Representing Inductive Inference with SOLD-Resolution

Akihiro YAMAMOTO*
Fachgebiet Intellektik, Fachbereich Informatik
Technische Hochschule Darmstadt
Alexanderstr. 10, D64283 Darmstadt, GERMANY
Email : yamamoto@intellektik.informatik.th-darmstadt.de

Abstract

In this paper we discuss how to use SOL-resolution, proposed by Inoue, for inductive inference. In order to use some results on Horn clause logic, we define a new inference method SOLD-resolution. SOLD-resolution is obtained by adding the skipping rule to SLD-resolution, and is also regarded as a special type of SOL-resolution. When we use SOLD-resolution to hypothesis finding, every derived hypothesis is a negation of a clause, while in induction it must be a clause. We give two solutions to the problem. The first solution is to use inverse entailment proposed by Muggleton. In order to give an inference procedure based on it we add the reducing rule and the saturation rule to SOLD-resolution. The reducing rule is another rule used in SOL-resolution, and saturation is an inference rule proposed by Rouveirol. With our procedure we can derive definite clauses as hypotheses. The second solution is to give a restriction that every hypothesis should be a unit program. Under the restriction we give an inductive inference procedure by combining SOLD-derivation with the identification rule proposed by Muggleton.

1. Introduction

Our current research subject is how to represent inductive inference in a deductive manner. Inductive inference means a process generating rules from given examples. The purpose of the research is to make it easy to compare and combine inductive inference with other inference methods, e.g. abduction. For the purpose we concern ourselves with SOL-resolution proposed by Inoue [Inoue, 1992]. He succeeded to represent SOL-resolution abduction, prime implicants, ATMS, and circumscription, but not inductive inference. In this paper we try to represent inductive inference with SOL-resolution, by combining it with some inference rules proposed in the research of inductive inference.

In the following section we start our discussion with comparing induction with abduction, and we explain a problem which arises when we represent inductive inference with SOL-resolution. The problem is that the formulas derived by SOL-derivation are not in general useful for generating hypotheses in induction. In this paper we solve the problem in Horn clause logic. In Section 3 we introduce SOLD-resolution by adding the skipping rule for SOL-resolution to SLD-resolution. SOLD-resolution can be regarded as SOL-resolution for Horn clause logic. In Section 4, by SOLD-resolution with the reducing rule and the saturation rule, we give a procedure with which definite clauses are derived as hypotheses. The reducing rule is another rule adopted for SOL-resolution. Saturation was invented in the research of inductive inference by Rouveirol [Rouveirol, 1992]. In Section 5 we show that if each hypothesis is a conjunction of unit clauses, we can derive it with SOLD-resolution and the identification rule proposed by Muggleton [Muggleton, 1990]. In the last section we give some comments on the relation between Muggleton's result and our result, and show some subjects for our future research.

2. From Abduction to Induction

At first we fix a formalization of abduction because various ones have already been proposed [Cox and Pietrzykowski, 1986; Demolonbe and Fariñas de Cerro, 1991; Hirata, 1995; Inoue, 1992; Poole, 1988; Pople Jr., 1973]. We adopt the formalization by Inoue, which is an extension of the one by Poole. In the formalization, abduction is to find an explanation H from a given surprising fact E and a given background theory B. (We choose the symbols H and E to avoid confusion in comparing abduction and induction.) Each of H, E, and B is a formula in first-order logic. We fix three sets $\mathcal{H}$, $\mathcal{E}$, and $\mathcal{B}$ of formulas and assume that H, E and B belong to $\mathcal{H}$, $\mathcal{E}$, and $\mathcal{B}$, respectively. The explanation H must satisfies the following conditions:

(A1) $B \wedge H$ is consistent.

(A2) $B \wedge H \models E$.

*The author is a visiting researcher from the Division of Electronics and Information Engineering, Hokkaido University, JAPAN.

