Speculative Computation and Abduction for an Autonomous Agent

Ken Satoh
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan

Abstract

In this paper, we propose an agent architecture for a combination of speculative computation and abduction. *Speculative computation* is a tentative computation when complete information for performing computation is not obtained. We use a default value to complement such incomplete information. Unlike usual default reasoning, the real value for the information can be obtained during the computation and the computation can be revised on the fly. In the previous work, we applied this technique to handling distributed problem solving under incomplete communication environments in the context of multi-agent systems and proposed correct procedures in abductive logic programming in terms of perfect model semantics. In the previous work, however, we regard assumptions as defaults and therefore, we could not perform a hypothetical reasoning which is the original usage of abduction. In this paper, we extend our framework so that speculative computation and abduction can be both performed. As a result, our procedure becomes an extension of the abductive procedure developed by Kakas and Mancarella augmented by dynamic belief revision mechanism about outside world.

1 Introduction

In this paper, we consider a method of combining a dynamically revisable computation and abductive reasoning. Consider the following example.

- A robot would like to go from a room to another room through intermediate rooms by making a plan before it takes an action.
- There is a door between rooms; and if the door is closed, the robot cannot get in the room of the destination.
- There is a lion which destroys a robot if they are in the same room.
- There is a sensor to detect which door between the rooms is open or not, and an image analysis module[1] to recognize a lion in a room.
- The information from the sensor and the recognizer will be obtained during the planning.
- We suppose that a robot has default values about the information sent from the sensor and the recognizer before starting planning.

The robot sends a request to the sensor and the recognizer for the necessary information to make a plan. Simultaneously, the robot can also make a tentative plan based on his default assumptions. This may be successful even if no information is returned from the sensor and the recognizer since the default assumptions are normally true under the usual circumstance. Suppose that the tentative planning takes a time and some information is returned from the sensor and the recognizer during planning. If the information is consistent with the default assumptions, then it does not cause any problem and the robot can continue the planning. If the robot waited for the returned information, the robot would not start the planning before this point. Therefore, making a tentative plan beforehand is effective in this case. However, suppose that the obtained information is contradictory with the default assumption. Then, it is no use to continue making a tentative

[1] We call the image analysis module a recognizer in this paper.

plan which does not reflect the outside world. In this case, the robot must revise the computation dynamically so that the obtained information is adapted. In this situation making a tentative plan is not effective. Thus, we call this tentative computation a *speculative computation*. Note that this kind of dynamic revision can occur many times as new information obtained from sensors since the world is dynamically changed. Therefore, we need a mechanism of iterative dynamic revisions as well.

We applied the idea of speculative computation in order to handle distributed problem solving under incomplete communication environments in the context of multi-agent systems and proposed correct procedures in abductive logic programming in terms of perfect model semantics (Satoh00; Satoh02). The usage for abductive logic programming in the work is default reasoning and so it is different from ordinary abduction treated in (Kakas90). The ordinary abduction is useful not only for default reasoning but also other applications (Kakas98) such as the above example of planning. In this paper, we extend our framework so that the ordinary abduction can be performed for a more powerful reasoning mechanism. We distinguish the assumption returned from the outside worlds and the abducibles used inside the agent's reasoning. As a result, our method is an extension of the procedure of Kakas and Mancarella (Kakas90) so that dynamic revision of computation is possible.

2 Framework of Speculative Computation in Master-Slave System

A *speculative framework for a master-slave system* SF_{MS} is the following tuple $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{A}, \mathcal{P}, \mathcal{I} \rangle$ where

- Σ is a finite set of constants. We call an element in Σ a *slave agent identifier*.
- $\mathcal{E}$ is a set of predicates called *external predicates*. When Q is a literal with an external predicate and S is a slave agent identifier, we call $Q@S$ an *askable literal*. We define $\sim (Q@S)$ as $(\sim Q)@S$.
- Δ is a set of ground askable literals satisfying the following condition:

  Δ does not contain both of $p(t_1, ..., t_n)@S$ and $\sim p(t_1, ..., t_n)@S$ simultaneously[2] .

  We call Δ a *default answer set*.
- $\mathcal{A}$ is a set of predicates called *abducible predicates*. When Q is a literal with an *abducible predicate*, Q is called *abducible*.
- $\mathcal{P}$ is a set of rules of the form:

  $$H \leftarrow B_1, B_2, ..., B_n.$$

  where

  - H is a positive ordinary literal[3] .
  - each of $B_1, ..., B_n$ is an ordinary literal or an askable literal or an abducible.

