause

$$l_0{:}\ \ (\mathit{refer}\ \mathit{emp}\ 4\ *d) \leftarrow (\mathit{emp}\ *n\ *a\ *s\ *d).$$

Several kinds of constraints can be constructed for formulating aggregate operations concerning a referenced database,

$$\begin{aligned} l_0{:}\ \ & (\mathit{emp}\ *n\ *a\ *s\ *d) \leftrightarrow \\ & (\{(\mathit{eq}\ *n\ \mathit{John}), (\mathit{eq}\ *a\ 34), (\mathit{eq}\ *s\ 8600), (\mathit{eq}\ *d\ d1)\} \vee \\ & \{(\mathit{eq}\ *n\ \mathit{Morgan}), (\mathit{eq}\ *a\ 24), (\mathit{eq}\ *s\ 6300), (\mathit{eq}\ *d\ d2)\} \vee \\ & \{(\mathit{eq}\ *n\ \mathit{Lewis}), (\mathit{eq}\ *a\ 42), (\mathit{eq}\ *s\ 9000), (\mathit{eq}\ *d\ d3)\} \vee \\ & \{(\mathit{eq}\ *n\ \mathit{Long}), (\mathit{eq}\ *a\ 34), (\mathit{eq}\ *s\ 8500), (\mathit{eq}\ *d\ d2)\} \vee \\ & \{(\mathit{eq}\ *n\ \mathit{Henry}), (\mathit{eq}\ *a\ 29), (\mathit{eq}\ *s\ 12300), (\mathit{eq}\ *d\ d1)\} \vee \\ & \{(\mathit{eq}\ *n\ \mathit{Thomas}), (\mathit{eq}\ *a\ 31), (\mathit{eq}\ *s\ 7300), (\mathit{eq}\ *d\ d3)\} \vee \\ & \{(\mathit{eq}\ *n\ \mathit{Martin}), (\mathit{eq}\ *a\ 45), (\mathit{eq}\ *s\ 7500), (\mathit{eq}\ *d\ d1)\}) \end{aligned}$$

Figure 1. Representing an employee table

including operations that are commonly supported by a standard SQL, e.g., max, min, average, and sum [8][9][10], and those that are not, e.g., argmax and argmin. Formulation of the sum, argmax, and argmin operations using constraints is illustrated below.

B. Summation

Summation can be formulated using a constraint defined by the mapping

$$\phi_{\mathit{sum}} : H^3 \to \mathit{Bool},$$

where $\phi_{\mathit{sum}}(id, G, t) = \mathit{true}$ iff t is the sum of the elements of the multi-set $\{s \mid (\mathit{sum}\ id\ n\ s) \in G\}$. Using ϕ_{sum}, the query "find the sum of salaries of all the employees who work in the department $d1$ or the department $d2$" is represented by:

$$\begin{aligned} l_0{:}\ \ & (\mathit{ans}\ *x) \leftarrow \langle \phi_{\mathit{sum}}, id, l_0, *x \rangle. \\ l_0{:}\ \ & (\mathit{sum}\ id\ *n\ *s) \leftarrow (\mathit{emp}\ *n\ *a\ *s\ d1). \\ l_0{:}\ \ & (\mathit{sum}\ id\ *n\ *s) \leftarrow (\mathit{emp}\ *n\ *a\ *s\ d2). \end{aligned}$$

C. Argmax and Argmin

Argmax stands for the argument of the maximum. Given a mapping f, argmax gives a value x that maximizes $f(x)$. The argmax constraint is defined using

$$\phi_{\mathit{argmax}} : H^3 \to \mathit{Bool},$$

given by: $\phi_{\mathit{argmax}}(id, G, t) = \mathit{true}$ iff $id \in \mathcal{G}_{\rm c}$, $G \subseteq \mathcal{G}$, $t \in \mathcal{G}_{\rm c}$, and there exists $t' \in \mathcal{G}_{\rm c}$ such that

1) $(\mathit{argmax}\ id\ t\ t') \in G$, and
2) for any $t_1, t_2 \in \mathcal{G}_{\rm c}$, if $(\mathit{argmax}\ id\ t_1\ t_2) \in G$, then $t_2 \le t'$.