Inductive inference has also been formalized in various ways. In the traditional computational theory a comprehensive framework has been proposed [Angluin and Smith, 1983]. We give our own formalization suitable for our purpose. An inductive system $\mathcal{M}$ sequentially takes inputs of examples E_1, E_2, E_3,... and returns hypotheses B_1, B_2, B_3,.... (We choose the symbol B_i because in our formalization the current hypothesis corresponds to the background theory in abduction.) Some initial hypothesis B_0 is given to $\mathcal{M}$. Two sets $\mathcal{E}$ and $\mathcal{B}$ of first-order formula are fixed, and each example E_i and each hypothesis B_i must belong to $\mathcal{E}$ and $\mathcal{B}$, respectively. We assume the following conditions for E_i and B_i $(i = 1, 2, \ldots)$:

(I1) B_i is consistent.

(I2) $B_i \models E_1 \wedge E_2 \wedge \ldots \wedge E_i$, that is, each example is positive and each hypothesis is consistent with the examples already given to $\mathcal{M}$.

(I3) $B_i = B_{i-1} \wedge H_i$ where H_i is in some set $\mathcal{H}$. The system $\mathcal{M}$ generates the hypothesis incrementally.

This formalization has already been used to design some inductive inference systems, e.g. ITOU [Rouveirol, 1992] and Progol [Muggleton, 1995]. The setting also attracts attentions of researchers in inductive inference theories [Cohen, 1995; Arimura *et al.*, 1995].

Now we compare induction with abduction. From assumptions (I1), (I2) and (I3) we can get the following conditions:

(I4) $B_{i-1} \wedge H_i$ is consistent.

(I5) $B_{i-1} \wedge H_i \models E_i$.

By comparing (I4) and (I5) with (A1) and (A2), we may consider that the inductive inference system $\mathcal{M}$ iteratively uses a procedure which can be regarded as abduction.

When we are in clausal logic, abduction in Inoue's formalization can be represented with SOL-resolution. Therefore we get an idea that we also represent induction with SOL-resolution. To realize our idea we must solve a problem. Let $\mathcal{B}$ is a set of clausal theories. Then for abduction, both $\mathcal{E}$ and $\mathcal{H}$ are sets of *negations* of clauses, while they should be sets of clauses for induction. This problem is due to the condition (I3). If we apply the setting for abduction to induction, the condition (I3) does not satisfied in general because $B \wedge H$ does not always belong to $\mathcal{B}$ for any pair of B in $\mathcal{B}$ and H in $\mathcal{H}$.

In the following sections, we show two solutions to the problem based on Horn clause logic. Horn clause logic is narrower than clausal logic, but we adopt it from two reasons. Firstly some results on Horn clause logic, especially the switching lemma for SLD-resolution, is useful for our discussions. Secondly many researches on inductive inference, including the four researches listed above, are aimed to develop inference systems which generates definite programs from examples.

3. SOLD-Derivation

We assume the reader is familiar with the concepts on first-order logic and Horn clause logic.

A *definite program* is a finite conjunction of definite clauses. A *Horn clause* is a formula of the form

$$F = \forall x_1 \ldots x_k (A_n \vee \neg A_1 \vee \ldots \vee \neg A_m)$$

where $n \geq 0$, $m \geq 0$, A_i's are all atoms, and $x_1, \ldots, x_k$ are all variables occurring in the atoms. We represent the formula in the form of implication:

$$A_n \leftarrow A_1, A_2, \ldots, A_m.$$

F is called a *definite clause* if $n = 1$ and is called a *goal clause* if $n = 0$. F^+ and F^- respectively denote clauses $A_n \leftarrow$ and $\leftarrow A_1, A_2, \ldots, A_m$.

SOL-resolution is based on OL-resolution in clausal logic. For Horn clause logic, we prepare a new deduction method based on SLD-resolution, and call it SOLD-resolution. In SLD-resolution an atom in goal clause is selected according to a *computation rule* and the variables in each input clause are standardized apart(see [Lloyd, 1987]). We assume the two conditions also for SOLD-derivation.

