Hierarchical Constraint Logic Programming for Multi-Agent Systems

Hiroshi Hosobe[1][a] and Ken Satoh[2]
[1]*Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan*
[2]*National Institute of Informatics, Tokyo, Japan*
hosobe@acm.org, ksatoh@nii.ac.jp

Keywords: Multi-Agent System, Logic Programming, Constraint.

Abstract: Logic programming is a powerful tool for modeling and processing multi-agent systems (MASs), where problems are collaboratively solved by multiple agents that exchange messages. Especially, (constraint) logic programming with default rules is useful for speculative computation for MASs, where tentative solutions are computed before answers arrive from other agents. However, the previous frameworks became complicated to enable speculative computation for MASs. In this paper, we propose a framework using hierarchical constraint logic programming (HCLP) for speculative computation for MASs. HCLP is an extension of constraint logic programming that allows soft constraints with hierarchical preferences. We simplify our MAS framework by utilizing HCLP and soft constraints to handle MASs with default rules. We show a prototype implementation of our framework and a case study on its execution.

1 INTRODUCTION

Multi-agent systems (*MASs*) have long been being studied in the field of artificial intelligence. A MAS is defined as "a loosely coupled network of problem solvers that interact to solve problems that are beyond the individual capabilities or knowledge of each problem solver" (Sycara, 1998). Such cooperative problem solvers are called agents. Research on MASs typically is focused on how to construct agents for given problems. Applications of MASs range over broad areas including computer networks, robotics, complex system modeling, city and building management, and smart grids (Dorri et al., 2018).

Logic programming (*LP*) (Lloyd, 1987), which uses formal logic for programming, and *constraint logic programming* (*CLP*) (Jaffar and Lassez, 1987; Jaffar and Maher, 1994), which extends LP with the notion of constraints, can be powerful bases for modeling and processing MASs. An extension of LP is suitable for the reactive component of a MAS as well as for the rational component of a single agent (Kowalski and Sadri, 1999). Also, an extension of CLP allows building MASs with constraint-solving capabilities (Vlahavas, 2002).

A MAS is usually designed to normally work in a situation with no problem of communication among agents. However, in actual situations, it is difficult to always guarantee efficient and reliable communication among agents. For example, if a MAS is deployed on a network like the internet that does not guarantee reliable communication, or if a MAS involves a human user as part of the MAS, it is possible that communication might largely delay or even fail.

Speculative computation for MASs is an effective means for coping with such problems. It enables MASs to compute tentative solutions in situations where communication among agents might largely delay or fail. Especially, LP and CLP with default rules are useful for speculative computation for MASs (Satoh et al., 2003; Satoh et al., 2000). In such situations, MASs speculatively compute tentative solutions, for which agents use default knowledge about other agents, instead of waiting for their answers.

However, the previous MAS frameworks became complicated to enable speculative computation. They were achieved by performing speculative computation and default rule handling within the same mechanisms. In other words, they introduced specialized mechanisms for operating such default rules and answers, which required more complicated mechanisms to enable more powerful MASs.

In this paper, we propose a framework for speculative computation for MASs by using an extension of CLP. The CLP extension that we use is *hierarchical CLP* (*HCLP*) (Wilson and Borning, 1993), which allows soft constraints with hierarchical preferences

[a] https://orcid.org/0000-0002-7975-052X

© 2023 SCITEPRESS
ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.

(i.e., constraint hierarchies (Borning et al., 1992)). Our main point is that we simplify our MAS framework by utilizing HCLP and soft constraints to handle MASs with default rules. Instead of introducing a specialized mechanism for operating default rules and answers in MASs, we express them by using the hierarchical preferences of soft constraints in HCLP, which enables us to treat default rules and answers declaratively rather than operationally. We present how to specify MASs in HCLP and how to perform speculative computation for such MASs. Also, we show a prototype implementation of our framework and a case study on its execution.

