울산대학교 연구논문집 제18권(자연과학 · 공학편) 제 2 호 pp.115~118. 1987.
UOU Report Vol. 18. (Natural Science and Engineering) No. 2. pp. 115~118. 1987,
University of Ulsan.

비균일 정의역 위에서의 일차 논리 언어의미 부여에 관한 연구

배 재 학
전자계산학과
(1987. 4. 30 접수)

〈要　約〉

논리 프로그램에 대한 '의도적 의미 부여'가 일차 논리 언어를 대상으로 하는 의미 부여에 관한 정의에 부합하지 않을 때가 있다. 논리 프로그램에 관한 의미 부여와 일차 논리 언어에 대한 의미 부여 사이의 괴리를 메우기 위해, 본 논문에서는 '형을 가진' 일차 논리 언어를 제안함과 더불어 그것에 대한 형을 가진 의미부여도 제시하였다. 뿐만 아니라, 절의 집합 S를 만족시키는 형을 가진 허브란드식 의미부여 I *가 없다는 사실은 S를 만족시키기가 불가능 하다는 데 대한 필요충분조건임도 보였다.

A Study on Interpretations of a First Order Language over Heterogeneous Domain

Bae, Jae-Hak
Dept. of Computer Science
(Received April 30, 1987)

〈Abstract〉

Sometimes the "intended interpretation" of a logic program is not an interpretation of a first order language. To fill the gap between the interpretation of a logic program and the interpretation of a first order language, we propose a typed first order language and a typed interpretation of it. Together with them it is proved that a set of clauses S is unsatisfiable iff S has no typed Herbrand interpretation I * that satisfies S.

I . Introduction

Sometimes the intended interpretation(Loveland〔1〕, Lloyd〔2〕) for a logic program is not an interpretation of a first order language. The reason for this is as follows: When an n-ary total function f appears in a logic program the intended interpretation for f is f : D1x... xDn → Df, where each Di is a subset of the domain of discourse D. In view of the interpretation of a first order language the function f may not be a total function. The intended interpretation, therefore, is not generally an interpretation of a first order language.

A logic program(Lloyd[2]) is a finite subset of a first order language. The intended interpretation of it, however, is not an interpretation of a first order language.

Naturally, the intended interpretation should be an interpretation of a first order language. Considering this discrepancy, we propose a typed first order language and a typed interpretation of it. Together with them it is proved that a set of clauses S is unsatisfiable iff S has no typed Herbrand interpretation I * that satisfies S.

II. Typed First Order Languages

In this section we introduce a typed first order language. Then, a logic program is viewed as a finite subset of a typed first order language.

Definition : If all elements in a set S have a identical data type then S is homogeneous. If a set S is not homogeneous then S is heterogeneous.

Definition : A type is a nonempty, homogeneous, and countable set or D1x... xDn → Df where each Di is a nonempty, homogeneous, and countable set.

Definition : Let c be a constant, v be a variable, and f be a function.
"a constant c of type Dc" means c∈ Dc. "a variable v of type Dv" means that v denotes an element e such that e∈ Dv. "a term t of type Dt" means that t denotes an element e such that e∈ Dt. "a function f of type D1x... xDn → Df" means that the domain of f is D1x... xDn and the codomain of f is Df.

Definition : An alphabet A consists of seven classes of symbols:
(a) constants (b) variables (c) functions
(d) predicates (e) connectives (f) quantifiers
(g) punctuation symbols

Definition : A term of type Dt is defined inductively as follows:
(a) A variable of type Dt is a term of type Dt.
(b) A constant of type Dt is a term of type Dt.
(c) If f is an n-ary function of type D1x... xDn → Dt and t1, ..., tn are terms of type D1, ..., Dn respectively, then f(t1, ..., tn) is a term of type Dt.

Definition : A (well-formed) formula is defined inductively as follows:
(a) If p is an n-ary predicate of type D1x... xDn → {true, false} and t1, ..., tn are terms of type D1, ..., Dn respectively, then p(t1, ..., tn) is a formula(called an atomic formula or an atom).
(b) If F and G are formulas then so are (~F), (F ∨ G), (F ∧ G), (F → G), and (F ↔ G).
(c) If F is a formula and x is a variable of type Dx, then ($\forall_{x\in D_x}$ F) and ($\exists_{x\in D_x}$ F) are formulas.