Definition. Let P be a definite program, G be a goal, and R be a computation rule. An *SOLD-derivation* for (P, G) is a finite sequence of quadruplets $\langle G_i, F_i, \theta_i, C_i \rangle$ $(i = 0, 1, 2, \ldots n)$ which satisfies the following conditions:

1. G_i, and F_i are goal clauses, θ_i is a substitution, and C_i is a variant of a clausal formula in P the variables of which are standardized apart by renaming.
2. $G_0 = G$ and $F_0 = \Box$.
3. $G_n = \Box$ and $F_n = F$.
4. If $G_i = \leftarrow A_1, \ldots, A_k$ $F_i = \leftarrow B_1, \ldots, B_h$ and A_m is the atom selected from G by R, then either of the following two holds:
   (a) (Skip) $G_{i+1} = \leftarrow A_1, \ldots, A_{m-1}, A_{m+1}, \ldots, A_k$ and $F_{i+1} = \leftarrow B_1, \ldots, B_h, A_m$.
   (b) (Resolve) θ_i is an mgu of C_i^+ and A_m, $G_{i+1} = \leftarrow (A_1, \ldots, A_{m-1}, C^-, A_{m+1}, \ldots, A_k)\theta_i$, and $F_{i+1} = F_i\theta_i$.

The goal F_n is called the *consequence* of the SOLD-derivation.

If the skipping rule is not applied in constructing an SOLD-derivation $\langle G_i, F_i, \theta_i, C_i \rangle$ $(i = 1, \ldots, n)$, every F_i is $\Box$ and the derivation can be regarded as an SLD-derivation. Therefore SOLD-resolution is an extension of SLD-resolution.

Definition. If $\Box$ is a consequence of an SOLD-derivation, the derivation is called is called an *SLD-refutation.*

The derivation rule (4(b)) for SOLD-resolution is different from the one for SOL-resolution [Inoue, 1992] in the following three points:

1. In an SOLD-derivation no variant of G can be used in C_i, while some variant of G can be used in an SOL-deduction.

2. The literal $A_m\theta_i$ is removed in an SOLD-derivation, while it is framed in a SOL-deduction. Therefore the ancestry rule for SOL-resolution is not necessary for SOLD-resolution.
3. Neither the factoring rule nor the merging rule for SOL-resolution is used in constructing SOLD-derivations.

These differences are due to the fact that P and G are respectively a definite program and a goal clause. SOL-resolution is complete as a consequence finding method. In spite of the differences above, SOLD-resolution is also a complete consequence finding method. The completeness can be shown by the result in [Nienhuys-Cheng and de Wolf, 1995].

Proposition 1. *Let P be a definite clause and G be a goal clause. Then a goal clause F is a logical consequence of $P \wedge G$ iff there is a SOLD-derivation whose consequence subsumes F.*

Proof. It is shown in [Nienhuys-Cheng and de Wolf, 1995] that SLD-resolution without any computation rule is complete as a consequence finding rule. A computation rule R is assumed for SOLD-resolution, but, by using the skipping rule 4(a), we can simulate SLD-resolution without any computation rule.

4. Finding Definite Clauses

In this section we give an inductive inference procedure based on SOLD-derivation when $\mathcal{B}$ is the set of all definite programs, $\mathcal{H}$ is the set of all definite clauses, and $\mathcal{E} \subseteq \mathcal{H}$. This setting is assumed for ITOU and Progol.

At first we introduce the complement of a Horn clause.

Definition. The *complement* of $F = A_0 \leftarrow A_1, \ldots, A_m$ is a formula

$$\overline{F} = (\neg A_0 \wedge A_1 \wedge \ldots \wedge A_m)\sigma_F$$

where σ_F is a substitution which replaces each variable in F with a Skolem constant symbol.

For a Horn clause F, $\overline{F^+}$ is a definite program, $\overline{F^-}$ is a goal clause, and $\overline{F} = \overline{F^+} \wedge \overline{F^-}$.