  Let R be a rule of the form $H \leftarrow B_1, B_2, ..., B_n$. We call H a head denoted as $head(R)$ and $B_1, ..., B_n$ a body denoted as $body(R)$. $head(R)$ is non-empty, but if there is no literal in the body, $body(R) = \emptyset$.
- $\mathcal{I}$ is a set of integrity constraints of the form:

  $$\perp \leftarrow B_1, B_2, ..., B_n.$$

  where

  - $\perp$ is a special symbol for contradiction.
  - each of $B_1, ..., B_n$ is is an ordinary literal or an askable literal or an abducible, but at least one of them is an askable literal or an abducible.

  □

Intuitively, an askable literal has two kinds of meaning.

1. An askable literal $Q@S$ in a rule in $\mathcal{P}$ represents a question a master agent asks to a slave agent S.
2. An askable literal in Δ represents the default truth value; if $p(t_1, ..., t_n)@S \in \Delta$, $p(t_1, ..., t_n)$ is normally true for a question to an agent S, and if $\sim p(t_1, ..., t_n)@S \in \Delta$, $p(t_1, ..., t_n)$ is normally false for a question to an agent S.

Example 1 *The example of robot planning in the Introduction is represented as the following speculative framework $SF_{MS} = \langle \Sigma, \mathcal{E}, \Delta, \mathcal{A}, \mathcal{P}, \mathcal{I} \rangle$. Below, $open(a, b)$ means that a door from the room a to the room b is open. $lion\_in(a)$ means that a lion is in the room a. We have default information that every door*

[2] Note that we can have different default values for different agents. For example, $p(t_1, ..., t_n)@S_1$ and $\sim p(t_1, ..., t_n)@S_2$ are possible for S_1 and S_2. Moreover, there might be a case that neither $p(t_1, ..., t_n)@S$ nor $\sim p(t_1, ..., t_n)@S$ is in Δ.

[3] An ordinary literal is a literal which is neither an askable literal nor an abducible.

is open and a lion is usually at the room 3. The set of rules is used to create a plan to go from a room to another room. However, there is an integrity constraint saying that there is no lion at the room of the destination to avoid being destroyed.

- $\Sigma = \{sensor, recognizer\}$
- $\mathcal{E} = \{open, lion\_in\}$
- $\Delta = \{open(1,2)@sensor, open(1,3)@sensor,$
  $open(2,4)@sensor, open(3,4)@sensor,$
  $\sim lion\_in(2)@recognizer,$
  $lion\_in(3)@recognizer,$
  $\sim lion\_in(4)@recognizer\}$
- $\mathcal{A} = \{go\_to\_room^\dagger\}$[4]
- $\mathcal{P}$ *is the following set of rules*[5] *:*

  $plan(F,F) \leftarrow .$

  $plan(X,F) \leftarrow$

  $\quad door(X,Y), open(X,Y)@sensor,$

  $\quad go\_to\_room^\dagger(X,Y), plan(Y,F).$

  $door(1,2) \leftarrow .$

  $door(1,3) \leftarrow .$

  $door(2,4) \leftarrow .$

  $door(3,4) \leftarrow .$
- $\mathcal{I}$ *is the following set of the integrity constraint:*

  $\bot \leftarrow$

  $\quad go\_to\_room^\dagger(X,Y),$

  $\quad lion\_in(Y)@recognizer.$

  □

In the above representation, every door is assumed to be open and a lion is assumed to be in the room 3 by default.

3 An Implementation of Speculative Computation

In this section, we show an implementation of speculative computation. The execution of speculative framework is based on two phases, a *process reduction phase* and a *fact arrival phase*. The process reduction phase is a normal execution of a program in a master agent and the fact arrival phase is an interruption phase when an answer arrives from a slave agent.

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 used defaults have not been contradictory with the current belief state.

In the process reduction phase, we reduce an active process into a new process. Reduction for an ordinary literal or an abducible corresponds with treatment of an ordinary literal or an abducible in *abductive derivation* of (Kakas90). Moreover, we use an extended literal in our procedure which corresponds with *consistency derivation* of (Kakas90) so that consistency derivation can be interleaved with abductive derivation whereas such interleaving is not implemented in (Kakas90).

The major difference between our procedure and that of (Kakas90) is treatment of askable literals. We assume that an agent has an belief set which initially defines a default of the outside information and is changed according to the information coming from outside world. When a process needs to reduce an askable literal, the literal is asked to the designated agent. Then, the literal is assumed and if it is in the current belief set, we can continue the process, otherwise we must suspend the process.