Definition : The typed first order language L given by an alphabet A consists of the set of all formulas constructed from symbols of A.

III. Interpretations of Typed First Order Languages

This section introduces a typed interpretation of a first order language, a typed Herbrand interpretation, and a typed Herbrand interpretation I *.

Definition : An (typed) interpretation of a typed first order language(TFOL) L consists of the following:
(a) A nonempty set D′ called the domain of the interpretation, such that $D' = \bigcup_{i=1}^{m} D'i$, where each D′i is a nonempty, homogenous, and countable set.
(b) For each constant of type Dc in L, the assignment of an element in D′c, where the data type of an element in D′c is the same as that of an element in Dc and there exists a natural number c(1 ≤ c ≤ m) such that D′c ⊆ D′.

(c) For each n-ary function of type D1x... xDn → Df in L, the assignment of a mapping from D1′x... xDn′ to Df′ where the data type of an element in Di is the same as that of an element in Di′ and for all Di′ there exists a natural number i(1≤i≤m) such that Di′⊆D′.

(d) For each n-ary predicate of type D1x... xDn → {true, false} in L the assignment of a relation on D1′x... xDn′ where the data type of an element in Di is the same as that of an element in Di′ and for all Di′ there exists a natural number i(1≤i≤m) such that Di′⊆D′.

Definition : Let I be an interpretation of a TFOL L. A variable assignment wrt I is an assignment to each variable v of type Dv in L of an element in D′v, where the data type of an element in Dv is the same as that of an element in D′v and there exists a natural number v(1≤v≤m) such that Dv′⊆D′.

Definition : Let I be an interpretation with domain $D' = U^m_{i=1} D'i$ of a TFOL L and let A be variable assignment.

The term assignment (wrt I and A) of the terms in L is defined as follows:

(a) Each variable is given its assignment according to A.

(b) Each constant is given its assignment according to I.

(c) Let $t'_1 \ldots, t'_n$ are the term assignments of t1,... tn and f′ is the assignment of f, then $f'(t'_1, \ldots, t'_n) \in D'_f$ is the term assignment of f(t1,..., tn), where the type of f′ is $D'_1x \ldots xD'_n \rightarrow D'_f$ and the type of t'_j is D'_j.

Definition : A ground term of type Dt is a term of type Dt not containing variables. Similarly, a ground atom is an atom not containing variables.

Definition : Let L be a TFOL. Suppose that the set of types in L is {D1,..., Dm}. The typed Herbrand universe Uth for L is the set $U^m_{i=1} D^h_i$ of all ground terms of type Di (1≤i≤m) which can be formed out of the constants of type Dj(1≤j≤m) and functions appearing in L. In the case that L has no constant of type Dk(1≤k≤m), we add some constant, say, a, of type Dk, to form ground terms. D^h_i corresponds to Di and the data type of an element in D^h_i is the same as that of an element in Di.

Definition : Let L be a TFOL. The typed Herbrand base Bth for L is the set of all ground atoms which can be formed by using predicates from L with typed ground terms from Uth as arguments.

Definition : Let L be a TFOL. An interpretation for L is a typed Herbrand interpretation (THI) if the following conditions are satisfied:

(a) The domain of the interpretation is Uth.

(b) Constants in L are assigned to themselves in Uth.

(c) If f is an n-ary function of type D1x.... x Dn → Df then f is assigned to the mapping from $D^h_1 x \ldots x D^h_n$ to D^h_f defined by $(t^h_1, \ldots, t^h_n) \rightarrow f(t^h_1, \ldots, t^h_n)$, where $t^h_i \in D^h_i$, $f(t^h_1, \ldots, t^h_n) \in D^h_f$, and the data type of an element in Di is the same as that of an element in D^h_i.

Definition : Let L be a TFOL and S be set of closed formulas of L. A typed Herbrand model (THM) for S is a THI for L which is a model for S.

