"All's Well that Ends Well" – a Proposal of Global Abduction –

Ken Satoh
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
Email: ksatoh@nii.ac.jp

Abstract

This paper presents a new form of abduction called *global abduction*. Usual abduction in logic programming is used to complement unknown information and used in one derivation path in a search tree. We call this kind of abduction *local abduction*. In this paper, we propose another abduction which is used over paths in a search tree for search control. As far as we know, this is the first attempt to formalize a search control in a logical way. We discuss applications of global abduction by using examples; a formalization of don't-care nondeterminism and a formalization of reuse of the previously obtained result in a different search path. Then, we give a correct proof procedure for global abduction. The correctness is defined as *"all's well that ends well" principle* meaning that the results obtained from a global abduction proof procedure are exactly the same as the ones which are logically true from the augmented program with the last set of abduced atoms.

Introduction

In the previous abductive logic programming framework (Kakas and Mancarella, 1990; Denecker and De Schreye, 1998), we use abduction to complement unknown information when the derivation of a goal needs such hypotheses. However, these hypotheses are valid only in the derivation path to achieve the goal and there is no effect to the other search path. We call this type of abduction *local abduction*.

In this paper, we propose another type of abduction which can be accessed from any search path and used to control the search. We sometimes want to pass information which is obtained in one derivation path to another path for search control. For example, to express don't care nondeterminism where we commit only one derivation, we need to tell the other derivations that one derivation is already selected. In a formalism of concurrent logic languages such as (Saraswat, 1993), meta-control "commitment operator" is introduced to a program, but there is no logical semantics for the "commitment operator". As another example, we use a tabling technique to keep a result obtained in one derivation which will be used for other derivations. However, to implement the technique, we can use "assert" command, but there is no logical semantics for the command so far. In this paper, we propose an abductive method called *global abduction* in order to formalize information passing over derivation paths. As far as we know, this is the first attempt to formalize a search control in a logical way. Moreover, we believe that global abduction is useful in cognitive robotics (Cognitive Robotics Workshop, 2002). In cognitive robotics, we perform planning under incomplete information and execute actions with side-effect in an interleaving manner. Suppose that we execute a plan, but fail because of some change of outside environments. Then, we need to revise our plan if possible, but in order to make a new plan, we need to consider the previous side-effects caused by some actions in the previous plan. If we tried to implement this revision of planning in a usual backtracking mechanism used in Prolog system, we would not be able to pass the information of such side-effects in other alternatives. However, we can use global abduction to pass these information in our framework.

For global abduction, we introduce two annotations, `announce`(P) and `hear`(P). `announce`(P) asserts a ground literal P in the global belief state. After `announce`(P) is executed, P becomes true globally as if it were asserted in the beginning of the execution. This means P is abduced not only in the search path where the announcement is done, but also propagated to the other search paths as if it were true in the beginning. `hear`(P) is used to access a global belief state. If P is asserted in the global belief state, `hear`(P) is succeeded. If P is not asserted yet or P is contradictory with the current belief state, then the execution suspends and other derivation path will be traversed. Moreover, we keep a track on the hearing literal so that if the global belief is changed and the internal belief used by a process contradicts the new global belief, the process will be suspended.

In this paper, we propose a semantics w.r.t. these annotations. In the semantics, we require that the obtained result from global abduction must be logically derived from a program which is obtained by adding the assumed annotated atoms to the original program. Therefore, we call the semantics in the paper *"all's well that ends well" principle* (*AWW principle* for short) since correctness is formalized w.r.t. the original program plus the last belief state.

The structure of the paper is as follows. We firstly give a framework of global abduction and then discuss applications of global abduction; a formalization of don't-care nondeterminism and a formalization of reuse of the different search path. Then, we give a correct proof procedure for global abduction and examples of execution by the proof procedure.

Framework of Global Abduction

Definition 1 *A* global abductive framework *GA is the following tuple $\langle \mathcal{B}, \mathcal{P}, \mathcal{IC} \rangle$ where*

- *$\mathcal{B}$ is a set of predicates called* belief predicates. *Let A be an atom with belief predicate. We call A a* positive belief literal, *or a* belief atom *and $\neg A$ a* negative belief literal. *We call a literal of the above form* belief literal. *Let Q be an belief literal. We introduce annotated literals* announce(Q) *and* hear(Q) *called* announcing literal *and* hearing literal *respectively. We say that* announce(Q) *contains Q and* hear(Q) *contains Q.*
- *$\mathcal{P}$ is a set of rules of the form:*

$$H :- B_1, B_2, ..., B_n.$$

where

  - *H is an* ordinary *atom, in other words, an atom with a predicate which is neither a belief predicate nor an equality predicate.*
  - *each of $B_1, ..., B_n$ is an ordinary atom, or an annotated literal, or an equality literal of the form $t = s$, or a disequality literal of the form $t \neq s$.*

  We call H a head denoted as $head(R)$ and $B_1, ..., B_n$ a body denoted as $body(R)$. If there are no atoms in the body, $body(R) = \emptyset$.
- *$\mathcal{IC}$ is a set of integrity constraints of the form:*

$$\texttt{false} :- B_1, B_2, ..., B_n.$$

where false *expresses falsity and each of $B_1, ..., B_n$ is a belief literal or an equality literal or a disequality literal.*

Integrity constraints are used to control announcing literals. Intuitively, annotated literals have the following meaning.