2 RELATED WORK

Researchers proposed frameworks and methods using LP for speculative computation for MASs. A logical framework based on abduction for speculative computation for master-slave MASs was first proposed (Satoh et al., 2000). Then this framework was extended to tree-structured MASs to enable agents to revise answers that they previously returned (Satoh and Yamamoto, 2002). Also, a framework using abduction was proposed (Satoh, 2005) and applied to orientation systems (Ramos et al., 2015a; Ramos et al., 2015b; Ramos et al., 2016; Ramos et al., 2014) and a decision support system (Oliveira et al., 2014). In addition, researchers proposed a framework that translates programs by assigning time stamps to predicates (Sakama et al., 2000), approaches using consequence finding (Inoue et al., 2001; Inoue and Iwanuma, 2004; Iwanuma and Inoue, 2002), a method that combines action execution (Hayashi et al., 2002), a method using defeasible logic (Lam et al., 2012), and a method using Bayesian networks (Gomes et al., 2016).

To treat more general problems than those LP-based frameworks and methods, a framework using CLP for speculative computation for master-slave MASs was proposed (Hosobe et al., 2007; Satoh et al., 2003). This framework was applied to a reasoning module for distributed clinical decision support systems (Oliveira et al., 2015). In addition, it was extended to enable agents to revise previous answers and also to return disjunctive answers (Ceberio et al., 2006). This extended framework was implemented in a practical multi-threaded manner (Ma et al., 2010b). Furthermore, it was extended to allow tree-structured MASs (Hosobe et al., 2010). Also, there have been more different CLP-based approaches to speculative computation for MASs including one using abductive reasoning (Ma et al., 2010a) and another using Bayesian networks (Oliveira et al., 2017). However, those CLP-based frameworks became complicated to treat default rules for speculative computation for MASs.

3 PRELIMINARIES

This section describes constraint hierarchies and HCLP as preliminaries of our framework.

3.1 Constraint Hierarchies

In constraint hierarchies (Borning et al., 1992), each constraint is associated with a hierarchical preference called a *strength*. A constraint hierarchy is divided into a finite number of levels. Each level consists of constraints with the equal strength corresponding to the level. The top level contains required (or hard) constraints that must always be satisfied, and lower levels contain preferential (or soft) constraints that should be relaxed if necessary. The strengths of constraints are usually represented symbolically as required, strong, medium, and weak. In this paper, we omit the explicit specification of required, expressing required constraints as ones without the specification of strengths.

Solutions to constraint hierarchies are obtained by the optimization concerning solution candidates. For this purpose, a relation called comparator is used to judge which of two solution candidates is better. Solution sets differ according to comparators, and there have been various comparators proposed. In this paper, we use the comparator called unsatisfied-count-better (UCB) that evaluates solution candidates to increase satisfied stronger constraints.

3.2 HCLP

HCLP (Wilson and Borning, 1993) is an extension of CLP (Jaffar and Lassez, 1987; Jaffar and Maher, 1994) that incorporates constraint hierarchies. Rules in HCLP hold constraints in the bodies of the Horn clauses in the same way as CLP, and furthermore constraints can be associated with strengths. The derivation of a goal (that is a successful sequence of reductions) is performed in the same way as CLP, and the solution is computed by solving the constraint hierarchy obtained when the goal becomes empty of atoms.

In general, derivations for different branches obtain different constraint hierarchies that also have constraints with different strengths. Therefore, HCLP obtains the best solutions by further performing inter-

© 2023 SCITEPRESS
ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.

hierarchy comparison that judges which of two constraint hierarchies is better satisfied.

4 PROPOSED FRAMEWORK

In this section, we propose a framework that uses HCLP for speculative computation for MASs. We first propose how to specify MASs in HCLP and then how to perform speculative computation for MASs.

