A Query Evaluation Method for ASP with Abduction

Ken Satoh

National Institute of Informatics, Tokyo, Japan

Abstract

In this paper, we present a goal-directed proof procedure for ASP with abduction. Our proposed procedure in this paper is correct for any consistent abductive framework proposed in [Kakas90a].

In other words, if the procedure succeeds, there is a set of hypotheses which satisfies a query, and if the procedure finitely fails, there is no such set. If we do not consider abducibles, this procedure is a goal-directed proof procedure for ASP as well.

Keywords

Logic Programming, Abduction, Topdown Proof Procedure

1. Introduction

In this paper, we present a query evaluation method that is correct for every consistent[1] abductive framework proposed in [Kakas90a]. If our procedure answers "yes", then there is a set of hypotheses which satisfies a query. If our procedure answers "no", then there is no set of hypotheses which satisfies a query.

Our procedure can be regarded as a correction of Kakas and Mancarella's procedure [Kakas90b] in the following two points.

Forward evaluation of rules:
It is important to use integrity constraints to exclude undesirable results from abduction. However, their procedure manipulates a class of integrity constraints in which there is at least one abducible predicate in each integrity constraint.

Check for Implicit Deletions:
We check consistency for "implicit deletions" first observed by Sadri and Kowalski [Sadri88]. The proposed methods [Kakas90b] do not check consistency of implicit deletion. We believe that this lack of check is a major culprit of incorrectness in these methods. Inversely, we check the implicit deletion in our procedure.

For example, consider the following program with an abducible predicate $normal\_barber^{\dagger}$ [2].

2nd Workshop on Goal-directed Execution of Answer Set Programs (GDE'22), August 1, 2022
This paper is an extended abstract of a paper with the title "A Query Evaluation Method for Abductive Logic Programming" that appeared in the Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP'92), pp. 671 – 685.
ksatoh@nii.ac.jp (K. Satoh)

[1] Consistency of an abductive framework means that there is a generalized stable model for the framework.
[2] $\sim$ expresses negation as failure

$man(noel) \leftarrow$ (1)
$barber(noel) \leftarrow$ (2)
$shaves(noel, X) \leftarrow man(X), \sim shaves(X, X)$ (3)
$shaves(X, X) \leftarrow barber(X), normal\_barber^{\dagger}(X)$ (4)
$shaves(casanova, X) \leftarrow barber(X), \sim normal\_barber^{\dagger}(X)$ (5)

Since $barber(noel)$ is true from (2), $shaves(casanova, noel)$ will be derived by using Kakas and Mancarella's proof procedure and assuming $\sim normal\_barber^{\dagger}(noel)$ in (5). However, actually, there is no generalized stable model which satsifies $shaves(casanova, noel)$. We solve this problem by using the above techniques.

2. A Semantics of Abductive Framework

We mainly follow the definition of abductive framework in [Kakas90a], but we modify it slightly for notational conveniences. Firstly, we define a rule and an integrity constraint.

Definition 1. *Let H be an atom, and $L_1, ..., L_m (m \geq 0)$ be literals each of which is an atom or a negated atom of the form $\sim B$. A rule is of the form:*

$$H \leftarrow L_1, L_2, ..., L_m.$$

We call H the *head* of the rule and $L_1, ..., L_m$ the *body* of the rule. Let R be a rule. $head(R)$, $body(R)$ and $pos(R)$ denote the head of R, the set of literals in the body of R and the set of positive literals in $body(R)$ respectively.

Definition 2. *Let $L_1, ..., L_m (m \geq 0)$ be literals. An integrity constraint is of the form:*

$$\bot \leftarrow L_1, L_2, ..., L_m.$$

For a given program (with integrity constraints), we define a stable model (in other words, answer set) as follows.

Definition 3. *Let T be a logic program and Π_T be a set of ground rules obtained by replacing all variables in each rule in T by every element of its Herbrand universe. Let M be a set of ground atoms from Π_T and Π_T^M be the following (possibly infinite) program.*

$$\Pi_T^M = \{H \leftarrow B_1, ..., B_k | \ H \leftarrow B_1, ..., B_k, \sim A_1, ..., \sim A_m \in \Pi_T \text{ and } A_i \notin M \text{ for each } i = 1, ..., m.\}$$

Let $min(\Pi_T^M)$ be the least model of Π_T^M. A stable model *for a logic program T is M iff $M = min(\Pi_T^M)$ and $\bot \notin M$.*

This definition gives a stable model of T which satisfies all integrity constraints. We say that T is *consistent* if there exists a stable model for T.

Now, we define an *abductive framework*.