Now we give a procedure which takes a definite program B and a definite clause E as inputs and returns definite clauses H such that $B \wedge H \models E$. Our procedure is founded on *inverse entailment* in [Muggleton, 1995], that is, the H is obtained by generalizing some ground clause K such that $B \wedge \overline{E} \models \overline{K}$. Since he complement $\overline{K}$ is a conjunction of ground literals, we need some methods to derive them. The next lemma shows that the positive literals can be obtained by the bottom-up evaluation of $B \wedge \overline{E^-}$.

Lemma 1 ([Yamamoto, 1997]). *Let A be a ground atom. Suppose that E is not a tautology and that $B \not\models E$. Then $B \wedge \overline{E} \models A$ iff $B \wedge \overline{E^-} \models A$.*

We derive the negative literal in $\overline{K}$ by SOLD-resolution. For the purpose we must modify SOLD-resolution because it may derive a consequence $F = \leftarrow A_1, \ldots, A_n$ such that $n \geq 2$. We add a process to check whether or not the consequence F is subsumed by a goal clause of the form $\leftarrow A$. We can check it by applying the well-known *mgu algorithm* to the set $\{A_1, \ldots, A_n\}$. The check is also regarded as the iterative application of the reducing rule for SOL-deduction.

Definition. A goal clause $\leftarrow A$ is *SOLDR-derivable* from (P, G) if there is a consequence of SOLD-derivation F which subsumes $\leftarrow A$.

Remark 1. We can embed the reducing rule into SOLD-resolution by replacing condition 4 for an SOLD-derivation with the following 4':

4' If G_i is $\leftarrow A_1, \ldots, A_k$ and A_m is the atom selected from G_i by R, then one of the following three holds:
- (a) (Skip) $F_i = \Box$, $G_{i+1} = \leftarrow A_1, \ldots, A_{m-1}, A_{m+1}, \ldots, A_k$ and $F_{i+1} = \leftarrow A_m$.
- (b) (Reduce) $F_i = \leftarrow A$, θ_i is an mgu of A and A_m, $G_{i+1} = G_i\theta_i$, and $F_{i+1} = F_i\theta_i$.
- (c) (Resolve) θ_i is an mgu of C_i^+ and A_m, $G_{i+1} = \leftarrow (A_1, \ldots, A_{m-1}, C^-, A_{m+1}, \ldots, A_k)\theta_i$, and $F_{i+1} = F_i\theta_i$.

Now we put the set

$$SOLDR(P, G) = \left\{ A \;\middle|\; \begin{array}{l} \leftarrow A \text{ is a ground instance of a goal} \\ \leftarrow A' \text{ which is SOLDR-derivable} \\ \text{from } (P, G). \end{array} \right\}.$$

Theorem 1. *Let P be a program, G be a goal, and $P \wedge G$ is consistent. Then*

$$SOLDR(P, G) = \{A \;;\; A \textit{ is a ground atom and } P \wedge G \models \neg A\}$$

Proof. ($\subseteq$) Let $P \wedge G \models \neg A$. Then there is an SLD-refutation for $(P \wedge (A \leftarrow), G)$. Using the switching lemma, we have that A is in $SOLDR(P, G)$.
($\supseteq$) Let A is in $SOLDR(P, G)$. Then there is an SLD-refutation for $(P \wedge (A \leftarrow), G)$. This means $P \wedge G \models \neg A$.

Now we given an inference procedure using SOLDR-derivation. Let $M(P)$ denote the least Herbrand model of a definite program P.

Procedure 1.
Input: a definite program B and a definite clause E
Output: a definite clause H
Method:

1. *If $B \models E$ then stop. Otherwise go to step 2.*
2. *Derive an atom A_0 in $SOLDR(B \wedge \overline{E^-}, \overline{E^+})$.*
3. *Derive some atoms A_1, A_2, ..., A_n in $M(B \wedge \overline{E^-})$ by bottom-up derivation.*
4. *Return a clause H which is a generalization of $A_0 \leftarrow A_1, A_2, ..., A_n$*

The step 3 is called saturation in Rouveirol [Rouveirol, 1992]. In the original definition of saturation the atom A_0 in step 2 is $\neg\overline{E^+}$, while in our procedure it is derived by SOLDR-resolution. The hypothesis generated by Procedure 1 is characterized as follows.