- Announcing literal announce(L) is an assertion of a ground positive/negative belief L to the global belief state. This means while we traverse a search space to achieve goal, some of the fact are added by the program itself. Then, from another search path, we can access this addition using a hearing literal. Therefore, L is "globally" abduced by announce(L). Moreover, due to the integrity constraint, we sometimes fail to announce L.
- Hearing literal hear(L) is a check of a ground positive/negative belief L in the current belief state. If L is included in the current belief state hear(L) coincides with the truth value of L w.r.t. the current belief state. Else if L is not included in the current belief state, the truth value of hear(L) is undefined and the evaluation is postponed. According to an update of the belief state, hear(L) changes its truth value according to the current belief state.

In the following examples, we show applications of global abduction.

Example 1 *The following predicate $merge(X, Y, Z, Pid)$ means a don't-care nondeterministic merge of two list X and Y resulting a unique merged list Z.*

The example can be formalized as the following global abductive framework $GA = \langle \mathcal{B}, \mathcal{P}, \mathcal{IC} \rangle$ where

- *$\mathcal{B} = \{$*cut*$\}$ where* cut(Pid, RN) *means that the goal with a goal identification term Pid is unified with the RN-th rule. The goal identification term, 1, is attached as an additional argument in the top level goal and the other identification term for an ordinary literal in the body of a rule is represented as* c(Pid, RN, LN) *where Pid is the goal identification term for the parent goal and RN is the rule number of the rule containing the ordinary literal and LN is the literal number of the literal in the body of the rule which is used in order to construct a unique Pid term for each call in a rule.*
- *$\mathcal{IC} = \{$*false :- cut(Id, R1), cut(Id, R2), R1 $\neq$ R2.$\}$
- *$\mathcal{P}$ is the following program:*

```
merge(X,Y,Z,Pid):-
    X=[A|X1],
    announce(cut(Pid,1)),
    Z=[A|Z1],
    merge(X1,Y,Z1,c(Pid,1,1)).
merge(X,Y,Z,Pid):-
    Y=[A|Y1],
    announce(cut(Pid,2)),
    Z=[A|Z1],
    merge(X,Y1,Z1,c(Pid,2,1)).
merge(X,Y,Z,Pid):-
    X=[ ],
    announce(cut(Pid,3)),
    Z=Y.
merge(X,Y,Z,Pid):-
    Y=[ ],
    announce(cut(Pid,4)),
    Z=X.
```

We add one extra argument Pid for each merge call in order to ensure that each call is identified by using Pid and RN.

We assume that all the rules whose head unify with the current goal are selected in a parallel manner and that the execution of the body is done from the left to the right. Then, the above program only gives one merge result because of the announcing literal announce(cut(...)) *and the above integrity constraint.* □

We can combine announcing literal and hearing literal as the following example shows. The example avoids a redundant check in the alternative path using announcement of the already obtained result of the check.

Example 2 *Consider the following global abductive framework $GA = \langle \mathcal{B}, \mathcal{P}, \mathcal{IC} \rangle$ where*

- $\mathcal{B} = \{\texttt{cut}, \texttt{alreadychecked}\}$ *where* `cut(...)` *is the same as the previous example and* $\texttt{alreadychecked}(X)$ *is the information that instance X is already checked by time-consuming predicate* `timeconsumingcheck(X)` *in the different search path.*
- $\mathcal{IC} = \{\texttt{false}\text{:} -\texttt{cut}(\texttt{Id}, \texttt{R1}), \texttt{cut}(\texttt{Id}, \texttt{R2}), \texttt{R1} \neq \texttt{R2}.\}$
- $\mathcal{P}$ *is the following program:*

```
main(X,Y,Pid):-
    q(X,Y),
    check(X,c(Pid,1,1)).
main(X,Y,Pid):-
    r(X,Y),
    check(X,c(Pid,2,1)).
q(a,b).
r(a,c).
check(X,Pid):-
    hear(alreadychecked(X)),
    announce(cut(Pid,1)).
check(X,Pid):-
    announce(cut(Pid,2)),
    timeconsumingcheck(X),
    announce(alreadychecked(X)).
```

Suppose that we ask `main(X,Y,1)`*. We assume that the execution of the body is done from left to right. Suppose that we have obtained* `X=a,Y=b` *from the first rule of* `main` *and during computation of the first answer, we made a time-consuming check. Then, suppose that we start an execution of the second rule of* `main`*. Then, thanks to the announcement of the check result for the first answer, the second answer (*`X=a,Y=c`*) is obtained without a time-consuming process.* □

Semantics

We give a semantics of the global abduction based on the extension of the three-valued least model semantics (Fitting, 1985; Przymusinski, 1990). We use the three-valued semantics since the truth value of the belief literals can be undefined when the current belief state does not decide their truth values. We extend the three-valued least model so that the semantic of the global abduction is indexed w.r.t. a belief state.

In the following, we basically follow the way of the definitions in (Przymusinski, 1990). Firstly, we define the three-valued interpretation and the three-valued least model semantics (Fitting, 1985). We denote **t** as truth, **f** as falsity, and **u** as undefinedness. We say a rule is *ordinary* if a rule consists of ordinary atoms, or equality atoms, or disequality atoms. Similarly, we say that a program is *ordinary* if a program consists of ordinary rules.