Definition 4. *An* abductive framework *is a pair $\langle T, A \rangle$ where A is a set of predicate symbols, called* abducible predicates *and T is a set of rules each of whose head is not in A.*

We call a set of all ground atoms for predicates in A *abducibles.* As pointed out in [Kakas90a], we can translate a program which includes a definition of abducibles to an equivalent framework that satisfies the above requirement. Moreover, we impose an abductive framework to be range-restricted, that is, any variable in a rule of a program must occur in non-abducible positive literals of the rule.

Now, we define a semantics of an abductive framework.

Definition 5. *Let $\langle T, A\rangle$ be an abductive framework and Θ be a set of abducibles. A* generalized stable model $M(\Theta)$ *is a stable model of $T \cup \{H \leftarrow | H \in \Theta\}$.*

We say that $\langle T, A\rangle$ is *consistent* if there exists a generalized stable model $M(\Theta)$ for some Θ.

3. Query Evaluation for Abduction

Before showing our query evaluation method, we need the following definitions. Let l be a literal. Then, $\bar{l}$ denotes the complement of l[3]

Definition 6. *Let T be a logic program. A set of resolvents w.r.t. a ground literal l and T, $resolve(l, T)$ is the following set of rules:*

$resolve(l, T) =$
$\{(\bot \leftarrow L_1, ..., L_k)\theta |$ *l is negative and*
$H \leftarrow L_1, ..., L_k \in T$ *and* $\bar{l} = H\theta$ *by a ground substitution* $\theta\} \cup$
$\{(H \leftarrow L_1, ..., L_{i-1}, L_{i+1}, ..., L_k)\theta |$
$H \leftarrow L_1, ..., L_k \in T$ *and* $l = L_i\theta$ *by a ground substitution* $\theta\}$

The first set of resolvents are for negation as failure and the second set of resolvents corresponds with "forward" evaluation of the rule introduced in [Sadri88].

Definition 7. *Let T be a logic program. A set of deleted rules w.r.t. a ground literal l and T, $del(l, T)$, is the following set of rules:*

$del(l, T) = \{(H \leftarrow L_1, ..., L_k)\theta |$
$H \leftarrow L_1, ..., L_k \in T$ *and* $\bar{l} = L_i\theta$ *by a ground substitution* $\theta\}$

Our query evaluation procedure consists of 4 subprocedures, $derive(p, \Delta)$, $literal\_con(l, \Delta)$, $rule\_con(R, \Delta)$ and $deleted\_con(R, \Delta)$ where p is a non-abducible atom and Δ is a set of ground literals already assumed and l is a ground literal and R is a rule.

$derive(p, \Delta)$ returns a ground substitution for the variables in p and a set of ground literals. This set of ground literals is a union of Δ and literals newly assumed during execution of the subprocedure. Other subprocedures return a set of ground literals.

The subprocedures have a **select** operation and a **fail** operation. The **select** operation expresses a nondeterministic choice among alternatives. The **fail** operation expresses immediate termination of an execution with failure. Therefore, a subprocedure succeeds when its inner calls of subprocedures do not encounter **fail**. We say *a subprocedure succeeds with (θ and) Δ* when the subprocedure successfully returns (θ and) Δ.

The informal specification of the 4 subprocedures is as follows.

[3] $\overline{p} = \sim p$ for a positive literal p, and $\overline{\sim p} = p$ for a negative literal $\sim p$.

1. $derive(p, \Delta)$ searches a rule R of p in a program T whose body can be made true with a ground substitution θ under a set of assumptions Δ. To show that every literal in the body can be made true, we call $derive$ for non-abducible positive literals in the body. Then, we check the consistency of other literals in the body with T and Δ. Note that because of the range-restrictedness, other literals in R become ground after all the calls of $derive$ for non-abducible positive literals.
2. $literal\_con(l, \Delta)$ checks the consistency of a ground literal l with T and Δ. To show the consistency for assuming l, we add l to Δ; then, we check the consistency of resolvents and deleted rules w.r.t. l and T.
3. $rule\_con(R, \Delta)$ checks the consistency of a rule R with T and Δ. If R is not ground, we must check the consistency for ground instances of R. But it is sufficient to consider every ground instance $R\theta$ in $\Omega_{T\cup\{R\}}$. We can prove the consistency by showing that either a literal in $body(R\theta)$ can be falsified or $body(R\theta)$ can be made true and $head(R\theta)$ consistent.
   This procedure can also be used to check integrity for rule addition.
4. $deleted\_con(R, \Delta)$ checks if a deletion of R does not cause any contradictions with T and Δ. To show the consistency of the implicit deletion of R, it is sufficient to prove that the head of every ground instance $R\theta$ in Ω_T [4] can be made either true or false.