For instance, the query "find all persons who earn the maximum salary" is represented by:

$$\begin{aligned} l_0{:}\ \ & (\mathit{ans}\ *x) \leftarrow \langle \phi_{\mathit{argmax}}, id, l_0, *x \rangle. \\ l_0{:}\ \ & (\mathit{argmax}\ id\ *n\ *s) \leftarrow (\mathit{emp}\ *n\ *a\ *s\ *d). \end{aligned}$$

Similarly, given a mapping f, argmin gives a value x that minimizes $f(x)$. The argmin constraint is defined using

$$\phi_{\mathit{argmin}} : H^3 \to \mathit{Bool},$$

given by: $\phi_{\mathit{argmin}}(id, G, t) = \mathit{true}$ iff $id \in \mathcal{G}_{\rm c}$, $G \subseteq \mathcal{G}$, $t \in \mathcal{G}_{\rm c}$, and there exists $t' \in \mathcal{G}_{\rm c}$ such that

1) $(\mathit{argmin}\ id\ t\ t') \in G$, and
2) for any $t_1, t_2 \in \mathcal{G}_{\rm c}$, if $(\mathit{argmin}\ id\ t_1\ t_2) \in G$, then $t_2 \ge t'$.

The query "find all the youngest persons who work at the department $d1$," for example, is represented by:

$$\begin{aligned} l_0{:}\ \ & (\mathit{ans}\ *x) \leftarrow \langle \phi_{\mathit{argmin}}, id, l_0, *x \rangle. \\ l_0{:}\ \ & (\mathit{argmin}\ id\ *n\ *a) \leftarrow (\mathit{emp}\ *n\ *a\ *s\ d1). \end{aligned}$$

 ISBN: 978-1-61208-354-4

D. func-atoms

Atoms of a special kind, called *func*-atoms, are used for meaning-preserving Skolemization [3]. They can be represented by constraints with function variables and function constants as arguments. A mapping

$$\phi_{\mathit{func}} : \mathbb{FC} \times \mathcal{G}_{\mathrm{c}}^{n+1} \rightarrow \mathit{Bool}$$

is defined by: $\phi_{\mathit{func}}(f, t_1, t_2, \ldots, t_n, t_{n+1}) = \mathit{true}$ iff f is an n-ary function constant and $t_1, t_2, \ldots, t_n, t_{n+1}$ are variable-free terms such that $f(t_1, t_2, \ldots, t_n) = t_{n+1}$.

Let a first-order formula F be given by:

$$F\colon\ \exists x\colon (\mathit{hasChild}(\mathit{Peter}, x) \wedge (\exists y\colon \mathit{motherOf}(x, y)))$$

By meaning-preserving Skolemization [3], F is converted into $\{C_1, C_2\}$, given as follows:

$$C_1\colon\ (\mathit{hasChild\ Peter} *x) \leftarrow \langle \phi_{\mathit{func}}, *h_1, *x\rangle$$
$$C_2\colon\ (\mathit{motherOf} *x\ *y) \leftarrow \langle \phi_{\mathit{func}}, *h_1, *x\rangle, \langle \phi_{\mathit{func}}, *h_2, *y\rangle$$

E. Negation

Negation [8] can also be represented as constraints using

$$\phi_{\mathit{not}} : \mathcal{G} \times \mathit{pow}(\mathcal{G}) \rightarrow \mathit{Bool},$$

given by: for any $g \in \mathcal{G}$ and any $G \subseteq \mathcal{G}$, $\phi_{\mathit{not}}(g, G) = \mathit{true}$ iff $g \notin G$.