Definition 2 *A three-valued Herbrand interpretation I of an ordinary program P is any pair $\langle T; F \rangle$ where T and F are disjoint subsets of the Herbrand base for P. We define $\texttt{TRUE}(I)$ as T and $\texttt{FALSE}(I)$ as F. The truth value of a ground atom A w.r.t. I, $val(A, I)$, is defined as follows:*

- *If A is an equality atom $t = s$, $val(A, I) = \mathbf{t}$ if and only if t and s are identical terms.*
- *If A is a disequality atom $t \neq s$, $val(A, I) = \mathbf{t}$ if and only if t and s are not identical terms.*
- *If A is an ordinary atom*
  $val(A, I) = \mathbf{t}$ *if* $A \in \texttt{TRUE}(I)$
  $val(A, I) = \mathbf{f}$ *if* $A \in \texttt{FALSE}(I)$
  $val(A, I) = \mathbf{u}$ *otherwise.*

Definition 3 *Let R be a ground ordinary rule of the form*

$$H\text{:} -B_1, B_2, ..., B_n.$$

R is satisfied w.r.t. a three-valued interpretation I if one of the following condition is satisfied.

- *There is an atom B_i in $body(R)$ such that $val(B_i, I) = \mathbf{f}$.*
- *For every atom B_i in $body(R)$, $val(B_i, I) = \mathbf{t}$ and $val(H, I) = \mathbf{t}$.*
- *There is an atom B_i in $body(R)$ such that $val(B_i, I) = \mathbf{u}$ and for every other atom B_j in $body(R)$ other than B_i, $val(B_j, I) = \mathbf{t}$ or $val(B_j, I) = \mathbf{u}$ and $val(H, I) = \mathbf{t}$ or $val(H, I) = \mathbf{u}$.*

Definition 4 *We say that a three-valued interpretation I is a model of an ordinary program P if every ground instance of every rule in P is satisfied w.r.t. I.*

Let I be a model of P. We say that a model is the least if there is no model J of P such that $I \neq J$ and $\texttt{TRUE}(I) \supseteq \texttt{TRUE}(J)$ and $\texttt{FALSE}(I) \subseteq \texttt{FALSE}(J)$.

Note that thanks to the result of (Przymusinski, 1990), for every program consisting of ordinary atom, P has the least model.

Now, we define an *assumption-based three-valued model* for a global abductive framework $\langle \mathcal{B}, \mathcal{P}, \mathcal{IC} \rangle$. An assumption-based three-valued model is defined by augmenting ordinary least model with the interpretation of annotated literals.

Definition 5 *Let BS be a pair of disjoint subsets of ground belief atoms, $\langle BS_T; BS_F \rangle$. We call BS a* belief state*. We define $\texttt{TRUE}(BS)$ as BS_T and $\texttt{FALSE}(BS)$ as BS_F.*
For a belief literal L, we say that L is true *in BS if one of the following conditions is satisfied.*

- *If L is a positive literal then $L \in TRUE(BS)$.*
- *If L is a negative literal of the form $\neg L'$ then $L' \in FALSE(BS)$.*

For a belief literal L, we say that L is false *in BS if one of the following conditions is satisfied.*

- *If L is a positive literal then $L \in FALSE(BS)$.*

- *If L is a negative literal of the form $\neg L'$ then $L' \in TRUE(BS)$.*

For a belief literal L, we say that L is undefined in *BS if L is neither true in BS nor false in BS.*

We also need evaluation of equality atoms and disequality atoms in a belief state for evaluation of integrity constraints. But, this evaluation is actually independent on a belief state.

Definition 6 *Let BS be a belief state and L be an equality atom or a disequality atom. we say that L* is true in *BS if one of the following conditions is satisfied otherwise we say that L* is false in *BS.*

- *If L is an equality atom $t = s$, then t and s are identical terms.*
- *If L is a disequality atom $t \neq s$, then t and s are not identical terms.*

For the evaluation of rules, we need the following evaluation of annotated literals.

Definition 7 *Let BS be a belief state and L be an annotated literal. We define the truth value of L w.r.t. BS, $val(L, BS)$, as follows. Let L be of the form either $\texttt{hear}(L')$ or $\texttt{announce}(L')$.*

- *$val(L, BS) = \mathbf{t}$ if L' is true in BS.*
- *$val(L, BS) = \mathbf{f}$ if L' is false in BS.*
- *$val(L, BS) = \mathbf{u}$ if L' is undefined in BS.*

From now on, we introduce special atoms `true`, and `undef` as well as `false`. For every interpretation I, we assume that $\texttt{TRUE}(I)$ always contains `true` and $\texttt{FALSE}(I)$ always contains `false` and neither $\texttt{TRUE}(I)$ nor $\texttt{FALSE}(I)$ contains `undef`. This means that `true` (`false`, `undef`, respectively) always has the truth value of $\mathbf{t}$ ($\mathbf{f}$, $\mathbf{u}$, respectively).

For a program $\mathcal{P}$ and a set of integrity constraints $\mathcal{IC}$, we denote ground program of $\mathcal{P}$ and a set of ground integrity constraints of $\mathcal{IC}$ as $\Pi_{\mathcal{P}}$ and $\Pi_{\mathcal{IC}}$ respectively.

