プログラミング－言語・基礎・実践－　7－7
（1992. 5. 29）

Minimal Model Semantics of Generalized Logic Programs

Kiyoshi Akama

Faculty of Engineering, Hokkaido University

GLP theory is an **axiomatic theory** of generalized logic programs which are the sets of definite program clauses consisting of generalized atoms. In this paper we propose a definition of **interpretations** for GLP theory. An interpretation is defined to be a subset of the interpretation domain (which corresponds to the Herbrand base in the usual theory). This definition does not refer to inner structures of atoms such as predicates, terms, functions and variables. This enables us to develop a very general, elegant and powerful theory of logic programming. In this paper we also define **models** and **logical consequences**, and develop the **minimal model semantics** of generalized logic programs.

一般化論理プログラムの最小モデル意味論

赤間 清

北海道大学　工学部　情報工学科

GLP の理論は、一般化論理プログラムの公理的理論である。一般化論理プログラムとは、縮小系という公理によって規定された一般化アトムからなる確定節の集合である。本論文では、一般化論理プログラムの理論にふさわしい解釈 の定義を提案している。解釈は、解釈領域（通常の理論では Herbrand base にあたる集合）の部分集合として定義される。この定義は、述語、項、関数、変数などの原子論理式の内部構造に言及しないもので、一般的でエレガントで強力な論理プログラムの理論を築くことを可能にする。本論文では、また、解釈の定義を基礎にモデルと論理的帰結を定義し、一般化論理プログラムの最小モデル意味論を構築する。

1 Introduction

1.1 Specialization Systems

Instead of starting with usual concrete definitions of atoms and substitutions, we adopt abstract definitions of them and are constructing an **axiomatic theory** of logic programming. We call it **GLP theory** (theory of **generalized logic programs**).

We defined the structure called **specialization systems** in terms of very simple axioms as follows:

Definition 1 A *specialization system* is a 4-tuple $< \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu >$ that satisfies the following conditions.

1. $\mu : \mathcal{S} \rightarrow partial\_map(\mathcal{A})$
2. $\forall s_1, s_2 \in \mathcal{S}, \exists s \in \mathcal{S} : \mu(s) = \mu(s_2) \circ \mu(s_1)$
3. $\exists s \in \mathcal{S}, \forall a \in \mathcal{A} : \mu(s)(a) = a$
4. $\mathcal{G} \subset \mathcal{A}$

Elements of $\mathcal{A}$ are called atoms. $\mathcal{G}$ is called the interpretation domain. Elements of $\mathcal{S}$ are called *specializations*. The specializations that satisfy the third condition are called *identity specializations*.

We defined generalized logic programs on specialization systems.

Definition 2 A *logic program on a specialization system* Γ is a (possibly infinite) set of definite program clauses on $\mathcal{A}$, that is a clause of the form : $H \leftarrow A_1, \ldots, A_n$, where $H, A_1, \ldots, A_n$ are atoms in $\mathcal{A}$.

GLP theory does not refer to predicates, variables, constants, functions and substitutions which are the basic components of ordinary logic programs.

1.2 Interpretations

The purpose of this paper is to give the definition of **interpretations**, **models** and **logical consequences** for generalized logic programs, and to develop the **minimal model semantics** of generalized logic programs.

The problem to be solved first in this paper is to introduce a general definition of interpretations for **c-formulas** on specialization systems. In the usual theory of logic, an interpretation is defined as follows.

Definition 3 An interpretation I of a first order language L consists of the following:

- A non-empty set D, called the domain of the interpretation.
- For each constant in L, the assignment of an element in D.
- For each function in L, the assignment of a mapping from D^n to D.
- For each predicate in L, the assignment of a relation on D^n.

This definition refers to constants, functions and predicates. Since we do not assume such structures in atoms on specialization systems, this definition can not be used to define interpretations in GLP theory.

1.3 Herbrand interpretation

We first review a Herbrand interpretation, a Herbrand base and a Herbrand model. We assume that L is a first order language.

Definition 4 The Herbrand universe U_L for L is the set of all ground terms, which can be formed out of the constants and functions in L. In the case that L has no constants, we add some constant to form ground terms.

Definition 5 A Herbrand interpretation for L is an interpretation that satisfies:

- The domain of the interpretation is the Herbrand universe U_L.
- Constants in L are assigned to themselves in U_L.
- If f is an n-ary function in L, then the mapping F from $(U_L)^n$ into U_L such that $F(t_1, ..., t_n) = f(t_1, ..., t_n)$ is assigned to f.