When an answer confirming a default comes from a slave agent, we just continue the current process. This is an effect of speculative computation since at this point, we have already executed a process without waiting. On the other hand, when an answer denying the current default comes from a slave agent, we must update the current belief state about the outside world and the current process must be suspended if the process uses the default which is contradictory with the new belief.

3.1 Preliminary Definitions

We define the following concepts for a proof procedure.

Definition 1 *An* extended literal *is either a literal or an expression of the form* $fail(\{L_1, ..., L_n\})$ *where* L_i *is a literal.* □

$fail(\{L_1, ..., L_n\})$ is used to show that there is no proof for one of L_i and corresponds with consistency derivation in (Kakas90).

Definition 2 *A* process *is the following tuple* $\langle GS, OD, IA, ANS\rangle$ *which consists of*

[4] We put † for an abducible predicate to distinguish from non-abducible predicate.

[5] A string beginning with an upper case letter represents a variable and a string beginning with a lower case letter represents a constant.

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

- *OD: A set of askable literals called* outside defaults.

- *IA: A set of negative literals or abducibles called* inside assumptions.

- *ANS: A set of instantiations of variables in the initial query.* □

Each process represents an alternative way of computation. The intuitive meaning of the above objects is as follows:

- GS expresses the current status of an alternative of computation.

- OD is a set of assumed information about outside world during a process.

- IA is a set of abducibles which has been assumed during a process.

- ANS gives an answer for variables in the initial query.

We use the following three sets for process reduction.

Definition 3

- A process set PS *is a set of processes.*

- *A set of* already asked queries AAQ *is a set of askable literals.*

- A current belief state CBS *is a set of askable literals.* □

PS is a set of processes which express all the alternative computations considered so far. AAQ is used to avoid from asking redundant questions to slave agents, and CBS is the current belief state which expresses the agent's belief of the current status of the outside world.

We define an *active* process and a *suspended* process as follows.

Definition 4

Let $\langle GS, OD, IA, ANS\rangle$ *be a process and* CBS *be a current belief state. A process is* active w.r.t. CBS *if* $OD \subseteq CBS$ *and a process is* suspended w.r.t. CBS *otherwise.* □

The intuitive meaning of active process is a process whose set of assumed information about outside world is consistent with the current belief set. It is because the current belief set is *very likely to reflect the outside world* and so the process using a subset of the current belief set as assumption is supposed to make a plan being consistent with the outside world.

3.2 Process Reduction Phase

Now, we give the precise description of the process reduction phase and the fact arrival phase. In the following description, we specify changed PS, AAQ, CBS as $NewPS$, $NewAAQ$, $NewCBS$; otherwise each PS, AAQ, CBS is unchanged.

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

Iteration Step: Do the following.

Case 1 If there is an active process $\langle \emptyset, OD, IA, ANS\rangle$ w.r.t. CBS in PS, then terminate the process by returning outside defaults OD, inside assumptions IA, and instantiation for variables ANS.

Case 2 If there is no active process then terminate the process by reporting a failure of the goal.

Case 3 Select an active process $\langle GS, OD, IA, ANS\rangle$ w.r.t. CBS from PS and select an extended literal L in GS. Let $PS' = PS - \{\langle GS, OD, IA, ANS\rangle\}$ and $GS' = GS - \{L\}$.

For the selected extended literal L, do the following.

Case 3.1 If L is a positive ordinary literal,
$$NewPS = PS' \cup \{\langle (\{body(R)\} \cup GS')\theta, OD, IA, ANS\theta\rangle \mid \exists R \in \mathcal{P} \text{ and } \exists \text{most general unifier(mgu) } \theta \text{ s.t. } head(R)\theta = L\theta\}$$

Case 3.2 If L is a ground negative ordinary literal or a ground abducible then do the following:

Case 3.2.1 If $L \in IA$ then
$$NewPS = PS' \cup \{\langle GS', OD, IA, ANS\rangle\}$$

Case 3.2.2 If $\overline{L}$[6] $\in IA$ then
$$NewPS = PS'$$

Case 3.2.3 If $L \notin IA$ then
$$NewPS = PS' \cup \{\langle NewGS, OD, IA \cup \{L\}, ANS\rangle\}$$

[6] $\overline{L}$ is a complement of a literal L.

where $NewGS =$
$\{fail(BS)|BS \in resolvent(L, \mathcal{P} \cup \mathcal{I})\} \cup GS'$
and $resolvent(L, T)$ is defined as follows:

- if L is a ground negative ordinary literal, $resolvent(L, T) =$
  $\{\{L_1\theta, ..., L_k\theta\}|H \leftarrow L_1, ..., L_k \in T$ s.t. $\overline{L} = H\theta$ by a ground substitution $\theta\}$.
- if L is a ground abducible, $resolvent(L, T) =$
  $\{\{L_1\theta, ..., L_{i-1}\theta, L_{i+1}\theta, ..., L_k\theta\}|$ $\perp \leftarrow L_1, ..., L_k \in T$ s.t. $L = L_i\theta$ by a ground substitution $\theta\}$.