Let BS be a belief state. Then for a program $\mathcal{P}$, we define $\Pi_{\mathcal{P}}/BS$ as the following program:

$$\Pi_{\mathcal{P}}/BS = \{H{:}\,-tr(B_1, BS), ..., tr(B_n, BS) \mid H{:}\,-B_1, ..., B_n \in \Pi_{\mathcal{P}}\}$$

where

- if B_i is an ordinary atom, or an equality atom or a disequality atom, $tr(B_i, BS) = B_i$.
- if B_i is an annotated literal,
  - $tr(B_i, BS) = \texttt{true}$ if $val(B_i, BS) = \mathbf{t}$.
  - $tr(B_i, BS) = \texttt{false}$ if $val(B_i, BS) = \mathbf{f}$.
  - $tr(B_i, BS) = \texttt{undef}$ if $val(B_i, BS) = \mathbf{u}$.

Please note that the least 3-valued model of an ordinary program without any belief literal is the same as the least 2-valued model, but if we introduce a belief literal into a program, then the interpretation of the belief literal is 3-valued and therefore, we need to consider the least 3-valued model for such a program.

Definition 8 *Let GA be a global abductive framework $\langle \mathcal{B}, \mathcal{P}, \mathcal{IC} \rangle$ and BS be a belief state and I be a pair of disjoint subsets of ground ordinary atoms defined from the Herbrand term of $\mathcal{P}$.*

I is an assumption-based three-valued model *w.r.t. $\mathcal{P}$, $\mathcal{IC}$ and BS if the following conditions are satisfied.*

- *There is no integrity constraint "$\texttt{false}{:}\,-L_1, ..., L_n$"$\in \Pi_{\mathcal{IC}}$ such that for every literal L_i, L_i is true in BS.*
- *I is the least three-valued model of $\Pi_{\mathcal{P}}/BS$.*

The above semantics has the following intuitive meaning. Firstly, the truth values of an announcing literal and a hearing literal are defined by the belief state and then we recursively propagate the truth value using rules as follows. Also note that if the body of an integrity constraint is satisfied with the current BS, then there is no model.

- If there is a rule such that every literal in the body is true, then the head of the rule, H, becomes true.
- If there is a rule for an ordinary atom H such that there is a literal L in its body whose truth value is undefined and every other literal in its body has the truth value of $\mathbf{t}$ or $\mathbf{u}$, then H becomes undefined.
- Otherwise H becomes false.

Proof Procedure

In this section we give a correct proof procedure which is correct in the above semantics. The execution of global abductive framework is based on *process reduction.* Intuitively, processes are created when a choice point of computation is encountered like case splitting. A process terminates successfully if all the computation is done and the belief literals used in the process are not contradictory with the last belief state. As the subsequent theorem shows, if we reflect the last belief state BS to the program $\mathcal{P}$ by considering $\mathcal{P}/BS$, then the same result is obtained. Therefore, we call this principle *"all's well that ends well"* principle.

In the procedure, we reduce an active process into a new process. Reduction for an ordinary atom is a usual goal reduction in logic programming and reduction for an announcing literal corresponds with an update of the belief state and reduction for a hearing literal corresponds with an inquiry to the belief state.

Updating the belief state by an announcing literal may result in the suspension of the current executed process and change of the execution to an alternative process.

Preliminary Definitions

We introduce the following for explanation of the proof procedure.

Definition 9 *A* process *is the following tuple $\langle GS, BA, ANS \rangle$ which consists of*

- *GS: A set of atoms to be proved called a* goal set.

- *BA: A set of ground belief literals called* belief assumptions.
- *ANS: A set of instantiations of variables in the initial query.*

A process expresses an execution status in a path in the search tree. The intuitive meaning of the above objects is as follows:

- GS expresses the current status of computation.
- BA is a set of belief assumptions used during a process.
- ANS gives an answer for variables in the initial query.

We use the following two sets for process reduction.

Definition 10

- A process set PS *is a set of processes.*
- A current belief state CBS *is a belief state.*

PS is a set of processes which express all the alternative computations considered so far, and CBS is the current belief state which expresses the agent's current belief.

Definition 11 *Let $\langle GS, BA, ANS\rangle$ be a process and CBS be a current belief state. A process is* active w.r.t. CBS *if for every $L \in BA$, L is true in CBS and a process is* suspended w.r.t. CBS *otherwise.*

If BA contradicts CBS, the execution of process is considered to be useless at the current belief state and therefore, the process will be suspended.

Description of Proof Procedure

In the following reduction, we specify only changed PS, CBS as $NewPS, NewCBS$; otherwise each PS, CBS is unchanged.

Initial Step: Let GS be the initial goal set.
We give $\langle GS, \emptyset, ANS\rangle$ to the proof procedure where ANS is a set of variables in GS. That is, $PS = \{\langle GS, \emptyset, ANS\rangle\}$. and let CBS be the initial set of belief literals.

Iteration Step: Do the following.

Case 1 If there is an active process $\langle \emptyset, BA, ANS'\rangle$ w.r.t. CBS in PS, return instantiation for variables ANS' and the current belief state CBS.

Case 2 If PS is empty, return "failure".