Definition 6 The Herbrand base B_L for L is the set of all ground atoms which can be formed by using predicates from L with ground terms from the Herbrand universe as arguments.

Definition 7 Let S be a set of closed formulas of L. A Herbrand model for S is a Herbrand interpretation for L which is a model for S.

In Herbrand interpretations, the assignment to constants and functions is fixed. So we can regard a Herbrand interpretation as a subset of the Herbrand base. For any Herbrand interpretation, the corresponding subset of the Herbrand base is the set of all ground atoms which are true with respect to the interpretation. Conversely, given an arbitrary subset of the Herbrand base, there is a corresponding Herbrand interpretation.

We also know the following proposition.

Proposition 1 If S is a set of clauses, then,
S is unsatisfiable $\leftrightarrow$ S has no Herbrand models.

Based on these observations, we will define interpretations as the subset of the interpretation domain of specialization systems.

2 Interpretations

In the following discussion we fix a specialization system $\Gamma =< \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu >$.

In order to discuss the meanings for c-formulas on $< \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu >$, we define interpretations, which determine the truth or falsity of all c-formulas.

Definition 8 An *interpretation* I on $\mathcal{G}$ is defined as a subset of interpretation domain $\mathcal{G}$. An interpretation I is often regarded as a mapping from $\mathcal{G}$ to $V = \{true, false\}$, that is,$I(g) = true \Leftrightarrow g \in I$.

Interpretations determine the truth or falsity of any c-formula on $\mathcal{A}$ in the following way.

Definition 9 Let V be $\{true, false\}$. Negation $\neg$ is associated with a mapping $\neg : V \to V$. Conjunction $\wedge$ and disjunction $\vee$ are associated with mappings $\wedge : V \times V \to V$ and $\vee : V \times V \to V$, respectively. We use $\neg$, $\wedge$ and $\vee$ not only as logical connectives, but also as the associated mappings. The three associated mappings are defined by the following table.

x	y	$\neg x$	$x \wedge y$	$x \vee y$
$true$	$true$	$false$	$true$	$true$
$true$	$false$	$false$	$false$	$true$
$false$	$true$	$true$	$false$	$true$
$false$	$false$	$true$	$false$	$false$

The two associated mappings $\wedge$ and $\vee$ are extended to the operation of any number (possibly infinite) of inputs which are given by a partial mapping to V.

Definition 10 Let S be any set and v be any partial mapping from S to $V = \{$ true, false $\}$. We define $[\wedge v(s) \mid s \in S]$ is true iff $v(s)$ is true for all $s \in S$ such that $v(s)$ is defined and $[\vee v(s) \mid s \in S]$ is true iff $v(s)$ is true for some $s \in S$ such that $v(s)$ is defined.

We begin with a mapping which determines the truth or falsity of p-formulas on $\mathcal{G}$.

Definition 11 An interpretation I on $\mathcal{G}$ determines the mapping val_I from $PF(\mathcal{G})$ to $V = \{true, false\}$ by the following formulas, where $g \in \mathcal{G}$ and $x, y \in PF(\mathcal{G})$.

1. $val_I(g) = I(g)$
2. $val_I(\neg x) = \neg val_I(x)$
3. $val_I(x \wedge y) = val_I(x) \wedge val_I(y)$
4. $val_I(x \vee y) = val_I(x) \vee val_I(y)$

Definition 12 For any element s in $\mathcal{S}$, the partial mapping $inst_s$ from $PF(\mathcal{A})$ to $PF(\mathcal{G})$ is defined as follows, where $a \in \mathcal{A}$ and $x, y \in PF(\mathcal{A})$.

1. $inst_s(a) = \mu(s)(a) \cdots \mu(s)(a)$ is defined and $\mu(s)(a) \in \mathcal{G}$
2. $inst_s(a) =$ undefined $\cdots$ $\mu(s)(a)$ is undefined or $\mu(s)(a) \notin \mathcal{G}$
3. $inst_s(\neg x) = \neg inst_s(x)$
4. $inst_s(x \wedge y) = inst_s(x) \wedge inst_s(y)$
5. $inst_s(x \vee y) = inst_s(x) \vee inst_s(y)$

$inst_s(x)$ is defined iff, for any atom a which appears in $x \in PF(\mathcal{A})$, $\mu(s)(a)$ is defined and is an element in $\mathcal{G}$.

The following mapping determines the truth or falsity of each c-formula on $\mathcal{A}$.