Case 3.3 If L is $fail(BS)$, then

- if $BS = \emptyset$, $NewPS = PS'$.
- if $BS \neq \emptyset$ then do the following:
  1. Select B from BS and let $BS' = BS - \{B\}$.
  2. **Case 3.3.1**
     if B is a positive ordinary literal,
     $NewPS = PS'$
     $\cup\{\langle NewGS \cup GS', OD, IA, ANS\rangle\}$
     where
     $NewGS =$
     $\{fail((\{body(R)\} \cup BS')\theta)|\exists R \in \mathcal{P}$ and $\exists$mgu θ s.t. $head(R)\theta = B\theta\}$
     Case 3.3.2
     if B is a ground negative ordinary literal or a ground askable literal or an abducible,
     $NewPS = PS'$
     $\cup\{\langle\{fail(BS')\} \cup GS', OD, IA, ANS\rangle\}$
     $\cup\{\langle\{\overline{B}\} \cup GS', OD, IA, ANS\rangle\}$

Case 3.4 If L is a ground askable literal, $Q@S$, then do the following:

1. if $L \notin AAQ$ and $\overline{L} \notin AAQ$, then send the question Q to the slave agent S and $NewAAQ = AAQ \cup \{L\}$.
2. if $\overline{L} \in OD$ then $NewPS = PS'$
   else $NewPS = PS'$
   $\cup\{\langle GS', OD \cup \{L\}, IA, ANS\rangle\}$.

□

Note that we can select any goal of any active process for a goal reduction and we guarantee the correctness of this procedure by the theorem below. This means that we can put additional strategies for an efficient goal reduction.

3.3 Fact Arrival Phase

Suppose an answer Q is returned from a slave agent S. Let $L = Q@S$. After one step of process reduction is finished, do the following:

- If $\overline{L} \in CBS$, then
  $NewCBS = CBS - \{\overline{L}\} \cup \{L\}$
- else if $L \notin CBS$, then
  $NewCBS = CBS \cup \{L\}$

Note that there might be some askable literals which are not included in the initial belief set. In this case, processes using such askable literals and ones using their complements are both suspended until the answers for the askable literals are returned and therefore no speculative computation is performed. This mechanism is used when we do not know the default value of the askable literal, or we need the definite answer to proceed the execution.

3.4 Correctness of the Proof Procedure

For the correctness of the proof procedure, we define the following. We use generalized stable model semantics (Kakas90) for the semantics of the above program.

Definition 5 *Let T be a set of rules and integrity constraints. We denote the set of ground rules obtained by replacing all the variables in every rule or every integrity constraint of T by every ground term in the language as Π_T.* □

Definition 6 *Let T be a set of rules and integrity constraints. Let M be a set of ground atoms and Π_T^M be the following program.*

$$\Pi_T^M = \{H \leftarrow B_1, ..., B_l| \text{“}H \leftarrow B_1, ..., B_l, {\sim}A_1, ..., {\sim}A_h.\text{''} \in \Pi_P \text{ and } A_i \notin M \text{ for each } i = 1, ..., h.\}$$

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 $\perp \notin M$* □

Definition 7 *(Kakas90) Let T be a set of rules and integrity constraints, and Θ be a set of ground abducibles.*

A generalized stable model *$M(\Theta)$ is a stable model of $T \cup \mathcal{F}(\Theta)$ where*
$\mathcal{F}(\Theta) =$
$\{H \leftarrow |H$ *is a positive literal and* $H \in \Theta\}$.
□

The following theorem shows correctness of the above procedure. The theorem intuitively means that for any evaluation strategy of processes, when we receive an answer with a set of outside defaults and a set of inside assumptions from the proof procedure, the answer is correct w.r.t. a generalized stable model w.r.t. inside assumptions and a program which is obtained from the original program and the current belief of the world.