Case 3 Select an active process $\langle GS, BA, ANS\rangle$ w.r.t. CBS from PS and select a literal L (an ordinary atom, or an equality atom, or a disequality atom, or an annotated literal) in GS which satisfies one of the conditions in the following subcase. If there is no such process, return "floundering". Let $PS' = PS - \{\langle GS, BA, ANS\rangle\}$ and $GS' = GS - \{L\}$.

Case 3.1 If L is an ordinary atom,

$$NewPS = PS' \cup \{\langle(\{body(R)\} \cup GS')\theta, BA, ANS\theta\rangle \mid R \in \mathcal{P} \text{ and } \exists \text{most general unifier(mgu) } \theta \text{ s.t. } head(R)\theta = L\theta\}$$

Case 3.2 If L is an equality atom $t = s$,

Case 3.2.1 if there is an mgu θ between t and s, then $NewPS = PS' \cup \{\langle GS'\theta, BA, ANS\theta\rangle\}$

Case 3.2.2 else if there is no such mgu, then $NewPS = PS'$.

Case 3.3 If L is an disequality atom $t \neq s$, and t and s is ground,

Case 3.3.1 if t and s are different ground terms then $NewPS = PS' \cup \{\langle GS', BA, ANS\rangle\}$

Case 3.3.2 else if t and s are identical terms, then $NewPS = PS'$.

Case 3.4 If L is a hearing literal `hear`(Q) and Q is ground and Q is in CBS, then $NewPS = PS' \cup \{\langle GS', BA \cup \{Q\}, ANS\rangle\}$.

Case 3.5 If L is an announcing literal `announce`(A) and A is ground and there is no integrity constraint "`false`: $-L_1, ..., L, ..., L_n$"$\in \Pi_{\mathcal{IC}}$ containing L such that for every literal L_i other than L, L_i is true in CBS, then $NewPS = PS' \cup \{\langle GS', BA \cup \{A\}, ANS\rangle\}$, and do the following:

- If A is a positive literal, then
  $NewCBS =$
  $\langle \texttt{TRUE}(CBS) \cup \{A\}; \texttt{FALSE}(CBS) \backslash \{A\}\rangle$
- If A is a negative literal of the form $\neg A'$, then
  $NewCBS =$
  $\langle \texttt{TRUE}(CBS) \backslash \{A'\}; \texttt{FALSE}(CBS) \cup \{A'\}\rangle$

Note that a belief state can be updated by announcing literals and some processes becomes active or suspended according to the update. In (Satoh et al., 2000), we provide an efficient implementation of this check. We introduce a list of suspended processes and record the cause of the suspension for each suspended process and every time an update is performed, a suspended process which is relevant to the update will be checked if the suspended process should be resumed or not. We can use this technique for the above procedure for an efficient handling of suspension and resumption of processes, but, in this paper, we present an abstract version for simplicity.

Note that we do not specify any strategy of selection of a literal L. However, the theorem shown in the subsequent section guarantees that if we have an answer from the above proof procedure in any arbitrary strategy, the result is always correct.

Note also that the initial belief state is not necessarily an empty set, but can be any arbitrary set of belief literals.If the set is not empty, belief literals of the initial belief state can be regarded as *default values* of these belief literals. We used this idea to implement *speculative computation* (Satoh et al., 2000; Satoh, 2002; Satoh and Yamamoto, 2002). Therefore, this work can be seen as a formalization of speculative computation as well. The difference between the present paper and the previous research on speculative computation is in that a belief state is updated from outside in the previous research whereas we can use `announce` annotation to update a belief dynamically within a program.

Correctness of the Proof Procedure

The following theorem shows correctness of the above procedure. The theorem intuitively means that when we receive an answer of execution, the answer is correct in the assumption-based three-valued model w.r.t. the program and the returned belief state. This is an idea of *AWW principle*.

Let ANS' be an instantiation of the variables and GS be the initial goal. We write $GS \circ ANS'$ as the goal obtained from GS by replacing variables in GS by corresponding term in ANS'. Let M be the assumption-based three-valued model and $\{L_1, ..., L_n\}$ be a set of ground literals. We write $M \models \{L_1, ..., L_n\}$ if L_i is in M.

Theorem 1 *Let GA be a global abductive framework $\langle \mathcal{B}, \mathcal{P}, \mathcal{IC} \rangle$. Let GS be an initial goal set. Suppose that an instantiation of the variables ANS' and the current belief state CBS are returned. Let M be the assumption-based three-valued model w.r.t. $\mathcal{P}$, $\mathcal{IC}$ and CBS. Then, $M \models GS \circ ANS'$.*

Sketch of proof: By the assumption that ANS' is returned, we can construct a derivation of G from $\mathcal{P}$ and the current belief state CBS. This is done by observing reduction steps in the procedure from $\langle G, \emptyset, ANS \rangle$ to $\langle \emptyset, BA, ANS' \rangle$ where ANS is a set of variables in GS and BA is a set of literals contained in hearing literals which are assumed during the execution.

The derivation corresponds with a derivation defined in (Kowalski et al., 1998) and according to the result of (Kowalski et al., 1998), it is correct for a stratified logic program augmented with the belief state.

Then, we can easily extend the above result to show that every reduction step is correct in terms of assumption-based three-valued model w.r.t. $\mathcal{P}$, $\mathcal{IC}$ and CBS. $\square$

Execution of Global Abduction

Don't-care nondeterminism

Suppose that we ask `merge([1,2],[3,4],Z,1)` to the program in Example 1. The following is an execution trace for the above goal. We abbreviate `merge` as `m` and `announce` as `a`. In the following, we omit the evaluation of equality. We assume that an active process for a reduction is selected randomly. We underline a selected literal in the selected active process and show PS, CBS only when a change occurs.