Example 1: Assume that $L = \{l_0\}$ and a is an arbitrary atom. Let definite clauses C_1 and C_2 be given by:

$$C_1 :\quad a \leftarrow (\mathit{not}\ a)$$
$$C_2 :\quad (\mathit{not}\ a) \leftarrow \langle \phi_{\mathit{not}}, a, l\rangle$$

Let D be a declarative description $\langle l_0, L, f\rangle$, where $f(l_0) = \{C_1, C_2\}$. Let $G = \mathit{rep}(a) \cup \mathit{rep}(\mathit{not}(a))$. We show that G is a model of D as follows: Let $C_1' = (g \leftarrow (\mathit{not}\ g))$ be any ground instance of C_1. C_1' is true since $g \in G$. So C_1 is true with respect to G. Let $C_2' = ((\mathit{not}\ g) \leftarrow \langle \phi_{\mathit{not}}, g, l_0\rangle)$ be any ground instance of C_2. C_2' is true since $(\mathit{not}\ g) \in G$. So C_2 is true with respect to G. ■

Example 2: Assume again that $L = \{l_0\}$ and a is an arbitrary atom. Let I_1 and I_2 be the following iff-formulas:

$$I_1 :\quad a \leftrightarrow \{(\mathit{not}\ a)\}$$
$$I_2 :\quad (\mathit{not}\ a) \leftrightarrow \{\langle \phi_{\mathit{not}}, a, l\rangle\}$$

Let D be a declarative description $\langle l_0, L, f\rangle$, where $f(l_0) = \{I_1, I_2\}$. We show that D is not satisfiable as follows: Assume that G is a model of D. Let $\theta \in \mathcal{S}$ such that $a\theta \in G$ and let $a\theta = g$. From $I_1\theta = (g \leftrightarrow \{(\mathit{not}\ g)\})$, we have: $g \in G$ iff $(\mathit{not}\ g) \in G$. From $I_2\theta = ((\mathit{not}\ g) \leftrightarrow \{\langle \phi_{\mathit{not}}, g, l_0\rangle\})$, we have: $(\mathit{not}\ g) \in G$ iff $\phi_{\mathit{not}}(g, l_0) = \mathit{true}$. From the definition of ϕ_{not}, $\phi_{\mathit{not}}(g, l_0) = \mathit{true}$ iff $g \notin G$. Then $g \in G$ iff $g \notin G$, which is a contradiction. ■

VI. Conclusions

We have defined a method for multiple-world extension of a clausal logical structure, and by applying this extension, we have introduced a multiple-world clausal logical structure for S-expression atoms. We can consider multiple worlds simultaneously. Not only usual terms but also models themselves can be considered as arguments of a relation. This class of declarative descriptions can be used for representing open worlds, closed worlds, and their mixtures, by using clauses and iff-formulas. This gives one solution for unifying open and closed world. Function variables in constraints enables us to use *func*-atoms, which are essential for meaning-preserving Skolemization [3].

Acknowledgment

This research was partially supported by JSPS KAKENHI Grant Number 25280078, and the Collaborative Research Program 2013, Information Initiative Center, Hokkaido University.

References

[1] K. Akama and E. Nantajeewarawat, "Logical Structures on Specialization Systems: Formalization and Satisfiability-Preserving Transformation," in Proceedings of the 7th International Conference on Intelligent Technologies, Taipei, Taiwan, 2006, pp. 100–109.

[2] ——, "Construction of Logical Structures on Specialization Systems," in Proceedings of the 2011 World Congress on Information and Communication Technologies, Mumbai, India, 2011, pp. 1030–1035.

[3] ——, "Meaning-Preserving Skolemization," in Proceedings of the 2011 International Conference on Knowledge Engineering and Ontology Development, Paris, France, 2011, pp. 322–327.

[4] M. Fitting, First-Order Logic and Automated Theorem Proving, 2nd ed. Springer-Verlag, 1996.

[5] J. W. Lloyd, Foundations of Logic Programming, second, extended ed. Springer-Verlag, 1987.

[6] C.-L. Chang and R. C.-T. Lee, Symbolic Logic and Mechanical Theorem Proving. Academic Press, 1973.

[7] K. Akama, "Declarative Semantics of Logic Programs on Parameterized Representation Systems," Advances in Software Science and Technology, vol. 5, 1993, pp. 45–63.

[8] S. Abiteboul, R. Hull, and V. Vianu, Foundations of Databases. Addison-Wesley, 1995.

[9] M. Dahr, Deductive Databases: Theory and Applications. Coriolis Group, 1996.

[10] C. J. Date, SQL and Relational Theory, 2nd Edition. O'Reilly Media, 2011.

 ISBN: 978-1-61208-354-4