Theorem 1 *Let $SF_{MS} = \langle \Sigma, \mathcal{E}, \Delta, \mathcal{A}, \mathcal{P}, \mathcal{I} \rangle$ be a speculative framework where $\mathcal{P}$ is a call-consistent logic program whose set of integrity constraints is satisfiable. Let GS be an initial goal set. Suppose that GS is reduced to $\emptyset$ when OD is outside defaults, IA is inside assumptions, ANS is set of instantiation of variables in GS and CBS is the current belief set. Let GS' be a goal obtained from GS by replacing all the variables in GS by ANS. Then, there is a generalized stable model $M(\Theta)$ for $\mathcal{P} \cup \mathcal{I} \cup \mathcal{F}(CBS)$, such that $M(\Theta) \models GS'$ and $OD \subseteq CBS$ and $IA \subseteq \Theta$.*

Sketch of Proof Since $\mathcal{P}$ is a call-consistent logic program, so is $\mathcal{P} \cup \mathcal{I} \cup \mathcal{F}(CBS)$. Then, we can construct an abductive derivation (Kakas90) from process reduction steps from $\langle GS, \emptyset, \emptyset, ANS' \rangle$ to $\langle \emptyset, OD, IA, ANS \rangle$ where ANS' is a set of variables in GS. This derivation is correct for generalized stable model semantics for a call-consistent logic program according to the result of (Kakas90). This means $M(\Theta) \models GS\theta$ and $OD \subseteq CBS$ and $IA \subseteq \Theta$. □

4 Example

We use the program in Example 1 and take the following strategy for process reduction.

1. When we reduce a positive literal, new processes are created along with the rule order in the program which are unifiable with the positive literal.
2. We always select a newly created or a newly resumed process and a left-most literal.

The following is an execution trace for $plan(1,4)$. We show PS, AAQ, CBS only when a change occurs. We abbreviate *plan* as p, *door* as d, *open* as o, *go_to_room* as $g^\dagger$, *lion_in* as l, *sensor* as s and *recognizer* as r. We underlined a selected literal in the selected active process.

1. $PS = \{\langle \{\underline{p(1,4)}\}, \emptyset, \emptyset \rangle\}$[7] ,
   $AAQ = \emptyset$,
   $CBS =$
   $\{o(1,2)@s, o(1,3)@s, o(2,4)@s, o(3,4)@s,$
   $\sim l(2)@r, l(3)@r, \sim l(4)@r\}$
2. By **Case 3.1**,
   $PS = \{$
   $\langle \{\underline{d(1,Y)}, o(1,Y)@s, g^\dagger(1,Y), p(Y,4)\},$
   $\emptyset, \emptyset \rangle\}$
3. By **Case 3.1**,
   $PS = \{$
   $\langle \{o(1,2)@s, g^\dagger(1,2), p(2,4)\}, \emptyset, \emptyset \rangle,$
   $\langle \{o(1,3)@s, g^\dagger(1,3), p(3,4)\}, \emptyset, \emptyset \rangle\}$
4. By **Case 3.4**,
   $o(1,2)$ **is asked to the agent** s.
   $PS = \{$
   $\langle \{\underline{g^\dagger(1,2)}, p(2,4)\}, \{o(1,2)@s\}, \emptyset \rangle, P_1$[8] $\}$
   $AAQ = \{o(1,2)@s\}$
5. By **Case 3.2.3**,
   $PS = \{$
   $\langle \{\underline{fail(\{l(2)@r\})}, p(2,4)\},$
   $\{o(1,2)@s\}, \{g^\dagger(1,2)\} \rangle,$
   $P_1\}$
6. By **Case 3.3.2**,
   $PS = \{$
   $\langle \{\underline{fail(\emptyset)}, p(2,4)\}, \{o(1,2)@s\}, \{g^\dagger(1,2)\} \rangle,$
   $\langle \{\sim l(2)@r, p(2,4)\},$
   $\{o(1,2)@s\}, \{g^\dagger(1,2)\} \rangle,$
   $P_1\}$
7. By **Case 3.3**,
   $PS = \{$
   $\langle \{\underline{\sim l(2)@r}, p(2,4)\},$
   $\{o(1,2)@s\}, \{g^\dagger(1,2)\} \rangle,$
   $P_1\}$
8. By **Case 3.4**,
   $\sim l(2)$ **is asked to the agent** r.
   $PS = \{$
   $\langle \{\underline{p(2,4)}\},$
   $\{o(1,2)@s, \sim l(2)@r\}, \{g^\dagger(1,2)\} \rangle,$
   $P_1\}$
   $AAQ = \{o(1,2)@s, \sim l(2)@r\}$
9. $o(1,2)$ **is returned from** s**!!**
   Nothing changes.
10. By **Case 3.1**,
    $PS = \{$
    $\langle \{\underline{d(2,Y)}, o(2,Y)@s, g^\dagger(2,Y), p(Y,4)\},$
    $\{o(1,2)@s, \sim l(2)@r\}, \{g^\dagger(1,2)\} \rangle,$
    $P_1\}$
11. By **Case 3.1**,
    $PS = \{$
    $\langle \{o(2,4)@s, g^\dagger(2,4), p(4,4)\},$
    $\{o(1,2)@s, \sim l(2)@r\}, \{g^\dagger(1,2)\} \rangle$
    $P_1\}$
12. $l(2)$ **is returned from** r**!!**