4.1 Specification of MASs

Our framework uses HCLP as a basis for the specification of MASs. A MAS forms a tree structure, and the agent corresponding to the root node is called the root agent. The root agent starts the execution of the MAS, and returns a tentative solution whenever it is obtained by speculative computation. The users who give information to a MAS in execution are also regarded as agents, and correspond to the leaf nodes of the tree. The agents representing the users are called the external agents, and the other agents are called the internal agents.

Formally, the structure of a MAS is specified with an *agent hierarchy* in the same way as the previous work on hierarchical agents.

Definition 1 (agent hierarchy (Hosobe et al., 2010))**.** An agent hierarchy H is a tree consisting of a set of nodes called agents. Let $\mathrm{root}(H)$ be the root node of H, called the root agent. Let $\mathrm{int}(H)$ be the set of all the non-leaf nodes of H, each called an internal agent. Let $\mathrm{ext}(H)$ be the set of all the leaf nodes of H, each called an external agent. Given an internal agent M, let $\mathrm{chi}(M,H)$ be the set of all the child nodes of M, each called a child agent of M. Given a non-root agent S, let $\mathrm{par}(S,H)$ be the parent node of S, called the parent agent of S.

Figure 1 depicts the agent hierarchy of a MAS consisting of five agents $r,a,b,a',$ and b'. Here r is the root agent, r,a, and b are the internal agents, and a' and b' are the external agent. Also, a and b are the child agents of r (and r is the parent agent of a and b), a' is the child agent of a (and a is the parent of a'), and b' is the child agent of b (and b is the parent of b').

Agents are assigned HCLP programs extended in the following way.

- Each internal agent has a program consisting of rules that describe how it is executed, and of rules that describe the default knowledge of its child agents.

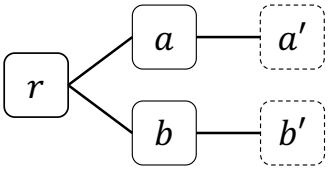

Figure 1: A MAS with a root agent r, non-root internal agents a and b, and external agents a' and b'.

- Each external agent initially has no program. To reply to a query from its parent agent, it is dynamically assigned a rule that is generated by the corresponding user according to the user's own knowledge. In addition, the program can be dynamically revised when the user's knowledge changes.

For each agent M, a rule in the program of M is a Horn clause described with a head H, required and/or preferential constraints C, and atoms $B_1,B_2,\ldots,B_n$ in the following form:

$$H \leftarrow C \,||\, B_1,B_2,\ldots,B_n$$

Each atom is expressed as either $p(t_1,t_2,\ldots,t_n)@M$ called a *non-askable atom* or $p(X_1,X_2,\ldots,X_n)@S$ called an *askable atom*, where S is a child agent of M, p is a predicate symbol, $t_1,t_2,\ldots,t_n$ are terms, and $X_1,X_2,\ldots,X_n$ are variables. Each constraint is either required or preferential.

The problem that should be solved concerning a MAS is called a goal, which is expressed as an askable atom $p(X_1,X_2,\ldots,X_n)@r$ for the root agent r. A solution to the problem is expressed as a valuation of variables $X_1,X_2,\ldots,X_n$. In general, there exist multiple solutions to a problem, and the set of all the solutions is called the solution set.

We use preferential constraints to express default knowledge. Typically, we use weaker constraints for agents closer to the root, using the strongest constraints for the external agents.

Formally, a MAS is defined with an agent hierarchy and a set of the programs associated with the agents in the hierarchy.

Definition 2 (MAS)**.** A MAS is a pair $\langle H,\mathcal{P}\rangle$, where H is an agent hierarchy, and $\mathcal{P}$ is a set of programs P_M of $M \in \mathrm{int}(H)\cup\mathrm{ext}(H)$.

Now we show an example of describing a MAS by using our framework. It reserves a twin or single room by checking the availabilities of two users, which we construct by rewriting the example given in the previous work (Hosobe et al., 2010).

