5P−3

Inductive Inference of Logic Programs based on Algebraic Semantics

Yasubumi SAKAKIBARA

IIAS-SIS, FUJITSU LIMITED

1. Introduction

The study of inductive inference of logic programs was initially and mostly done by E.Shapiro and his work is known by the *Model Inference System* (MIS for short) [3]. Shapiro's algorithm deeply depends on the theory of predicate logic and logic programming. In the theory of logic programming, the least model $\cap M(LP)$ of a logic program LP is taken as the mathematical semantics, called *model-theoretic semantics*, for it. This semantics provides the denotation of a predicate symbol P in a logic program LP : $D(P) = \{(t_1,\dots,t_n) : P(t_1,\dots,t_n) \in \cap M(LP)\}$. On the other hand, *algebraic semantics* which connects between the theory of tree languages and the semantics of programming languages is now well known and recently introduced to logic programming in [2]. It studies the use of tree languages in the semantics of logic programming. In algebraic semantics, the set of terms computed by a logic program LP can be viewed as a tree language. That is to say, the denotation of P, $D(P)=\{t : P(t) \in \cap M(LP)\}$, is a tree language. From the result in [2], a set of trees is rational iff it can be computed by a linear monadic logic program, where a *rational* set of trees is a set of trees which can be recognized by a tree automaton and a *linear monadic logic program* is a class of logic programs defined by some syntactic restrictions. Therefore, the denotation of P can be written as $D(P) = \{t : t$ is accepted by a tree automaton T_A about P in LP$\}$. Based on such an algebraic semantics, we can establish a new inductive inference schema of logic programs so that the problem of inductive inference of logic programs is reduced to the problem of inductive inference of tree automata. Then we can get an efficient inductive inference method of logic programs which is extended from the one of automata [1].

2. Tree automaton and linear monadic logic program

Let Γ be a ranked alphabet. Γ^T denotes the set of all trees defined over a ranked alphabet Γ. Let \$ be a new symbol of arity 0. $\Gamma_\$^T$ denotes the subset of $(\Gamma \cup \{\$\})^T$ which is the set of all trees $t \in (\Gamma \cup \{\$\})^T$ such that t exactly contains one \$-symbol. For trees $t \in \Gamma^T$ and $s \in \Gamma_\$^T$, we define an operation $\cdot$ to replace the node labeled \$ of s with t, denoted by $s \cdot t$.

Definition A *deterministic (frontier to root) tree automaton* over Γ is a 4-tuple $T_A = (Q, \Gamma, \delta, F)$: (a) Q is a nonempty finite set of states, (b) Γ is a nonempty finite ranked alphabet, (c) $\delta = (\delta_0, \delta_1, \dots, \delta_m)$ is a state transition function such that $\delta_k : \Gamma_k \times Q^k \longrightarrow Q$ ($k = 0,1,\dots,m$), (d) $F \subseteq Q$ is the set of final states. δ can be extended to Γ^T by letting : $\delta(f(t_1,\dots,t_k)) = \delta_k(f, \delta(t_1),\dots,\delta(t_k))$. The tree t is *accepted* by T_A iff $\delta(t) \in F$. The set of trees accepted by T_A is the subset $L(T_A)$, called *rational*, of Γ^T defined as : $L(T_A) = \{t : \delta(t) \in F\}$.

Definition ([2]) A *linear monadic logic program* is a logic program in which all predicate symbols are monadic and all the terms occurring in atomic formulas belong to one of the following two forms : (a) x_i ($i \in N$), (b) $f(x_{i_1},\dots,x_{i_m})$ with $f \in \Gamma_m$, $\{i_1,\dots,i_m\} \subseteq N$ the i_k being pairwise distinct.

Proposition 1 If LMLP is a linear monadic logic program, then the set of trees $\{t : P(t) \in \cap M(LMLP)\}$ is rational. Conversely, if a set of trees T is rational, then there is a linear monadic logic program LMLP such that $T = \{t : P(t) \in \cap M(LMLP)\}$.