Theorem 2. *Suppose that E is not a tautology and that $B \not\models E$. Then every hypothesis H generated by Procedure 1 subsumes a ground clause K such that $B \wedge K \models E$.*

Proof. ($\Longrightarrow$) Let $K = A_0 \leftarrow A_1, A_2, ..., A_n$ be a clause such that $A_0 \in SOLDR(B \wedge \overline{E^-}, \overline{E^+})$ and $A_1, A_2, ..., A_n \in M(B \wedge \overline{E^-})$. Then from Theorem 1 $B \wedge \overline{E} \models \overline{K}$ which implies $B \wedge K \models E$ because K is ground.
($\Longleftarrow$) Conversely, suppose that H subsumes a ground clause $K = A_0 \leftarrow A_1, A_2, ..., A_n$ such that $B \wedge K \models E$. Then it holds that

$$B \wedge \overline{E} \models \neg A_0 \wedge A_1 \wedge \ldots \wedge A_n.$$

From Theorem 1 and Lemma 1 we have that $A_0 \in SOLDR(B \wedge \overline{E^-}, \overline{E^+})$ and that $A_1, A_2, ..., A_n \in M(B \wedge \overline{E^-})$.

We have shown in [Yamamoto, 1997] that every H obtained by Procedure 1 subsumes E relative to B in the sense of [Plotkin, 1971].

Example 1. We demonstarte how Procedure 1 works with an example in [Muggleton, 1995]. Let us define B_1 and E_1 as follows:

$$B_1 = \begin{array}{l}(\text{pet}(x) \leftarrow \text{cat}(x)) \wedge \\ (\text{cuddly-pet}(x) \leftarrow \text{small}(x), \text{fluffy}(x), \text{pet}(x))\end{array}$$
$$E_1 = \text{cuddly-pet}(x) \leftarrow \text{fluffy}(x), \text{cat}(x).$$

Since

$$M(B_1 \wedge \overline{E_1}) = \{\text{fluffy}(c_x), \text{cat}(c_x), \text{pet}(c_x)\}$$

and

$$SOLDR(B_1 \wedge \overline{E_1}^+, \overline{E_1}^-) = \{\text{small}(c_x)\},$$

a definite clause

$$H = \text{small}(x) \leftarrow \text{fluffy}(x), \text{cat}(x), \text{pet}(x)$$

is one of the outputs.

5. Finding Unit Programs

In the last section we show a method to derive one clause H from given B and E by adding the reducing rule to SOLD-resolution. Then one might expect that, if we do not use the reducing rule, We could derive definite programs consisting of more than one clauses. In this section we show that we can derive programs consisting of more than one unit clauses.

Definition. A *unit program P* is a program of the form

$$(A_1 \leftarrow) \wedge (A_2 \leftarrow) \wedge \ldots \wedge (A_n \leftarrow).$$

Let $\mathcal{H}$ is the set of all unit programs and $\mathcal{E}$ is a set of all definite clauses. Without loss of generality we can assume that the variables introduced in SOLD-resolution occurs in no programs in $\mathcal{H}$.