1. $PS = \{\langle\{\underline{\mathtt{m([1,2],[3,4],Z,1)}}\}, \emptyset, \{\mathtt{Z}\}\rangle\}$.
   $CBS = \langle \emptyset; \emptyset \rangle$.
2. $PS = \{\langle\{\mathtt{a(cut(1,1)), m([2],[3,4],Z_1,c(1,1,1))}\}, \emptyset, \{\mathtt{[1|Z_1]}\}\rangle,$
   $\langle\{\underline{\mathtt{a(cut(1,2))}}, \mathtt{m([1,2],[4],Z_2,c(1,2,1))}\}, \emptyset, \{\mathtt{[3|Z_2]}\}\rangle\}$.
3. $PS = \{\mathtt{P_1},$
   $\langle\{\underline{\mathtt{m([1,2],[4],Z_2,c2)}}\}, \emptyset, \{\mathtt{[3|Z_2]}\}\rangle\}$[1].
   $CBS = \langle\{\mathtt{cut(1,2)}\}; \emptyset\rangle$.
4. $PS = \{\mathtt{P_1},$
   $\langle\{\underline{\mathtt{a(cut(c2,1)))}}, \mathtt{m([2],[4],Z_3,c(c2,1,1))}\}, \emptyset, \{\mathtt{[3,1|Z_3]}\}\rangle,$
   $\langle\{\mathtt{a(cut(c2,2))), m([1,2],[\,],Z_4,c(c2,2,1))}\}, \emptyset, \{\mathtt{[3,4|Z_4]}\}\rangle\}$.
5. $PS = \{\mathtt{P_1},$
   $\langle\{\underline{\mathtt{m([2],[4],Z_3,c21)}}\}, \emptyset, \{\mathtt{[3,1|Z_3]}\}\rangle, \mathtt{P_2}\}$[2].
   $CBS = \langle\{\mathtt{cut(1,2), cut(c2,1)}\}; \emptyset\rangle$.
6. $PS = \{\mathtt{P_1},$
   $\langle\{\underline{\mathtt{a(cut(c21,1))}}, \mathtt{m([\,],[4],Z_5,c(c21,1,1))}\}, \emptyset, \{\mathtt{[3,1,2|Z_5]}\}\rangle,$
   $\langle\{\mathtt{a(cut(c21,2)), m([2],[\,],Z_6,c(c21,2,1))}\}, \emptyset, \{\mathtt{[3,1,4|Z_6]}\}\rangle, \mathtt{P_2}\}\rangle\}$.
7. $PS = \{\mathtt{P_1},$
   $\langle\{\underline{\mathtt{m([\,],[4],Z_5,c211)}}\}, \emptyset, \{\mathtt{[3,1,2|Z_5]}\}\rangle,$
   $\mathtt{P_3}, \mathtt{P_2}\}$[3].
   $CBS = \langle\{\mathtt{cut(1,2), cut(c2,1), cut(c21,1)}\}; \emptyset\rangle$.
8. $PS = \{\mathtt{P_1},$
   $\langle\{\mathtt{a(cut(c211,2)), m([\,],[\,],Z_7,c(c211,2))}\}, \emptyset, \{\mathtt{[3,1,2,4|Z_7]}\}\rangle,$
   $\langle\{\underline{\mathtt{a(cut(c211,3))}}\}, \emptyset, \{\mathtt{[3,1,2,4]}\}\rangle,$
   $\mathtt{P_3}, \mathtt{P_2}\}$.
9. $PS = \{\mathtt{P_1},$
   $\langle\{\mathtt{a(cut(c211,2)), m([\,],[\,],Z_7,c(c211,2))}\}, \emptyset, \{\mathtt{[3,1,2,4|Z_7]}\}\rangle,$
   $\langle \emptyset, \emptyset, \{\mathtt{[3,1,2,4]}\}\rangle,$
   $\mathtt{P_3}, \mathtt{P_2}\}$.
   $CBS = \langle\{$ `cut(1,2),cut(c2,1),` `cut(c21,1),cut(c211,3)` $\}; \emptyset\rangle$.
10. Since $\langle \emptyset, \emptyset, \{\mathtt{[3,1,2,4]}\}\rangle$ is found, $\mathtt{Z} = \mathtt{[3,1,2,4]}$ and $CBS = \langle\{$ `cut(1,2),cut(c2,1),` `cut(c21,1),cut(c211,3)` $\}; \emptyset\rangle$ is returned.

Note that other goals cannot be reduced because of the integrity constraint

$$\mathtt{false}\text{: } -\mathtt{cut(Id,R1)}, \mathtt{cut(Id,R2)}, \mathtt{R1} \neq \mathtt{R2}.$$

It is interesting to note that *cut* literals actually represent a search path to lead to the result of $\mathtt{Z} = \mathtt{[3,1,2,4]}$. So, in some cases, the global abduction would be not only useful for a search control but also for a trace of execution.