[7] We omit ANS part in the process since there is no variable in the initial query.

[8] From now on P_1 is an abbreviation of $\langle \{o(1,3)@s, g^\dagger(1,3), p(3,4)\}, \emptyset, \emptyset \rangle$.

By **Fact Arrival Phase**,
$CBS =$
$\{o(1,2)@s, o(1,3)@s, o(2,4)@s, o(3,4)@s,$
$l(2)@r, l(3)@r, \sim l(4)@r\}$
$PS = \{$
$\langle\{\underline{o(1,3)@s}, g^\dagger(1,3), p(3,4)\}, \emptyset, \emptyset\rangle, P_2$[9] $\}$

13. By **Case 3.4**,
$o(1,3)$ **is asked to the agent** s.
$PS = \{$
$\langle\{\underline{g^\dagger(1,3)}, p(3,4)\}, \{o(1,3)@s\}, \emptyset\rangle, P_2\}$
$AAQ = \{o(1,2)@s, \sim l(2)@r, o(1,3)@s\}$

14. By **Case 3.2.3**,
$PS = \{$
$\langle\{\underline{fail(\{l(3)@r\})}, p(3,4)\},$
$\{o(1,3)@s\}, \{g^\dagger(1,3)\}\rangle,$
$P_2\}$

15. By **Case 3.3.2**,
$PS = \{$
$\{\underline{fail(\emptyset)}, \{o(1,3)@s\}, \{g^\dagger(1,3)\}\rangle,$
$\langle\{\sim l(3)@r, p(3,4)\},$
$\{o(1,3)@s\}, \{g^\dagger(1,3)\}\rangle,$
$P_2\}$
$AAQ =$
$\{o(1,2)@s, \sim l(2)@r, o(1,3)@s, \sim l(3)@r\}$

16. By **Case 3.3**,
$PS = \{$
$\langle\{\underline{\sim l(3)@r}, p(3,4)\},$
$\{o(1,3)@s\}, \{g^\dagger(1,3)\}\rangle,$
$P_2\}$
$AAQ =$
$\{o(1,2)@s, \sim l(2)@r, o(1,3)@s, \sim l(3)@r\}$

17. By **Case 3.4**,
$\sim l(3)$ **is asked to the agent** r.
$PS = \{$
$\langle\{p(3,4)\},$
$\{o(1,3)@s, \sim l(3)@r\}, \{g^\dagger(1,3)\}\rangle,$
$P_2\}$

18. $\sim l(2)$ **is returned from** r**!!**
By **Fact Arrival Phase**,
$CBS =$
$\{o(1,2)@s, o(1,3)@s, o(2,4)@s, o(3,4)@s,$
$\sim l(2)@r, l(3)@r, \sim l(4)@r\}$
$PS = \{$
$\langle\{\underline{o(2,4)@s}, g^\dagger(2,4), p(4,4)\},$
$\{o(1,2)@s, \sim l(2)@r\}, \{g^\dagger(1,2)\}\rangle$
P_3[10] $\}$

19. By **Case 3.4**,
$o(2,4)$ **is asked to the agent** s.
$PS = \{$
$\langle\{\underline{g^\dagger(2,4)}, p(4,4)\},$
$\{o(1,2)@s, \sim l(2)@r, o(2,4)@s\}, \{g^\dagger(1,2)\}\rangle$
$P_3\}$
$AAQ =$
$\{o(1,2)@s, \sim l(2)@r, o(1,3)@s,$
$\sim l(3)@r, o(2,4)@s\}$

20. By **Case 3.2.3**,
$PS = \{$
$\langle\{\underline{fail(\{l(4)@r\})}, p(4,4)\},$
$\{o(1,2)@s, \sim l(2)@r, o(2,4)@s\},$
$\{g^\dagger(1,2), g^\dagger(2,4)\}\rangle$
$P_3\}$