Theorem 3. *Let H be a unit program and E a definite clause, and B a definite program. Then $B \wedge H \models E$ iff there is a goal F such that*

1. *there is an SOLD-derivation for $(B \wedge \overline{E^-}, \overline{E^+})$ whose consequence is F, and*
2. *there is an SLD-refutation for (H, F).*

Proof. ($\Longrightarrow$) The assumption $B \wedge H \models E$ is equivalent to the condition that there is an SLD-refutation $\langle G_i, \Box, \theta_i, C_i \rangle$ $(i = 0, 1, 2, \ldots n)$ for $(B \wedge H \wedge \overline{E^-}, \overline{E^+})$. Using the switching lemma, we can assume that the input clauses $C_1, C_2, \ldots, C_k$ are in variants of clausual formula in $B \wedge \overline{E^-}$, and the rest is not. Then there is an SLD-refutation for (H, G_k).
($\Longleftarrow$) Let $\langle G_i, F_i, \theta_i, C_i \rangle$ $(i = 0, 1, 2, \ldots n)$ be the SOLD-derivation. Since there is an SLD-refutation for $(H, F_n = F)$, the SOLD-derivation can be extended to a SLD-refutation for $(B \wedge H \wedge \overline{E^-}, \overline{E^+})$.

The rule with which a unit clause $C_i \leftarrow$ is derived from $\leftarrow A_i$ was called identification in [Muggleton, 1990]. The theorem above shows that the hypothesis H can be derived with repeated application of identification. Identification can be realized by the next lemma.

Lemma 2. *Two atoms A and B which share no variables are unifiable iff B is a generalization of some ground instance of A.*

Procedure 2.
Input: a definite program B and a definite clause E
Output: a unit program H
Method:

1. *If $B \models E$ then stop. Otherwise go to step 2.*
2. *Derive a goal clause $F = \leftarrow B_1, B_2, \ldots, B_n$ which is the consequence of some SOLD-derivation for $(B \wedge \overline{E^-}, \overline{E^+})$.*
3. *For every i from 1 to n, apply the followings:*
   (a) *Make A_i is a generalization of a ground instance of B_i such that every variable in A_i is new.*
   (b) *Apply the mgu θ_i of A_i and B_i to F.*
4. *Return $H = (A_1 \leftarrow) \wedge (A_2 \leftarrow) \wedge \ldots \wedge (A_n \leftarrow)$.*

Example 2. We illustrate how Procedure 2 works with the example in [Inoue, 1992]. Let us assume the following a background theory B_2 and an example E_2 are given:

$$B_2 = p(x, z) \leftarrow p(x, y), p(y, z),$$
$$E_2 = p(a, b) \leftarrow .$$

Then the goal clauses which are consequences of some SOLD-derivations for $(B_1 \wedge \overline{E_2}^-, \overline{E_2}^+)$ are :

$$\begin{array}{l}\leftarrow p(a, b), \\ \leftarrow p(a, x_1), p(x_1, b), \\ \leftarrow p(a, x_2), p(x_2, x_3), p(x_3, b), \\ \ldots .\end{array}$$

From the goal clauses we get the following hypotheses of unit programs:

$$p(a, b) \leftarrow,$$

$$(p(a,y_1) \leftarrow) \wedge (p(y_2,b) \leftarrow),$$
$$(p(a,a) \leftarrow) \wedge (p(a,b) \leftarrow),$$
$$(p(a,b) \leftarrow) \wedge (p(b,b) \leftarrow),$$
$$(p(a,y_1) \leftarrow) \wedge (p(y_2,y_3) \leftarrow) \wedge (p(y_4,b) \leftarrow),$$
$$(p(a,a) \leftarrow) \wedge (p(a,y_3) \leftarrow) \wedge (p(y_4,b) \leftarrow),$$
$$\ldots .$$

6. Concluding Remarks

In this paper we give two procedures for inductive inference based on SOLD-resolution.

Procedure 1 is an implementation of inverse entailment in [Muggleton, 1995] in Horn clause logic. We assumed that E is not a tautology and that $B \not\models E$. Muggleton also proposed a procedure based on inverse entailment in Horn clause logic, but he assumed more restrictions to B, E and H. Since the two condition are necessary for Theorem 2, our procedure is one of the most general ones based on inverse entailment in Horn clause logic. We have also given more results on inverse entailment [Yamamoto, 1996; 1997]. To extend of Procedure 1 so that we can derive non-definite clauses is in our future research plan.