Definition 13 An interpretation I on $\mathcal{G}$ determines the mapping $value_I$ from $CF(\mathcal{A})$ to $V = \{true, false\}$ by the following formulas, where $z \in PF(\mathcal{A})$ and $x, y \in CF(\mathcal{A})$.

1. $value_I(\forall z) = [\wedge val_I(inst_s(z)) \mid s \in \mathcal{S}]$
2. $value_I(\exists z) = [\vee val_I(inst_s(z)) \mid s \in \mathcal{S}]$
3. $value_I(\neg x) = \neg value_I(x)$
4. $value_I(x \wedge y) = value_I(x) \wedge value_I(y)$
5. $value_I(x \vee y) = value_I(x) \vee value_I(y)$

In the definition of the usual theory universal and existential quantifiers bind all variables in universal and existential closures. On the other hand in the definition of c-formulas in GLP theory we do not use the concept of variables. Can we give well-defined semantics to universal or existential closures without referring the concept of variables? Definition 13 answers yes to this question because universal (existential) closures are equivalent in the usual theory to the conjunction (disjunction) of all ground instances of them and that ground instances of them can be generated by specializations in GLP theory.

We give one example using $\Gamma_5 =< \mathcal{A}_5, \mathcal{G}_5, \mathcal{S}_5, \mu_5 >$. It is defined as follows [1].

1. $\mathcal{A}_5 = \{x, y, z, p, q, r, s, t\}$
2. $\mathcal{G}_5 = \{p, q, r, s, t\}$
3. $\mathcal{S}_5 = \{e, a, b, c, o\}$
4. μ_5 is a mapping from $\mathcal{S}_5$ to $partial\_map(\mathcal{A}_5)$ which is defined by:

$$\mu_5(e) =< \mathcal{A}_5, \mathcal{A}_5, \{(x,x),(y,y),(z,z),(p,p),(q,q),(r,r),(s,s),(t,t)\} >$$
$$\mu_5(a) =< \mathcal{A}_5, \mathcal{A}_5, \{(x,p),(y,q),(z,r)\} >$$
$$\mu_5(b) =< \mathcal{A}_5, \mathcal{A}_5, \{(z,p)\} >$$
$$\mu_5(c) =< \mathcal{A}_5, \mathcal{A}_5, \{(x,q),(y,s)\} >$$
$$\mu_5(o) =< \mathcal{A}_5, \mathcal{A}_5, \{\} >$$

Let z be a p-formula $y \vee (\neg x)$ on Γ_5. From the definition $\forall z$ is a c-formula. Now consider $value_I(\forall z)$ for an interpretation $I = \{p, q\}$. There are five specializations(e,a,b,c and o) in $\mathcal{S}_5$.

1. $inst_e(z)$ is undefined because $\mu_5(e)(x) = x$ and $\mu_5(e)(y) = y$ are not in $\mathcal{G}_5$.
2. $inst_a(z) = (q \vee (\neg p))$
3. $inst_b(z)$ is undefined because $\mu_5(b)(x)$ and $\mu_5(b)(y)$ are undefined.
4. $inst_c(z) = (s \vee (\neg q))$
5. $inst_o(z)$ is undefined because $\mu_5(o)(x)$ and $\mu_5(o)(y)$ are undefined.

Then from definition 13,

$$value_I(\forall z) = val_I(q \vee (\neg p)) \wedge val_I(s \vee (\neg q))$$

When $I = \{p, q\}$, we know $val_I(q) = true$, $val_I(p) = true$, and $val_I(\neg p) = false$. Therefore $val_I(q \vee (\neg p)) = true$. Similarly, as $val_I(s) = false$, $val_I(q) = true$, and $val_I(\neg q) = false$, then $val_I(s \vee (\neg q)) = false$. Therefore,

$$value_I(\forall z) = false$$

[1] It was used also in [2].

3 Models and Logical Consequences

The concepts of models and logical consequences are also defined by using the mapping $value_I$ in the same way as in the usual theory of logic programs.

Definition 14 An interpretation I on $\mathcal{G}$ is a *model* of a subset E of $CF(\mathcal{A})$ iff $value_I(e) = true$ for any element e in E. The set of all models of E is denoted by $Model(E)$. An interpretation I on $\mathcal{G}$ is a model of an element e of $CF(\mathcal{A})$ iff I is a model of $\{e\}$. The set of all models of e is denoted by $Model(e)$.