Definition : Let L be a TFOL and S be a set of closed formulas of L. Given an interpretation I for S over a domain $D' = U^m_{i=1} D'i$, a THI I^* corresponding to I is a THI that satisfies the following condition:

Let h1,.., hn be element of Uth and the type of hi be D^h_i. Let every hi be mapped to some di ∊ D′i.

If p(d1,.., dh) is assigned T(or F) by I, then p(h1,.., hn) is also assigned T(or F) in I^*, where the predicate p has the type of D1x... xDn → {true, false}.

IV. Unsatisfiability of Set of Clauses

Definition : A literal is an atom or the negation of an atom.

Definition : A clause is a formula of the following form :

$$\forall x_1 \epsilon D_1 \cdots \forall x_S \epsilon D_S (L_1 \vee \cdots \vee L_m)$$

where each Li is a literal, x1, ..., xs are all the variables occurring in L1v... v Lm, and each Di is a type.

Lemma I : If an interpretation I over some domain $D' = U^m{}_{i=1} D'_i$ satisfies a set S of clauses, then any one of the THI I * corresponding to I also satisfies S.

Proof : First, let us consider how hi ϵ Uth is mapped to an element di ϵ D'_i. Suppose that a constant ci of type Di appears in S. When ci is assigned to an element di ϵ D'_i wrt I we map ci in Uth to di ϵ D'_i. In the case that a constant cj of type $D^h{}_i$ appears only in Uth of I *, cj is mapped to an arbitary element dj ϵ D'_j, where the data type of an element in $D^h{}_j$ is the same as that of an element in D'_j.

It is assumed that we have generated each hi ϵ Uth of type $D^h{}_i$ according to the definition of a THI. In general the set Hi={hi | hi is a term of type $D^h{}_i$ and is generated according to the definition of a THI} is countable. So it can be enumerated. D'_i is also a countable set.

If D'_i and Hi a are countable infinite sets then the mapping M is given as follows :

M : Hi → D'_i
M(hj) = dj
where D'_i = ⟨do, d1, ... ⟩
Hi = ⟨ho, h1, ... ⟩
jϵ N

If D'_i is a finite set then the mapping M can be given as written below.

M : Hi → D'_i
M(hj) = dj if $0 \leq j \leq n$
M(hj) = dn if $n < j$
where D'_i = ⟨do, d1, ..., dn⟩
Hi = ⟨ho, h1, ..., hn, hn+1, ... ⟩

If Hi is a finite set then we can always make a mapping from Hi to D'_i.

Now, we obtain a THI I * corresponding to I. Essentially, the mapping M is a renaming function.

Hence if | Hi | $\geq$ | D'_i | then any atom P(d1, ..., dn) in I is equivalent to P(hi, ..., hn) in I * where M(hi)=di. Thus if I satisfies S then I * also satisfies S. Even though | Hi | < | D'_i |, it is held that if I satisfies S then I * also satisfies S. This is followed by the fact that S is satisfied by I over D'_i then S is also satisfied by I * over a "narrower" domain Hi.

Lemma II : Let S be set of clauses. Then S is unsatisfiable iff S has no THI I * that satisfies S.

Proof : (→) It is obvious by definition of unsatisfiability. (←) Assume that S is not unsatisfiable. Then there is an interpretation I over some domain $D' = U^m{}_{i=1}$ D'i such that S is satisfied by I. Let I * be a THI corresponding to I. According to Lemma I, I * satisfies S. This contradicts the assumption that S has no THI I * that satisfies S. Therefore, S must be unsatisfiable.

References

〔1〕 Loveland, D.W., Automated Theorem Proving : A Logical Basis, North-Holland, New York, 1978.

〔2〕 Lloyd, J.W., Foundations of Logic Programming, Springer-Verlag, Berlin, 1984.

〔3〕 Robinson, J.A., Logic : Form and Function, Edinburgh University Press, 1979.

〔4〕 Chang, C.L. and Lee, R.C.T., Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.

〔5〕 Sun, H. and Wang, L., A Model Theory of Logic Programming Methodology, 2nd Int. Logic Programming Conf., Uppsala, Sweden, 1984.