21. By **Case 3.3.2**,
$PS = \{$
$\langle\{\underline{fail(\emptyset)},$
$\{o(1,2)@s, \sim l(2)@r, o(2,4)@s\},$
$\{g^\dagger(1,2), g^\dagger(2,4)\}\rangle$
$\langle\{\sim l(4)@r, p(4,4)\},$
$\{o(1,2)@s, \sim l(2)@r, o(2,4)@s\},$
$\{g^\dagger(1,2), g^\dagger(2,4)\}\rangle$
$P_3\}$

22. By **Case 3.3**,
$PS = \{$
$\langle\{\underline{\sim l(4)@r}, p(4,4)\},$
$\{o(1,2)@s, \sim l(2)@r, o(2,4)@s\},$
$\{g^\dagger(1,2), g^\dagger(2,4)\}\rangle$
$P_3\}$

23. By **Case 3.4**,
$\sim l(4)$ **is asked to the agent** r.
$PS = \{$
$\langle\{\underline{p(4,4)}\},$
$\{o(1,2)@s, \sim l(2)@r, o(2,4)@s, \sim l(4)@r\},$
$\{g^\dagger(1,2), g^\dagger(2,4)\}\rangle$
$P_3\}$
$AAQ =$
$\{o(1,2)@s, \sim l(2)@r, o(1,3)@s,$
$\sim l(3)@r, o(2,4)@s, \sim l(4)@r\}$

24. By **Case 3.1**,
$PS = \{$
$\langle\emptyset,$
$\{o(1,2)@s, \sim l(2)@r, o(2,4)@s, \sim l(4)@r\},$
$\{g^\dagger(1,2), g^\dagger(2,4)\}\rangle$
$P_3\}$

25. $\{o(1,2)@s, \sim l(2)@r, o(2,4)@s, \sim l(4)@r\}$ and $\{g^\dagger(1,2), g^\dagger(2,4)\}$ are returned as outside defaults

[9] From now on P_2 is an abbreviation of $\langle\{o(2,4)@s, g^\dagger(2,4), p(4,4)\},$ $\{o(1,2)@s, \sim l(2)@r\}, \{g^\dagger(1,2)\}\rangle$.

[10] From now on, P_3 is an abbreviation of $\langle\{p(3,4)\}, \{o(1,3)@s, \sim l(3)@r\}, \{g^\dagger(1,3)\}\rangle$.

and inside assumptions.

At Step 4, a default $o(1,2)@s$ is assumed and computation of this process continues and at the same time, the real truth value of $o(1,2)$ is asked to the sensor s. If we had to wait for an answer from the agent s, we would have to suspend this process. This is an effect of speculative computation.

At Step 5, we perform an ordinary abduction. That is, we assume $g^{\dagger}(1,2)$ and check if any integrity constraint does not lead to contradiction. In this case, we check if $l(2)@r$ is not derived.

At Step 9, suppose that the answer is returned from s. Since this answer confirms the default, nothing changes and we can continue the reduction. This is the effect of speculative computation.

At Step 12, suppose the answer $l(2)$ is returned from r. Since this answer denies the default, we have to change the CBS and the current reduced process is suspended because the outside default is contradictory with the CBS.

At Step 17, the current reduced process is suspended because the askable literal is contradictory with the CBS.

At Step 18, suppose that the new answer $\sim l(2)$ is returned from r. Since this answer denies the default, we have to change the CBS and the process which becomes consistent with CBS is activated.

5 Related Research

In computer science, there are several research on speculative computation such as optimistic transaction in database, three-phase transition in fault-tolerant system, efficient execution mechanism for functional programming (Burton85) and parallel logic programming (Gregory93). This work is inspired by some of these works and is an abstract form of the above technique in multi-agent systems.

There are works on a formalization of an agent in terms of logic programming (Kowalski99; Hayashi99; Shanahan98). (Kowalski99) gives an agent cycle which reconciles reactive reasoning and proactive reasoning. However, they do not consider a dynamic belief revision about environment during proactive reasoning. (Shanahan98) proposes an important formalism of combining sensory data, planning and acting so that they can be interleaved. In his formalism, he considers not only a dynamic belief revision, but also a formalization of action which is not considered in our framework. The difference between his approach and our approach is its implementation. In his approach, every time new information leads to contradiction, the reasoning is restarted whereas in our framework, we return to the choice point where new information contradicts with the assumption. Moreover, he does not consider a speculative computation in his framework. The most closely related work with ours is (Hayashi99). He proposes a procedure of agent in planning where sensory data can be influenced even during reasoning process. In his procedure, the previous planning fragment can be reused. However, he does not consider speculative nature of the technique nor an abductive mechanism. Moreover, his procedure seems very complex because he considers deletion and addition of rules whereas our procedure is relatively simple since our procedure is a natural extension of the procedure of (Kakas90) with treatment of derivation using the current belief state.