Assume that $P = \{y \leftarrow x, z \leftarrow\}$ is a program on the specialization system Γ_5. Then all models of P are $\{p, q, r, s\}$ and $\{p, q, r, s, t\}$.

Proposition 2 The interpretation I on $\mathcal{G}$ is a model of a subset E of $CF(\mathcal{A})$ iff I is a model of all elements in E.

Proof
$$\begin{aligned} I \in Model(E) &\Leftrightarrow \forall e \in E : value_I(e) = true \\ &\Leftrightarrow \forall e \in E : I \in Model(\{e\}) \\ &\Leftrightarrow \forall e \in E : I \in Model(e) \end{aligned}$$

Definition 15 Let E_1 and E_2 be subsets of $CF(\mathcal{A})$. E_2 is a logical consequence of E_1 ($E_1 \models E_2$) iff, for any interpretation I on $\mathcal{G}$, if I is a model of E_1 then I is a model of E_2. A c-formula $e \in CF(\mathcal{A})$ is a logical consequence of E_1 ($E_1 \models e$) iff $E_1 \models \{e\}$. E_2 is a logical consequence of a c-formula $e \in CF(\mathcal{A})$ ($e \models E_2$) iff $\{e\} \models E_2$.

4 Minimal Model Semantics

We prove that there is a minimal model of any logic program on a specialization system. The meaning of a logic program on a specialization system should be the minimal model of it [1].

Proposition 3 (Model Intersection Property) If $\{M_j \mid j \in J\}$ is a non-empty set of models of a Horn clause h, then the interpretation $[\bigcap M_j \mid j \in J]$ is also a model of h.
Proof See appendix.

Proposition 4 If there is a model of a set of Horn clauses P, there is a minimal model M_P, and

$$M_P = [\bigcap M \mid M \in Model(P)].$$

Proof Let $\{M_j \mid j \in J\}$ be the set of all models of P. From the assumption this set is not empty. Let h be any element of P. From proposition 2 M_j is a model of h. As h is a Horn clause and $\{M_j \mid j \in J\}$ is a non-empty set of the models of h, then using proposition 3, the interpretation $M_P = [\bigcap M_j \mid j \in J]$ is also a model of h. Since h is any element of P, then using proposition 2 M_P is a model of P. Since M_P is a subset of any element of $\{M_j \mid j \in J\}$, and $\{M_j \mid j \in J\}$ is the set of all models of P, M_P is the minimal model of P and $M_P = [\bigcap M \mid M \in Model(P)]$.

Theorem 1 Every logic program P has a minimal model M_P, and

$$M_P = [\bigcap M \mid M \in Model(P)]$$

Proof Let c be any element of a logic program P. As c is a program clause, we can assume $c = \forall(H \vee B)$, where H is the head of c, B is $B_1 \vee B_2 \vee \ldots \vee B_n$ and $n \geq 0$. And let an interpretation M on $\mathcal{G}$ be defined by $M(g) = true$ for all $g \in \mathcal{G}$,

$$\begin{aligned} value_M(c) &= value_M(\forall(H \vee B)) \\ &= [\wedge val_M(inst_s(H \vee B)) \mid s \in \mathcal{S}] \\ &= [\wedge(val_M(inst_s(H)) \vee val_M(inst_s(H \vee B))) \mid s \in \mathcal{S}] \\ &= [\wedge(true \vee val_M(inst_s(H \vee B))) \mid s \in \mathcal{S}] \\ &= true. \end{aligned}$$

Then M is a model of c. As c is any element of P, M is a model of P. From these P is a set of Horn clauses which has a model. Using proposition 4, we conclude that there is the minimal model M_P of P, and that $M_P = [\bigcap M \mid M \in Model(P)]$.

Assume that $P = \{y \leftarrow x, z \leftarrow\}$ is a program on the specialization system Γ_5. The minimal model of P is $\{p, q, r, s\}$.

5 Concluding Remarks

In this paper, we have defined interpretations as a subset of the interpretation domain. Based on the definition of interpretations, we also defined models and logical consequences for GLP theory. We have developed the minimal model semantics of generalized logic programs. In this theory we do not use the notions of predicates, variables, terms and substitutions which are used in the usual logic programming theory. This theory is very general and can be applied to many declarative knowledge representation systems.