3. Predicate characterization matrix

Definition Let S be a finite set of Γ^T, $X(S) = \{f(\hat{u}) : f \in \Gamma_i, \hat{u} \in S^i$, and $f(\hat{u}) \notin S$ for $i \geq 0\}$, and E be a finite set of $\Gamma_\$^T$. S is called *subtree-closed* if $s \in S$ implies all subtrees of s are elements of S. E is called *\$-prefix-closed w.r.t.* S if $e \in E$ except \$ implies there exists an e' in E such that $e = e' \cdot f(s_1,\dots,s_{i-1},\$, s_i,\dots,s_{n-1})$ for some $s_1,\dots,s_{n-1} \in S$. A predicate characterization matrix is a triple (S, E, M) where M is a matrix such that (1) the rows are labeled with the elements of $S \cup X(S)$, (2) the columns are labeled with the elements of E, (3) each entry of M is either 0 or 1, (4) if $s_i, s_j \in S \cup X(S)$ and $e_i, e_j \in E$ and $e_i \cdot s_i = e_j \cdot s_j$, then the (s_i, e_i) and (s_j, e_j) positions in M must have the same entry. The *data contained in* M is $D(M) = \{(e \cdot s, y) : s \in S \cup X(S), e \in E$, and $y \in \{0, 1\}\}$. For s in $(S \cup X(S))$, *row(s)* denotes the finite function f from E to {0, 1} defined by $f(e) = D(M)(e \cdot s)$. A predicate characterization matrix is called *closed* if every row(x) of $x \in X(S)$ is identical to some row(s) of $s \in S$. A

Inductive Inference of Logic Programs based on Algebraic Semantics

Yasubumi SAKAKIBARA

International Institute for Advanced Study of Social Information Science, FUJITSU LIMITED predicate characterization matrix is called *consistent* if whenever s_1 and s_2 are in S such that $\mathrm{row}(s_1)$ is equal to $\mathrm{row}(s_2)$, for all $f \in \Gamma_n$ and $u_1,\ldots,u_{n-1} \in S$, $\mathrm{row}(f(u_1,\ldots,u_{i-1},s_1,u_i,\ldots,u_{n-1}))$ is equal to $\mathrm{row}(f(u_1,\ldots,u_{i-1},s_2,u_i,\ldots,u_{n-1}))$ for $0 \leq i \leq n$. Let (S, E, M) be a closed, consistent predicate characterization matrix such that E contains \$. The *constructed linear monadic logic program* LMLP_M over Γ from (S, E, M) is defined with predicate set $\{R_{\mathrm{row}(s)}(x) : s \in S\}$, calling predicate P, and the set of clauses LMLP_M as follows :

$$\mathrm{LMLP}_M = \{P(x) \leftarrow R_{\mathrm{row}(s)}(x) : s \in S \text{ and } D(M)(s) = 1\}$$
$$\cup \{R_{\mathrm{row}(f(s_1,\ldots,s_n))}(f(x_1,\ldots,x_n)) \leftarrow R_{\mathrm{row}(s_1)}(x_1),\ldots,R_{\mathrm{row}(s_n)}(x_n) : f \in \Gamma_n\}$$
$$\cup \{R_{\mathrm{row}(a)}(a) \leftarrow \ : a \in \Gamma_0\}.$$

The idea of the characterization matrix is essentially the extension of Angluin's one [1].

Theorem 2 Suppose that (S, E, M) is a closed, consistent predicate characterization matrix such that S is subtree-closed and E is \$-prefix-closed w.r.t. S. Then the constructed linear monadic logic program LMLP_M agrees with the data in M. That is, for s in $(S \cup X(S))$ and e in E, $P(e \cdot s) \in \cap M(\mathrm{LMLP}_M)$ iff $D(M)(e \cdot s) = 1$.