Thanks to the range-restrictedness, we can compute all ground instances of a rule R (if they are finite) in Ω_T (or $\Omega_{T\cup\{R\}}$). For this, we compute every SLD derivation of a query which consists of all non-abducible positive literals in $body(R)$ to the abducible-and-negation-removed program T^- (or $(T \cup \{R\})^-$).

Please see the details of each subprocedure at Figure 1 and Figure 2 in Appendix. In Figure 1, ε denotes empty substitution and $\theta_i\sigma_i$ expresses a composition of two substitutions θ_i and σ_i. Also, we denote a set of non-abducible positive literals, non-abducible negative literals, and abducibles (either negative or positive) in a rule R as $pos(R)$, $neg(R)$ and $abd(R)$.

If we remove $deleted\_con$ and do not consider resolvents obtained with "forward" evaluation of the rule, then this procedure coincides with that of Kakas and Mancarella [Kakas90a]. That is, our procedure is obtained by augmenting their procedure with an integrity constraint checking in a bottom-up manner and with an implicit deletion checking.

We can show the following theorems for correctness of successful derivation, consistency check for addition of a rule and finite failure.

Theorem 1. *Let $\langle T, A\rangle$ be a consistent abductive framework. Suppose $derive(p, \{\})$ succeeds with (θ, Δ). Then, there exists a generalized stable model $M(\Theta)$ for T such that Θ includes all positive abducibles in Δ and $M(\Theta) \models p\theta$.*

This theorem means that if the procedure $derive(p, \{\})$ answers "yes" with (θ, Δ), then there is a generalized stable model which satisfies $p\theta$. However, we cannot say in general that we make $p\theta$ true only with positive abducibles in Δ, because there might be some hypotheses which are irrelevant to a query but which we must assume to get consistency.

The following theorem states correctness for consistency for addition of a rule.

[4] Note that $\Omega_{T\cup\{R\}} = \Omega_T$ since R is an instance of a rule in T.

Theorem 2. *Let $\langle T, A\rangle$ be a consistent abductive framework. Suppose $rule\_con(R, \{\})$ succeeds, then $\langle T \cup R, A\rangle$ is a consistent abductive framework.*

This means that this procedure can be used to guarantee the existence of a generalized stable model for a given abductive framework $\langle T, A\rangle$ if iterative calls of $rule\_con$ succeed from $\langle \{\}, A\rangle$ by adding each rule in T.

The following is a theorem related to correctness for finite failure.

Theorem 3. *Let $\langle T, A\rangle$ be an abductive framework. Suppose that every selection of rules terminates for $derive(p, \{\})$ with either success or failure. If there exists a generalized stable model $M(\Theta)$ for $\langle T, A\rangle$ and a ground substitution θ such that $M(\Theta) \models p\theta$, then there is a selection of rules such that $derive(p, \{\})$ succeeds with (θ, Δ) where Θ includes all positive abducibles in Δ.*

This theorem means that if we can search exhaustively in selecting the rules and there is a generalized stable model which satisfies a query, then the procedure always answers "yes".

With this theorem, we obtain the following corollary for a finite failure.

Corollary 1. *Let $\langle T, A\rangle$ be an abductive framework. If $derive(p, \{\})$ fails, then for every generalized stable model $M(\Theta)$ for $\langle T, A\rangle$ and for every ground substitution θ, $M(\Theta) \not\models p\theta$.*

When the procedure $derive(p, \{\})$ answers "no", there is no generalized stable model which satisfies the query. Also, this corollary means that we can use a finite failure to check if the negation of a ground literal is true in all generalized stable models since finite failure of $derive(p, \{\})$ means that every generalized stable model satisfies $\sim p$.

4. Conclusion

In this paper, we propose a query evaluation method for an abductive framework. Our procedure can be regarded as an extension of the procedure of Kakas and Mancarella by augmenting forward evaluation of rules and consistency check for implicit deletion.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers, JP17H06103 and JP19H05470 and JST, AIP Trilateral AI Research, Grant Number JPMJCR20G4.

References

[Kakas90a] Kakas, A. C., Mancarella, P., Generalized Stable Models: A Semantics for Abduction, *Proc. of ECAI'90*, pp. 385 – 391 (1990).

[Kakas90b] Kakas, A. C., Mancarella, P., On the Relation between Truth Maintenance and Abduction, *Proc. of PRICAI'90*, pp. 438 – 443 (1990).