In general it is quite difficult to extend Procedure 2 so that we can generate arbitrary definite programs as hypotheses, because such programs may use some recursive calls of predicates. It should also be in our future plan to find some interesting set of hypotheses which can derive with some extension of Procedure 2.

Acknowledgements

The author wishes to thank Prof.Dr. Wolfgang Bibel and the members of his group for many discussions and suggestions on this issue. He also thanks the anonymous referees for their valuable comments to improve this paper.

References

[Angluin and Smith, 1983] D. Angluin and C. H. Smith. Inductive Inference: Theory and Methods. *Computing Surveys*, 15:237–269, 1983.

[Arimura *et al.*, 1995] H. Arimura, H. Ishizaka, and T. Shinohara. Learning Unions of Tree Patterns Using Queries. In *Proceedings of the 6th Workshop on Algorithmic Learning Theory(LNAI 997)*, pages 66–79. Springer-Verlag, 1995.

[Cohen, 1995] W. W. Cohen. Pac-learning Recursive Logic Programs: Efficient Algorithms. *J. of Artificial Intelligence Research*, 2:501–539, 1995.

[Cox and Pietrzykowski, 1986] P. T. Cox and T. Pietrzykowski. Incorporating Equality into Logic Programming. *Ann. Pure Appl. Logic*, 31:177–189, 1986.

[Demolonbe and Farinàs de Cerro, 1991] R. Demolonbe and L. Farinàs de Cerro. An Inference Rule for Hypothesis Generation. In *Proc. of IJCAI-91*, pages 152–157, 1991.

[Hirata, 1995] K. Hirata. A Classification of Abduction: Abduction for Logic Programming. In *Machine Intelligence 14*, pages 397–424. Oxford University Press, 1995.

[Inoue, 1992] K. Inoue. Linear Resolution for Consequence Finding. *Artificial Intelligence*, 56:301–353, 1992.

[Lloyd, 1987] J. W. Lloyd. *Foundations of Logic Programming : Second, Extended Edition.* Springer - Verlag, 1987.

[Muggleton, 1990] S. Muggleton. Inductive Logic Programming. In S. Arikawa and S. Goto and S. Ohsuga and T. Yokomori, editor, *Proceedings of the First International Workshop on Algorithmic Learning Theory*, pages 42–62. JSAI, 1990.

[Muggleton, 1995] S. Muggleton. Inverse Entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[Nienhuys-Cheng and de Wolf, 1995] S.-H. Nienhuys-Cheng and Ronald de Wolf. The Subsumption Theorem Revisited: Restricted to SLD-Resolution. Computer Science Netherlands 95, 1995.

[Plotkin, 1971] G. D. Plotkin. *Automatic Methods of Inductive Inference.* PhD thesis, Edinburgh University, 1971.

[Poole, 1988] D. Poole. A Logical Framework for Default Reasoning. *Artificial Intelligence*, 36:27–47, 1988.

[Pople Jr., 1973] H. E. Pople Jr. On the Mechanization of Abductive Logic. In *Proc. of IJCAI-73*, pages 147–152, 1973.

[Rouveirol, 1992] C. Rouveirol. Extentions of Inversion of Resolution Applied to Theory Completion. In S. Muggleton, editor, *Inductive Logic Programming*, pages 63–92. Academic Press, 1992.

[Yamamoto, 1996] A. Yamamoto. Improving Theories for Inductive Logic Programming Systems with Ground Reduced Programs. Technical Report AIDA-9619, Fachgebiet Intellektik, Fachbereich Informatik, Technische Hochschule Darmstadt, 1996. Submitted to New Generation Computing.

[Yamamoto, 1997] A. Yamamoto. Which Hypotheses Can Be Found with Inverse Entailment? Submitted to the Seventh International Workshop on Inductive Logic Programming. Its extended abstract is submitted to the IJCAI'97 Workshop on Frontiers of Inductive Logic Programming, 1997.