4. Inductive inference algorithm for linear monadic logic program

Input : An oracle EX() for a sufficient set of examples of the predicate P in the unknown linear monadic logic program LMLP_U, and an oracle MEMBER(P(t)) on a ground atom P(t) as input for a membership query to output 1 or 0 according to whether P(t) is true in $\cap M(\mathrm{LMLP}_U)$,

Output : A sequence of conjectures of linear monadic logic program,

Procedure :

```
S := ∅; E := {$}; LMLP := ∅; Examples := ∅;
do forever
add an example EX() to Examples;
while there is a negative example $-P(t) \in$ Examples such that
    LMLP ⊢ P(t) or there is a positive example $+P(t) \in$
    Examples such that LMLP ⊬ P(t);
    add t and all its subtrees to S;
    extend (S, E, M) to E·(S∪X(S)) using MEMBER;
    repeat
    if (S, E, M) is not consistent
    then find $s_1$ and $s_2$ in S, $f \in \Gamma_n$, $u_1,\ldots,u_{n-1} \in S$, $e \in E$, and i
        ($1 \leq i \leq n$) such that row($s_1$) is equal to row($s_2$) and
        $D(e \cdot f(u_1,\ldots,u_{i-1},s_1,u_i,\ldots,u_{n-1})) \neq$
            $D(e \cdot f(u_1,\ldots,u_{i-1},s_2,u_i,\ldots,u_{n-1}))$;
        add $e \cdot f(u_1,\ldots,u_{i-1},\$,u_i,\ldots,u_{n-1})$ to E;
        extend (S, E, M) to E·(S∪X(S)) using MEMBER;
    if (S, E, M) is not closed;
    then find $f(\hat{u}) \in X(S)$ for $\hat{u} \in S^*$ and $f \in \Gamma_n$ such that row($f(\hat{u})$)
        is different from row(s) for all $s \in S$;
        add $f(\hat{u})$ to S;
        extend (S, E, M) to E·(S∪X(S)) using MEMBER;
    until (S, E, M) is closed and consistent;
    LMLP := $\mathrm{LMLP}_M$;
end;
output LMLP;
end.
```

We call an example t presented by EX a *counter-example* when the last conjecture LMLP_M does not agree with t.

(Correctness) The algorithm identifies in the limit a linear monadic logic program LMLP such that $\{t : P(t) \in \cap M(\mathrm{LMLP})\}$ is equal to the denotation of P by the unknown model.

(Time complexity) Let n be the number of states in the minimum tree automaton for the denotation of the predicate P in the unknown model, and m be the maximum size of any counter-examples presented by EX. Then the total time which the while loop consumes during the running of the algorithm can be bounded by a polynomial function of m and n.

5. Concluding remarks

MIS is the excellent and only existing system to infer logic programs. MIS can infer a whole class of logic programs, but ours only for a restricted class of logic programs. However our algorithm has several unique features compared with MIS. (1) Our algorithm is based on algebraic semantics and the target of the inference is a tree language computed by a logic program, and hence it is different from Shapiro's approach. (2) It is not easy to analyse the time complexity of inductive inference algorithm, and neither in MIS. We have shown in the last section the time complexity of our algorithm. (3) Our algorithm is based on the constructive method, while MIS is based on the enumerative method, where the constructive method systematically use examples to construct the conjecture and the enumerative method use them to select a conjecture in enumeration. It is said that the constructive method is in general more efficient than the enumerative method. (4) In our algorithm, the predicate symbol P and its interpretation are only given as the observational language and the oracle, and any information about the hypothesis language is not given. The algorithm automatically generates other predicates whenever they are needed. However in MIS, all predicates used to construct the conjectures and those intended interpretations must also be given as the hypothesis language and the oracle, and this is often referred to as the problem about *theoretical terms* of MIS.

This is part of the work in the major R&D of the FGCP, conducted under program set up by MITI.

References

[1] Angluin,D., Learning regular sets from queries and counter-examples, *Yale DCS TR-464*, 1986.
[2] Marque-Pucheu,G., Rational set of trees and the algebraic semantics of logic programming, *Acta Informatica 20*, 249-260 (1983).
[3] Shapiro,E., Algorithmic program debugging, MIT Press, 1983.