Example 1. Consider a MAS $\langle H,\mathcal{P}\rangle$, where H is the agent hierarchy depicted in Figure 1, and $\mathcal{P}$ is the set of the following programs $P_r,P_a,P_b,P_{a'}$, and $P_{b'}$.

- The programs P_r,P_a, and P_b of the internal agents r,a, and b respectively are defined as follows.

© 2023 SCITEPRESS
ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.

- Program P_r of r:
  $\mathsf{reserve}(R,L,D)@r \leftarrow$
  $\quad R = \mathsf{twin\_room}, L = [a,b] \,||$
  $\quad \mathsf{available}(D)@a, \mathsf{available}(D)@b$
  $\mathsf{reserve}(R,L,D)@r \leftarrow$
  $\quad R = \mathsf{single\_room}, L = [a] \,||$
  $\quad \mathsf{available}(D)@a, \mathsf{unavailable}(D)@b$
  $\mathsf{reserve}(R,L,D)@r \leftarrow$
  $\quad R = \mathsf{single\_room}, L = [b] \,||$
  $\quad \mathsf{unavailable}(D)@a, \mathsf{available}(D)@b$
  $\mathsf{available}(D)@a \leftarrow \mathsf{weak}\ D \in \{1,2,3\} \,||$
  $\mathsf{available}(D)@b \leftarrow \mathsf{weak}\ D \in \{1,2,3\} \,||$
- Program P_a of a:
  $\mathsf{available}(D)@a \leftarrow \,||\, \mathsf{free}(D)@a'$
  $\mathsf{unavailable}(D)@a \leftarrow \,||\, \mathsf{busy}(D)@a'$
  $\mathsf{free}(D)@a' \leftarrow \mathsf{medium}\ D \in \{1,2\} \,||$
  $\mathsf{busy}(D)@a' \leftarrow \mathsf{medium}\ D \in \{3\} \,||$
- Program P_b of b:
  $\mathsf{available}(D)@b \leftarrow \,||\, \mathsf{free}(D)@b'$
  $\mathsf{unavailable}(D)@b \leftarrow \,||\, \mathsf{busy}(D)@b'$
  $\mathsf{free}(D)@b' \leftarrow \mathsf{medium}\ D \in \{2\} \,||$

- The programs $P_{a'}$ and $P_{b'}$ of the external agents a' and b' respectively indicate their answers, and are defined with the rules with strong constraints. For example, $P_{a'}$ is defined as $\emptyset$ (i.e., no answers), and $P_{b'}$ is defined as follows.
  - Answer $P_{b'}$ of b':
    $\mathsf{free}(D)@b' \leftarrow \mathsf{strong}\ D \in \{2,3\} \,||$

The intuitive meaning of this example is as follows. Agents a' and b' are the external ones corresponding to the two users. Agents a and b are the internal ones that automatically answer to a posed query by using their default knowledge as well as by asking a' and b' about their actual availabilities respectively. Agent r is the root that processes the entire coordination by asking a and b about their availabilities. The programs of r, a, and b and the answer of b' can be understood as follows.

- Program P_r of r reserves a twin room for the day when both a and b are available, and reserves a single room for the day when only one of them is available. In addition, it has default knowledge that a and b are available on days 1, 2, and 3.
- Program P_a of a answers the availability of a' to the query of r. In addition, it has default knowledge that a' is free on days 1 and 2 and is busy on day 3.
- Program P_b of b answers the availability of b' to the query of r. In addition, it has default knowledge that b' is free on day 2.
- Answer $P_{b'}$ of b' replies to the query of b that this user is free on days 2 and 3.

1: $P \leftarrow P_r$
2: $\langle \Theta, \Phi \rangle \leftarrow \mathrm{solve}(Q@r, P, \emptyset)$
3: Output Θ as the initial answer
4: **loop**
5: Await an answer P' from a child
6: $P \leftarrow P \cup P'$
7: $\langle \Theta, \Phi \rangle \leftarrow \mathrm{solve}(Q@r, P, \Phi)$
8: Output Θ as a revised answer
9: **end loop**

Figure 2: Algorithm of root agent r with program P_r for processing goal $Q@r$.