After the original work of speculative computation (Satoh00), related research with speculative computation has been published (Sakama00; Inoue01). (Sakama00) gives a local semantics of an agent by translating a program into a new one attaching time stamp with each predicate. The revision of each agent is formalized as overriding the truth value of the predicate with the previous time stamp by the truth value of the predicate with the new time stamp. The idea of revision in the belief state is inspired by this revision mechanism and in fact the belief set in our procedure records the newest truth value of the predicate. (Inoue01) gives a bottom-up evaluation of speculative computation by calculating every answer w.r.t. possible replies. However, the work has a difficulty in controlling when queries are sent to other agents for the setting considered here.

This defect of controlling of queries also applies to answer set programming since answer set programming also computes a stable model in a bottom-up manner. Moreover, there is a problem of computation for abduction using answer set programming. Suppose that we have n abducibles. If we take a translation of abducibles using a technique of (Satoh91), then, at the worst case scenario, we have to check 2^n combinations of truth values of abducibles. Therefore, to apply answer set programming to abduction, more sophisticated control method of evaluation is necessary.

6 Conclusion

The following are the contributions of this paper.

- We combine speculative computation and abductive reasoning. We believe this combination gives a foundation for more powerful reasoning systems.

- We propose a sound implementation of the combination for a master-slave multi-agent system in terms of generalized stable model semantics.

This is an initial work on dynamic belief revision and many issues remain for future research.

- Extensions to other kinds of multi-agent systems such as a system where every agent can communicate each other.
- Introducing other kinds of action than reasoning and sensing.
- Controlling effective manipulation of alternative processes. This is related with decision theoretic search since we need to estimate probabilities and utilities to assess exploration of alternative processes.
- Reducing redundancies of computation in order to reuse the same computation in alternative processes.
- Applying our framework to real applications such as robotics.

Acknowledgments

This research is partly supported by Grant-in-Aid for Scientific Research, "Construction of Multi-agent Systems with Speculative Computation", by The Ministry of Education, Culture, Sports, Science and Technology in Japan. I would be very grateful to Robert Kowalski and Murray Shanahan for instructive comments on this paper.

References

Burton, F. W., "Speculative Computation, Parallelism, and Functional Programming", *IEEE Transactions on Computers*, vol c-34 pp. 1190 – 1193 (1985).

Gregory, S., "Experiment with Speculative Parallelism in Parlog", Proceedings of ILPS'93, pp. 370 – 387 (1993).

Hayashi, H., "Replanning in Robotics by Dynamic SLDNF", *Proceedings of IJCAI-99 WS on Scheduling and Planning Meet – Real-Time Monitoring in a Dynamic and Uncertain World* (1999).

Inoue, K., Kawaguchi, S., and Haneda, H., "Controlling Speculative Computation in Multi-Agent Environments", *Proc. of ICLP2001 Workshop on Computational Logic in Multi-Agent Systems (CLIMA-01)*, pp. 9 – 18 (2001).

Kakas, A. C., Kowalski, R., Toni, F., "The Role of Abduction in Logic Programming", Handbook of Logic in Artificial Intelligence and Logic Programming 5, pp. 235 – 324 (1998).

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

Kowalski, R. A., Sadri, F., "From Logic Programming to Multi-Agent Systems", To appear in Annals of Mathematics and Artificial Intelligence (1999).

Sakama, C., Inoue, K., Iwanuma, K., and Satoh, K., "A Defeasible Reasoning System in Multi-Agent Environments", *Proc. of CL2000 Workshop on Computational Logic in Multi-Agent Systems (CLIMA-00)*, pp.1 – 6 (2000).

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

Satoh, K., Iwayama, N., "Computing Abduction by Using the TMS", *Proc. ICLP91*, pp. 505 – 518 (1991).

Satoh, K., Yamamoto, K., "Speculative Computation with Multi-Agent Belief Revision", *Proc. of AAMAS2002*, to appear (2000).

Shanahan, M. P., "Reinventing Shakey", *Working Notes of AAAI Fall Symposium on Cognitive Robotics* (1998), also in *Logic-Based Artificial Intelligence*, J. Minker ed., Kluwer Academic Press, pp. 233 – 253 (2000).