[Sadri88] Sadri, F., Kowalski, R., A Theorem-Proving Approach to Database Integrity, *Foundations of Deductive Database and Logic Programming*, (J. Minker, Ed.), Morgan Kaufmann Publishers, pp. 313 – 362 (1988).

Appendix: Topdown Proof Procedure

$derive(p, \Delta)$ p: a non-abducible atom; Δ: a set of literals
begin
  if p is ground and $p \in \Delta$ **then return** (ε, Δ)
  elseif p is ground and $\sim p \in \Delta$ **then fail**
  else
  begin
    select $R \in T$ s.t. $head(R)$ and p are unifiable with an mgu θ
    if such a rule is not found **then fail**
    $\Delta_0 := \Delta$, $\theta_0 := \theta$, $B_0 := pos(R\theta)$, $i := 0$
    while $B_i \neq \{\}$ **do**
    begin
      take a literal l in B_i
      if $derive(l, \Delta_i)$ succeeds with (σ_i, Δ_{i+1})
        then $\theta_{i+1} := \theta_i\sigma_i$, $B_{i+1} := (B_i - \{l\})\sigma_i$, $i := i + 1$ and **continue**
    end
    $\delta := \theta_i$
    for every $l \in neg(R\delta) \cup abd(R\delta)$ **do**
    begin
      if $literal\_con(l, \Delta_i)$ succeeds with Δ_{i+1}
        then $i := i + 1$ and **continue**
    end
    if $literal\_con(p\delta, \Delta_i)$ succeeds with Δ' **then return** (δ, Δ')
  end
end ($derive$)

$literal\_con(l, \Delta)$ l: a ground literal; Δ: a set of literals
begin
  if $l \in \Delta$ **then return** Δ
  elseif $l = \bot$ or $\bar{l} \in \Delta$ **then fail**
  else
  begin
    $\Delta_0 := \{l\} \cup \Delta$, $i := 0$
    for every $R \in resolve(l, T)$ **do**
      if $rule\_con(R, \Delta_i)$ succeeds with Δ_{i+1}
        then $i := i + 1$ and **continue**
    for every $R \in del(l, T)$ **do**
      if $deleted\_con(R, \Delta_i)$ succeeds with Δ_{i+1}
        then $i := i + 1$ and **continue**
  end
  return Δ_i
end ($literal\_con$)

Figure 1: The definition of $derive$ and $literal\_con$

$rule\_con(R, \Delta)$ R: a rule; Δ: a set of literals
begin
  $\Delta_0 := \Delta, i := 0$
  for every ground rule $R\theta \in \Omega_{T \cup \{R\}}$ **do**
  begin
    select (a) or (b)
    (a) **select** $l \in body(R\theta)$
      if $l \in pos(R\theta) \cup abd(R\theta)$ and $literal\_con(\bar{l}, \Delta_i)$ succeeds with Δ_{i+1}
        then $i := i + 1$ and **continue**
      elseif $l \in neg(R\theta)$ and $derive(\bar{l}, \Delta)$ succeeds with $(\varepsilon, \Delta_{i+1})$
        then $i := i + 1$ and **continue**
    (b) $\Delta_i^0 := \Delta_i, j := 0$
      for every $l \in body(R\theta)$ **do**
      begin
        if $l \in pos(R\theta)$
          and $derive(l, \Delta_i^j)$ succeeds with $(\varepsilon, \Delta_i^{j+1})$
          then $j := j + 1$ and **continue**
        elseif $l \in neg(R\theta) \cup abd(R\theta)$
          and $literal\_con(l, \Delta_i^j)$ succeeds with Δ_i^{j+1}
          then $j := j + 1$ and **continue**
      end
      if $literal\_con(head(R\theta), \Delta_i^j)$ succeeds with Δ_{i+1}
        then $i := i + 1$ and **continue**
  end
  return Δ_i
end ($rule\_con$)

$deleted\_con(R, \Delta)$ R: a rule; Δ: a set of literals
begin
  $\Delta_0 := \Delta, i := 0$
  for every ground rule $R\theta \in \Omega_T$ **do**
  begin
    select (a) or (b)
    (a) **if** $derive(head(R\theta), \Delta_i)$ succeeds with $(\varepsilon, \Delta_{i+1})$
      then $i := i + 1$ and **continue**
    (b) **if** $literal\_con(\sim head(R\theta), \Delta_i)$ succeeds with Δ_{i+1}
      then $i := i + 1$ and **continue**
  end
  return Δ_i
end ($deleted\_con$)

Figure 2: The definition of $rule\_con$ and $deleted\_con$