Reuse of the result in the different search path

Suppose that we ask `main(X,Y,1)` to the program in Example 2. The following is an execution trace for

[1]We abbreviate a process $\langle\{\mathtt{a(cut(1,1)), m([2],[3,4],Z_1,c1)}\}, \emptyset, \{\mathtt{[1|Z_1]}\}\rangle$ as $\mathtt{P_1}$, and `c(1,2,1)` as `c2`

[2]We abbreviate a process $\langle\{\mathtt{a(cut(c2,2))), m([1,2],[\,],Z_4,c(c2,2,1))}\}, \emptyset, \{\mathtt{[3,4|Z_4]}\}\rangle$ as $\mathtt{P_2}$, and `c(c2,1,1)` as `c21`

[3]We abbreviate `c(c21,1,1)` as `c211`, and a process $\langle\{\mathtt{a(cut(c21,2)), m([2],[\,],Z_6,c212)}\}, \emptyset, \{\mathtt{[3,1,4|Z_6]}\}\rangle, \mathtt{P_2}\}\rangle$ as P_3.

the above goal. We abbreviate `main` as `m`, `check` as `ch`, `alreaydchecked` as `al`, `timeconsumingcheck` as `t`, `announce` as `a` and `hear` as `h`. In the following, we omit the evaluation of equality. We assume that the execution of the body is done from left to right.

1. $PS = \{\langle\{\underline{\mathtt{m(X,Y,1)}}\}, \emptyset, \{\mathtt{X,Y}\}\rangle\}$. $CBS = \langle\emptyset;\emptyset\rangle$.
2. $PS = \{\langle\{\underline{\mathtt{q(X,Y)}}, \mathtt{ch(X,c(1,1,1))}\}, \emptyset, \{\mathtt{X,Y}\}\rangle,$
   $\langle\{\mathtt{r(X,Y)}, \mathtt{ch(X,c(1,2,1))}\}, \emptyset, \{\mathtt{X,Y}\}\rangle\}$.
3. $PS = \{\langle\{\underline{\mathtt{ch(a,c1)}}\}, \emptyset, \{\mathtt{a,b}\}\rangle, \mathtt{P}_1\}$[4].
4. $PS = \{\langle\{\underline{\mathtt{h(al(a))}}, \mathtt{a(cut(c1,1))}\}, \emptyset, \{\mathtt{a,b}\}\rangle,$
   $\langle\{\underline{\mathtt{a(cut(c1,2))}}, \mathtt{t(a)}, \mathtt{a(al(a))}\}, \emptyset, \{\mathtt{a,b}\}\rangle, \mathtt{P}_1\}$.
5. $PS = \{\mathtt{P}_2, \langle\{\underline{\mathtt{t(a)}}, \mathtt{a(al(a))}\}, \emptyset, \{\mathtt{a,b}\}\rangle, \mathtt{P}_1\}$[5].
   $CBS = \langle\{\mathtt{cut(c1,2)}\};\emptyset\rangle$
6. Suppose a time-consuming check $t(a)$ is executed.
   $PS = \{\mathtt{P}_2, \langle\{\underline{\mathtt{a(al(a))}}\}, \emptyset, \{\mathtt{a,b}\}\rangle, \mathtt{P}_1\}$.
7. $PS = \{\mathtt{P}_2, \langle\emptyset, \emptyset, \{\mathtt{a,b}\}\rangle, \mathtt{P}_1\}$.
   $CBS = \langle\{\mathtt{cut(c1,2)}, \mathtt{al(a)}\};\emptyset\rangle$
   Since $\langle\emptyset, \emptyset, \{\mathtt{a,b}\}\rangle$ is found, $\mathtt{X} = \mathtt{a}, \mathtt{Y} = \mathtt{b}$ and $CBS = \langle\{\mathtt{cut(c1,2)}, \mathtt{al(a)}\};\emptyset\rangle$ is returned.
8. We furtherly reduce the process $\mathtt{P}_1$.
   $PS = \{\mathtt{P}_2, \langle\{\underline{\mathtt{r(X,Y)}}, \mathtt{ch(X,c2)}\}, \emptyset, \{\mathtt{X,Y}\}\rangle\}$.
9. $PS = \{\mathtt{P}_2, \langle\{\underline{\mathtt{ch(a,c2)}}\}, \emptyset, \{\mathtt{a,c}\}\rangle\}$.
10. $PS = \{\mathtt{P}_2, \langle\{\underline{\mathtt{h(al(a))}}, \mathtt{a(cut(c2,1))}\}, \emptyset, \{\mathtt{a,c}\}\rangle,$
    $\langle\{\mathtt{a(cut(c2,2))}, \mathtt{t(a)}, \mathtt{a(al(a))}\}, \emptyset, \{\mathtt{a,c}\}\rangle\}$.
11. Since $al(a) \in TRUE(CBS)$, $h(al(a))$ is succeeded.
    $PS = \{\mathtt{P}_2, \langle\{\underline{\mathtt{a(cut(c2,1))}}\}, \emptyset, \{\mathtt{a,c}\}\rangle, \mathtt{P}_3\}$[6].
12. $PS = \{\mathtt{P}_2, \langle\emptyset, \emptyset, \{\mathtt{a,c}\}\rangle, \mathtt{P}_3\}$.
    $CBS = \langle\{\mathtt{cut(c1,2)}, \mathtt{al(a)}, \mathtt{cut(c2,1)}\};\emptyset\rangle$
    Since $\langle\emptyset, \emptyset, \{\mathtt{a,c}\}\rangle$ is found, $\mathtt{X} = \mathtt{a}, \mathtt{Y} = \mathtt{c}$ and $CBS = \langle\{\mathtt{cut(c1,2)}, \mathtt{al(a)}, \mathtt{cut(c2,1)}\};\emptyset\rangle$ is returned.

