Speculative Computation
佐藤 健 (Satoh, Ken)
National Institute of Informatics

1. Motivation: Kiga-kiku Computing
2. Speculative Computation in Master-Slave Multi-Agent System
3. Conclusion

Explanation will be done by Ken Satoh, or Martine Ceberio, a visiting research collaborator from Univ. of Texas, El Paso.
Underlying Motivation: toward “kiga-kiku” computing
(気が利く計算メカニズム)

● “kiga-kiku”: meaning a mixture of “attentive”, “considerate”, “proactive”, “context-sensitive”....

● A “kiga-kiku” computer can understand a situation and take an appropriate action for the situation without being told explicitly what to do.

● In Japan, a “kiga-kiku” person is respected very much, so Japanese people might be good at making a “kiga-kiku” computer.
Application of “kiga-kiku” computing

• Ubiquitous computing
• Pervasive computing
• Attentive computing
• Intelligent electric appliance
• Multi-agent systems under incomplete communication
What is necessary for “kiga-kiku” computing?

- Situation-awareness or context-awareness
  (ex. serving a hot green tea for the first cup and a warm green tea for the second cup)

- Understanding user intention without much interaction
  (ex. making a flight reservation usually means a reservation of hotels)

- Learning of user’s preference
  (ex. some people like cheap hotels than luxury hotels and vice versa)

- Handling incompleteness and a mechanism of back-up if the “kiga-kiku” action is not appropriate

  ⇒ Speculative Computation by Abduction
Current Research on Multi-Agent System

Assumption

• Communication is guaranteed.

• Process is suspended until some response comes.

However, ...

• In the Internet, communication is not guaranteed. Therefore, process will be suspended forever when response is lost.

• Even if communication is guaranteed, computation in other agent might continue forever.
Motivating Example for Multi-Agent System

• $A$, $B$ and $C$ to attend the meeting.
• If a person is available, then he/she will attend the meeting.
• We ask a person whether he/she is free or not.
• If all the persons are available, we reserve a big room.
• If only two persons are available, we reserve a small room.

Suppose we have answers from $A$ and $B$ that he/she is free but we do not have an answer from $C$.

Then, usual agent cannot decide a room reservation since the answers from $C$ are not obtained.
Solution

• We can decide a room reservation based on a plausible answer whether $C$ is usually busy or not.

• If the answers from $C$ is an exception, then we cancel the room and make a new reservation.

We call this process *speculative computation*. 
Characteristics of Speculative Computation

- Even if information is not complete, we continue computation.
- Reducing a risk of losing benefit. (cf. tentative reservation)
- Sometimes, computation would not be effective.
Speculative Computation and Abduction

• To supplement unknown information, we use a default information by *abduction* (in other words, hypothetical reasoning).

• If the answer for unknown information confirms the default rule, we just continue computation.

• If the answer denies the default rule, an alternative computation is considered.
Ordinary Computation

\textit{START}
Ordinary Computation

START

Query to Other Agent
Ordinary Computation

\[ \text{START} \]

suspended \quad \rightarrow \quad \text{Query to Other Agent}
Ordinary Computation

\[
\text{START}
\]

suspended $\rightarrow$ Query to Other Agent

$\leftarrow$ Answer from Other Agent
Ordinary Computation

\[ \text{START} \]

\[ \text{suspended} \rightarrow \text{Query to Other Agent} \]

\[ \text{resumed} \leftarrow \text{Answer from Other Agent} \]
Speculative Computation

START


Speculative Computation

START

Query to Other Agent
Speculative Computation

START

Query to Other Agent

continued by default value

suspended
Speculative Computation

\[
\text{START} \rightarrow \text{Query to Other Agent} \\
\text{continued by default value} \\
\text{suspended} \\
\text{Answer from Other Agent}
\]

When returned answer is consistent with a default...
Speculative Computation

When returned answer is consistent with a default, we just continue the computation.
Speculative Computation

START

Query to Other Agent

continued by default value

suspended

Answer from Other Agent

When returned answer contradicts a default...
Speculative Computation

When returned answer contradicts a default, we resume other alternative computation.
Speculative Computation in Master-Slave Multi Agent System

(Joint work with Inoue, Iwanuma, Sakama (ICMAS00))

• The master (room reservation agent in the example) asks questions to slaves (participants of meeting in the example) and only the master performs a speculative computation.

• If a slave returns an answer for the question, the master might revise his computation.