References

[1] Akama,K. : Declarative Semantics of Logic Programs on Parameterized Representation Systems, *Hokkaido University Information Engineering Technical Report*, HIER-LI-9001 (1990)

[2] Akama,K. : Generalized Logic Programs on Specialization Systems, *Preprints Work. Gr.for Programming, IPSJ* 6-4-PRG, pp.31-40 (1992)

[3] Apt,K.R. and van Emden,M,H. : *Contribution to The Theory of Logic Programming, J.ACM*, Vol.29, No.3 (1982)

[4] Lloyd,J.W. : Foundations of Logic Programming, 2nd edition, Springer-Verlag, p.212 (1987)

[5] van Emden,M.H. and Kowalsky,R.A. : The semantics of Predicate logic as a Programming Language, *J.ACM*, vol.23, No.4 pp.733-742 (1976)

Appendix

[Proof of Model Intersection Property]

Assume that $\{M_j \mid j \in J\}$ is a non-empty set of models of a Horn clause h. Let $h = (\forall L)$, $L = L_1 \vee L_2 \vee \cdots \vee L_n$ and $M = [\bigcap M_j \mid j \in J]$. From the definition of models,

$$\forall j \in J : value_{M_j}(h) = true.$$

We transform this formula.

$$\Leftrightarrow \forall j \in J : value_{M_j}(\forall L) = true$$
$$\Leftrightarrow \forall j \in J : [\wedge val_{M_j}(inst_s(L)) \mid s \in \mathcal{S}] = true$$

Let s be any element of $\mathcal{S}$ where $inst_s(L)$ is defined. Then,

$$\forall j \in J : val_{M_j}(inst_s(L)) = true.$$

As $L = L_1 \vee L_2 \vee \cdots \vee L_n$, this is equivalent to

$$\Leftrightarrow \forall j \in J, \exists k : val_{M_j}(inst_s(L_k)) = true.$$

For any j, let L_{k_j} be one of literals L_k which satisfies $val_{M_j}(inst_s(L_k)) = true$, then

$$(f) \forall j \in J : val_{M_j}(inst_s(L_{k_j})) = true$$

Here the following two cases are possible.

(1) For all j, L_{k_j} is a positive literal.

(2) For some j, L_{k_j} is a negative literal.

- Case (1) A Horn clause has at most one positive literal, then there is an atom L_i such that $L_{k_j} = L_i$ for any j. Letting $g = inst_s(L_i)$, (f) is transformed as follows.

$$\begin{aligned}
&\forall j \in J : [val_{M_j}(inst_s(L_{k_j})) = true] \\
\Leftrightarrow\ &\forall j \in J : [val_{M_j}(inst_s(L_i)) = true] \\
\Leftrightarrow\ &\forall j \in J : [val_{M_j}(g) = true] \\
\Leftrightarrow\ &\forall j \in J : [M_j \ni g] \\
\Leftrightarrow\ &M \ni g \\
\Leftrightarrow\ &val_M(g) = true \\
\Leftrightarrow\ &val_M(inst_s(L_i)) = true \\
\rightarrow\ &val_M(inst_s(L)) = true
\end{aligned}$$

- Case (2) Assume L_{k_j} is a negative literal $\neg a$, and let $inst_s(\neg a) = \neg g$. Then from (f),

$$val_{M_j}(inst_s(L_{k_j})) = true$$

This is transformed as follows.

$$\begin{aligned}
\Leftrightarrow\ &val_{M_j}(inst_s(\neg a)) = true \\
\Leftrightarrow\ &val_{M_j}(\neg g) = true \\
\Leftrightarrow\ &val_{M_j}(g) = false \\
\Leftrightarrow\ &M_j \not\ni g \\
\rightarrow\ &M \not\ni g \\
\Leftrightarrow\ &val_M(g) = false \\
\Leftrightarrow\ &val_M(\neg g) = true \\
\Leftrightarrow\ &val_M(inst_s(\neg a)) = true \\
\Leftrightarrow\ &val_M(inst_s(L_{k_j})) = true \\
\rightarrow\ &val_M(inst_s(L)) = true
\end{aligned}$$

In both cases (1) and (2), we have $val_M(inst_s(L)) = true$. This is proved for any $s \in \mathcal{S}$ where $inst_s(L)$ is defined, then

$$[\wedge val_M(inst_s(L)) \mid s \in \mathcal{S}] = true.$$

This is transformed as follows.

$$\begin{aligned}
\Leftrightarrow\ &value_M(\forall L) = true \\
\Leftrightarrow\ &value_M(h) = true \\
\Leftrightarrow\ &M \in Model(h) \\
\Leftrightarrow\ &[\bigcap M_j \mid j \in J] \in Model(h)
\end{aligned}$$