This example uses strong, medium, and weak as the strengths of the preferential constraints in the default knowledge, more specifically using strong for a' and b', medium for a and b, and weak for r. It makes the MAS treat the knowledge of the actual users a' and b' by the highest priority, the default knowledge of a and b working for the individual users by the next highest priority, and the default knowledge of r working for the entire coordination by the lowest priority.

4.2 Speculative Computation for MASs

We present how to perform speculative computation for MASs in our framework. The execution is started by submitting a goal to the root agent of a MAS. During the execution, internal agents send askable atoms to their child agents when necessary, and the children answer to their parents by returning the necessary rules for the derivation of the askable atoms.

Given a MAS $\langle H, \mathcal{P} \rangle$, the root agent $r = \mathrm{root}(H)$ is executed basically in the same way as ordinary HCLP. The main difference is that it revises its program P with the progress of the execution of the MAS. The execution of the MAS is started by submitting a goal $Q@r$ to r. Initially, P is the same as the program P_r of r. After computing the solution set of P for $Q@r$, r outputs it as the initial answer, and moves to the waiting state. When r receives an answer from a child agent of it, it updates P by adding the answer, recomputes the solution set of P for $Q@r$, outputs it as a revised answer, and moves to the waiting state again. Figure 2 shows this algorithm.

When root agent r computes a solution set, it processes each atom as follows.

- If the atom is a non-askable one for r, it is processed in the same way as ordinary HCLP.
- If the atom is an askable one, $p(X_1, X_2, \ldots, X_n)@a$, for a child agent S of r, it is processed as follows.
  1. If r has not yet processed this askable atom, r sends it to S.

© 2023 SCITEPRESS
ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.

1: $\Phi' \leftarrow \Phi$
2: $\Psi \leftarrow \emptyset$
3: **for** each "$Q@r \leftarrow C \,||\, B_1, \ldots, B_n$" in P **do**
4: $C' \leftarrow C$
5: **for** $i \leftarrow 1$ to n **do**
6: **if** B_i is an askable atom $Q@S$ for a child S of r and $Q@S \notin \Phi'$ **then**
7: Send $Q@S$ to S
8: $\Phi' \leftarrow \Phi' \cup \{Q@S\}$
9: **end if**
10: Update C' by processing B_i in the same way as ordinary HCLP
11: **end for**
12: $\Psi \leftarrow \Psi \cup \{C'\}$
13: **end for**
14: Compute the best solutions Θ to Ψ by performing the inter-hierarchy comparison
15: **return** $\langle \Theta, \Phi' \rangle$

Figure 3: Algorithm solve$(Q@r, P, \Phi)$ of root agent r for computing the solution set of program P for goal $Q@r$ and processed askable atoms Φ.

2. The atom is processed in the same way as ordinary HCLP.

After completing the derivations for all the branches, r computes the best solutions by performing the inter-hierarchy comparison. Figure 3 shows this algorithm, where Φ indicates the set of the processed askable atoms.

A non-root internal agent $M \in \mathrm{int}(H) \setminus \{\mathrm{root}(H)\}$ is normally in the waiting state. When it receives an askable atom from its parent agent $\mathrm{par}(M)$, it collects all the necessary rules for the derivation and sends them as an answer to the parent. In addition, if the collected rules include an askable atom for a child agent of $S \in \mathrm{chi}(M)$, it sends the askable atom to the child. When it receives an answer from the child, it further returns the answer to the parent. Figure 4 shows this algorithm.

When external agent $S \in \mathrm{ext}(H)$ receives an askable atom from its parent agent $\mathrm{par}(S)$, it dynamically generates a rule based on its own knowledge, and returns the rule as an answer to the parent. It is not necessary to immediately return the answer. Also, if it wants to revise the answer that it previously returned, it generates a new rule by using a stronger preferential constraint, and returns to the parent.