Note: This dynamic belief revision is essentially different from ordinary (static) belief revision!!
Program Example

• A program:
  \[ \text{meeting}([a, b]) \leftarrow \text{free}@a, \text{free}@b, \text{not free}@c. \]
  \[ \text{meeting}([b, c]) \leftarrow \text{not free}@a, \text{free}@b, \text{free}@c. \]
  \[ \text{meeting}([c, a]) \leftarrow \text{free}@a, \text{not free}@b, \text{free}@c. \]
  \[ \text{meeting}([a, b, c]) \leftarrow \text{free}@a, \text{free}@b, \text{free}@c. \]
  \[ \text{plan}(\text{small\_room}, [a, b]) \leftarrow \text{meeting}([a, b]). \]
  \[ \text{plan}(\text{small\_room}, [b, c]) \leftarrow \text{meeting}([b, c]). \]
  \[ \text{plan}(\text{small\_room}, [c, a]) \leftarrow \text{meeting}([c, a]). \]
  \[ \text{plan}(\text{large\_room}, [a, b, c]) \leftarrow \text{meeting}([a, b, c]). \]

• A set of default answers: \{\text{free}@a, \text{free}@b, \text{free}@c\}
Execution Example

• A program:
  
  \[\text{meeting}(\{a, b\}) \leftarrow \text{free}@a, \text{free}@b, \text{not} \enspace \text{free}@c.\]
  
  \[\text{meeting}(\{b, c\}) \leftarrow \text{not} \enspace \text{free}@a, \text{free}@b, \text{free}@c.\]
  
  \[\text{meeting}(\{c, a\}) \leftarrow \text{free}@a, \text{not} \enspace \text{free}@b, \text{free}@c.\]
  
  \[\text{meeting}(\{a, b, c\}) \leftarrow \text{free}@a, \text{free}@b, \text{free}@c.\]

  \[\text{plan}(\text{small}_\text{room}, \{a, b\}) \leftarrow \text{meeting}(\{a, b\}).\]

  \[\text{plan}(\text{small}_\text{room}, \{b, c\}) \leftarrow \text{meeting}(\{b, c\}).\]

  \[\text{plan}(\text{small}_\text{room}, \{c, a\}) \leftarrow \text{meeting}(\{c, a\}).\]

  \[\text{plan}(\text{large}_\text{room}, \{a, b, c\}) \leftarrow \text{meeting}(\{a, b, c\}).\]

• A set of default answers: \{ free@a, free@b, free@c \}
Execution Example

\[ mt([a, b]) \leftarrow fr@a, fr@b, \textbf{not} fr@c. \]
\[ mt([b, c]) \leftarrow \textbf{not} fr@a, fr@b, fr@c. \]
\[ mt([c, a]) \leftarrow fr@a, \textbf{not} fr@b, fr@c. \]
\[ mt([a, b, c]) \leftarrow fr@a, fr@b, fr@c. \]
\[ plan(sr, [a, b]) \leftarrow mt([a, b]). \]
\[ plan(sr, [b, c]) \leftarrow mt([b, c]). \]
\[ plan(sr, [c, a]) \leftarrow mt([c, a]). \]
\[ plan(lr, [a, b, c]) \leftarrow mt([a, b, c]). \]

1. Active Process: \( \langle plan(R, L) \rangle \)
Execution Example (continued)

\[ mt([a, b]) \leftarrow fr@a, fr@b, \text{ not } fr@c. \]
\[ mt([b, c]) \leftarrow \text{ not } fr@a, fr@b, fr@c. \]
\[ mt([c, a]) \leftarrow fr@a, \text{ not } fr@b, fr@c. \]
\[ mt([a, b, c]) \leftarrow fr@a, fr@b, fr@c. \]

2. The process becomes the following active processes:

\[ \langle mt([a, b]) \rangle \text{ where } R = sr, L = [a, b] \]
\[ \langle mt([b, c]) \rangle \text{ where } R = sr, L = [b, c] \]
\[ \langle mt([c, a]) \rangle \text{ where } R = sr, L = [c, a] \]
\[ \langle mt([a, b, c]) \rangle \text{ where } R = lr, L = [a, b, c] \]
Execution Example (continued)

\[ mt([a, b]) \leftarrow fr@a, fr@b, \text{ not } fr@c. \]
\[ mt([b, c]) \leftarrow \text{ not } fr@a, fr@b, fr@c. \]
\[ mt([c, a]) \leftarrow fr@a, \text{ not } fr@b, fr@c. \]
\[ mt([a, b, c]) \leftarrow fr@a, fr@b, fr@c. \]

3. The first process \( mt([a, b]) \) becomes:
   \[ \langle fr@a, fr@b, \text{ not } fr@c \rangle \]

4. \( fr \) is asked to a slave agent \( a \)
   and the first process becomes
   \[ \langle fr@b, \text{ not } fr@c \rangle \]
   with an assumption \( \{ fr@a \} \).
   (Starting speculative computation)
Execution Example (continued)

\[ mt([a, b]) \leftarrow fr@a, fr@b, \ not \ fr@c. \]
\[ mt([b, c]) \leftarrow \ not \ fr@a, fr@b, fr@c. \]
\[ mt([c, a]) \leftarrow fr@a, \ not \ fr@b, fr@c. \]
\[ mt([a, b, c]) \leftarrow fr@a, fr@b, fr@c. \]