In this example, thanks to the announcement of `al(a)` in the first search path, we do not need to compute time-consuming check again for the second solution.

Related Work

There exists a similar operator to announcing annotation in concurrent constraint languages such as *cc* language (Saraswat, 1993) called `tell`. However, when a constraint is asserted by `tell` operator, it is valid only in the subsequent paths extended under the node where the assertion was made. Dietzen and Pfenning (Dietzen and Pfenning, 1991) or Dung (Dung, 1992) proposed formalizations of assertions in terms of hypothetical reasoning. These approaches manipulate assertion during the execution as hypotheses and use these hypotheses in only the paths in the search tree extended under the node where the hypotheses are assumed. Therefore, this formalization is similar to an idea of `tell` operator.

On the other hand, the operator `announce` is like "assert" command in PROLOG that in one path, after `announce`(P) is made, P becomes true in every path.

(Hayashi, 1998) gives a procedure of agent in planning where sensory data can be influenced even during reasoning process. This is a kind of search control coming from the outside world whereas our framework concerns search control *within* a logic program.

[4]We abbreviate `c(1,1,1)` as `c1`, `c(1,2,1)` as `c2`, and a process $\langle\{\mathtt{r(X,Y)}, \mathtt{ch(X,c2)}\}, \emptyset, \{\mathtt{X,Y}\}\rangle$ as $\mathtt{P}_1$.

[5]We abbreviate a process $\langle\{\mathtt{h(al(a))}, \mathtt{a(cut(c1,1))}\}, \emptyset, \{\mathtt{a,b}\}\rangle$, as $\mathtt{P}_2$.

[6]We abbreviate a process $\langle\{\mathtt{a(cut(c2,2))}, \mathtt{t(a)}, \mathtt{a(al(a))}\}, \emptyset, \{\mathtt{a,c}\}\rangle$ as $\mathtt{P}_3$.

Conclusion

The contributions of this paper are as follows.

- We propose *global abduction* in which we abduce a belief literal and can use abduced belief in a different search path.
- We formalize a semantics of global abduction as "all's well that ends well" principle.
- We discuss about the application of global abduction, namely, don't-care nondeterminism and reuse of the result in the different search path.

We would like to purse the following research.

- For the above examples to work properly, we need to assume the specific control structure and we need to relax the assumption.
- Finding interesting applications furtherly for the global abduction including formalizing actions with side-effect and reactive planning using the global abduction.
- Extension of language to include negation as failure and manipulate a wider class of integrity constraints.

Acknowledgements I thank Bob Kowalski, Thomas Eiter and Ilkka Niemelä for discussions with this paper. This research is supported by Grants-in-Aid for Scientific Research from JSPS, No. 16200010.

References

Baral, C. and McIlraith, S. 2002. The Third International Cognitive Robotics Workshop http://www.ksl.stanford.edu/cogrob2002/.

Dietzen, S. and Pfenning, F. 1991. A Declarative Alternative to "Assert" in Logic Programming. *Proc. of ILPS'91*, pp. 372 – 386.

Denecker M. and De Schreye, D. 1998. SLDNFA: An Abductive Procedure for Abductive Logic Programs. *JLP*, 34(2), pp. 111-167.

Dung, P., M. 1992. Declarative Semantics of Hypothetical Logic Programming with Negation as Failure. *Proc. of ELP'92*, LNCS 660, pp. 45 – 58.

Fitting, M. C. 1985. A Kripke/Kleene Semantics for Logic Programs. *JLP*, Vol 2. pp. 295 – 312.

Hayashi, H. 1998. Knowledge Assimilation and Proof Restoration Through the Addition of Goals. *Proc. of the 8th International Conference on Artificial Intelligence: Methodology, Systems, Applications*, pp. 291 – 302.

Kakas, A. C. and Mancarella, P. 1990. On the Relation between Truth Maintenance and Abduction. *Proc. of PRICAI-90*, pp. 438 – 433.

Kowalski, R. A., Wetzel, G. and Toni, F. 1998. Executing Suspended Logic Programs. *Fundamenta Informaticae* 34 (3), pp. 203 – 224.

Przymusinski, T. 1990. The Well-Founded Semantics Coincides with the Three-Valued Stable Semantics. *Fundamenta Informaticae* 13 (4), pp. 445 – 463.

Saraswat, V. A., 1993. Concurrent Constraint Programming. Doctoral Dissertation Award and Logic Programming Series, MIT Press.

Satoh, K., Inoue, K., Iwanuma, K., and Sakama, C. 2000 Speculative Computation by Abduction under Incomplete Communication Environments. *Proc. of ICMAS2000*, pp. 263 – 270.

Satoh, K. 2002 Speculative Computation and Abduction for an Autonomous Agent. *Proc. of the NMR'02*, pp. 191 – 199.

Satoh, K. and Yamamoto, K. 2002 Speculative Computation with Multi-Agent Belief Revision. *Proc. of AAMAS2002*, pp. 897 – 904.