5 IMPLEMENTATION

We implemented a prototype MAS processor based on our framework in the Scala language. In addition,

1: $\Phi \leftarrow \emptyset$
2: **loop**
3: Await an askable atom $Q@M$ from $\mathrm{par}(M)$ or an answer P' from $S \in \mathrm{chi}(M)$
4: **if** $Q@M$ was received from $\mathrm{par}(M)$ **then**
5: Collect all the necessary rules $P \subseteq P_M$ for the derivation of $Q@M$
6: **for** each "$Q@M \leftarrow C \,||\, B_1, \ldots, B_n$" in P **do**
7: **for** $i \leftarrow 1$ to n **do**
8: **if** query B_i is an askable atom $Q@S$ for $S \in \mathrm{chi}(M)$ and $Q@S \notin \Phi$ **then**
9: Send $Q@S$ to S
10: $\Phi \leftarrow \Phi \cup \{Q@S\}$
11: **end if**
12: **end for**
13: **end for**
14: Send P to $\mathrm{par}(M)$
15: **else** // P' was received from S
16: Send P' to $\mathrm{par}(M)$
17: **end if**
18: **end loop**

Figure 4: Algorithm of a non-root internal agent M with program P_M for processing askable atom $Q@M$.

we constructed an algorithm for solving constraint hierarchies that is necessary for processing HCLP programs, and implemented the solver by using the satisfiability modulo theories (SMT) solver Z3 (de Moura and Bjørner, 2008). The MAS processor and the constraint hierarchy solver consist of approximately 700 lines of code in total.

6 CASE STUDY

In this section, we illustrate the execution of the MAS in Example 1 as a case study on how our framework works. It is started with a goal $\mathsf{reserve}(R, L, D)@r$ for root agent r (Figure 5(a)).

Initially, agent r holds only P_r as the internal program P. It obtains the following initial tentative solution set by processing this P, which is an effect of speculative computation (Figure 5(b)).

```
R=twin_room, L=[a,b], D=1
R=twin_room, L=[a,b], D=2
R=twin_room, L=[a,b], D=3
```

These were obtained from the default knowledge of r meaning that both a and b are available on days 1, 2, and 3. Also, note that, during this process, r sent askable atoms $\mathsf{available}(D)@a$ and $\mathsf{unavailable}(D)@a$ to its child agent a, and $\mathsf{available}(D)@b$ and $\mathsf{unavailable}(D)@b$ to its child agent b.

© 2023 SCITEPRESS
ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.