5. \( fr \) is asked to a slave agent \( b \)
   and the first process becomes
   \[ \langle \text{not} \ fr@c \rangle \]
   with an assumption \( \{fr@a, fr@b\} \).
Execution Example (continued)

\[
mt([a, b]) \leftarrow fr@a, fr@b, \textbf{not} \ fr@c.
\]

\[
mt([b, c]) \leftarrow \textbf{not} \ fr@a, fr@b, fr@c.
\]

\[
mt([c, a]) \leftarrow fr@a, \textbf{not} \ fr@b, fr@c.
\]

\[
mt([a, b, c]) \leftarrow fr@a, fr@b, fr@c.
\]

6. \textbf{not} \ fr \ is asked to a slave agent \( c \)
and the first process becomes suspended
because \textbf{not} \ fr@c \ is not a default.
Execution Example (continued)

\[
\begin{align*}
mt([a, b]) & \leftarrow fr@a, fr@b, \text{not fr}@c. \\
mt([b, c]) & \leftarrow \text{not fr}@a, fr@b, fr@c. \\
mt([c, a]) & \leftarrow fr@a, \text{not fr}@b, fr@c. \\
mt([a, b, c]) & \leftarrow fr@a, fr@b, fr@c. \\
\end{align*}
\]

7. The second process \( mt([b, c]) \) becomes:

\[\langle \text{not fr}@a, fr@b, fr@c \rangle\]

and it is suspended because \( \text{not fr}@a \) is not a default.
Execution Example (continued)

\[ mt([a, b]) \leftarrow fr@a, fr@b, \text{ not } fr@c. \]
\[ mt([b, c]) \leftarrow \text{ not } fr@a, fr@b, fr@c. \]
\[ mt([c, a]) \leftarrow fr@a, \text{ not } fr@b, fr@c. \]
\[ mt([a, b, c]) \leftarrow fr@a, fr@b, fr@c. \]

8. The third process \( mt([c, a]) \) becomes:
\[ \langle fr@a, \text{ not } fr@b, fr@c \rangle \]

9. The third process becomes
\[ \langle \text{ not } fr@b, fr@c \rangle \] with an assumption \( \{ fr@a \} \).

10. The third process becomes suspended because \( \text{ not } fr@b \) is not a default.
Execution Example (continued)

\[
mt([a, b]) \leftarrow fr@a, fr@b, \text{ not } fr@c.
\]
\[
mt([b, c]) \leftarrow \text{ not } fr@a, fr@b, fr@c.
\]
\[
mt([c, a]) \leftarrow fr@a, \text{ not } fr@b, fr@c.
\]
\[
mt([a, b, c]) \leftarrow fr@a, fr@b, fr@c.
\]

11. The fourth process \(mt([a, b, c])\) becomes:

\[
\langle fr@a, fr@b, fr@c \rangle
\]

and it would be succeeded if there were no returned answers.

12. Suppose that \(fr\) is returned from \(a\):

Nothing changes!!

\(\Rightarrow\) Effect of Speculative Computation
Execution Example (continued)

\[ \text{mt}([a, b]) \leftarrow \text{fr}@a, \text{fr}@b, \text{not fr}@c. \]
\[ \text{mt}([b, c]) \leftarrow \text{not fr}@a, \text{fr}@b, \text{fr}@c. \]
\[ \text{mt}([c, a]) \leftarrow \text{fr}@a, \text{not fr}@b, \text{fr}@c. \]
\[ \text{mt}([a, b, c]) \leftarrow \text{fr}@a, \text{fr}@b, \text{fr}@c. \]

13. Suppose that \text{not fr} is returned from \( b!! \)

- The fourth process \( \langle \text{fr}@a, \text{fr}@b, \text{fr}@c \rangle \) is deleted
- and the third process \( \langle \text{not fr}@b, \text{fr}@c \rangle \) is resumed. \textbf{(Backup Mechanism)}
- and if there is no further communication, \( R = sr, L = [c, a] \)
  will be returned with the assumption \( \{ \text{fr}@c \} \).
Correctness of the Procedure

**Theorem** If \( \mathcal{P} \) is a stratified program, the output is true in the perfect model semantics of the program and returned answers and assumptions.

For any strategy of process reduction, the output is correct for the program reflecting returned answers and assumptions.
Conclusion

• A proposal of a speculative computation in multi-agent systems
• A proof procedure for a speculative computation with dynamic belief revision
• Correctness of the proof procedure
Extensions So Far

• Speculative Constraint Processing
• Multiagent Answer Revision
• Back-up Mechanism Considering Side-effects of Action
References