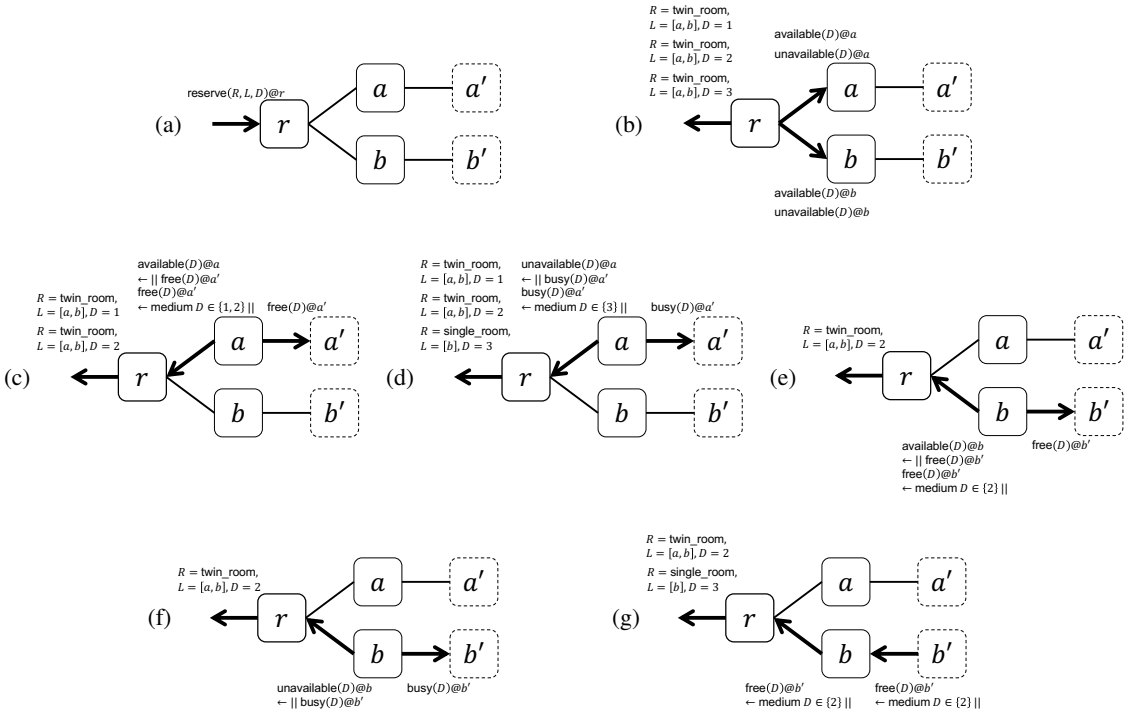

Figure 5: Execution of the MAS in Example 1.

Next, suppose that agent a returns rules $\mathsf{available}(D)@a \leftarrow ||\,\mathsf{free}(D)@a'$ and $\mathsf{free}(D)@a' \leftarrow \mathsf{medium}\ D \in \{1,2\}\,||$ by answering to the query $\mathsf{available}(D)@a$ of r. Then r updates P by merging these rules, and obtains the following revised tentative solution set by processing the updated P (Figure 5(c)).

```
R=twin_room, L=[a,b], D=1
R=twin_room, L=[a,b], D=2
```

These reflect part of the default knowledge of a meaning that a' is free on days 1 and 2. Note that this revised answer dropped the reserve of a room for day 3 because $\mathsf{free}(D)@a' \leftarrow \mathsf{medium}\ D \in \{1,2\}\,||$ (originally in P_a) has a stronger constraint than $\mathsf{available}(D)@a \leftarrow \mathsf{weak}\ D \in \{1,2,3\}\,||$ (originally in P_r). Also, note that a sent askable atom $\mathsf{free}(D)@a'$ to its child agent a'.

Next, suppose that agent a returns rules $\mathsf{unavailable}(D)@a \leftarrow ||\,\mathsf{busy}(D)@a'$ and $\mathsf{busy}(D)@a' \leftarrow \mathsf{medium}\ D \in \{3\}\,||$ by answering to the query $\mathsf{unavailable}(D)@a$ of r. Then r updates P and obtains the following revised solution set (Figure 5(d)).

```
R=twin_room, L=[a,b], D=1
R=twin_room, L=[a,b], D=2
R=single_room, L=[b], D=3
```

These reflect other part of the default knowledge of a meaning that a' is busy on day 3, introducing the reserve of a single room for day 3. Also, note that a sent askable atom $\mathsf{busy}(D)@a'$ to a'.

Next, suppose that agent b returns rules $\mathsf{available}(D)@b \leftarrow ||\,\mathsf{free}(D)@b'$ and $\mathsf{free}(D)@b' \leftarrow \mathsf{medium}\ D \in \{2\}\,||$ by answering to the query $\mathsf{available}(D)@b$ of r. Then r updates P and obtains the following revised solution (Figure 5(e)).

```
R=twin_room, L=[a,b], D=2
```

This reflects the default knowledge of b meaning that b' is free on day 2. Note that this answer dropped the reserve of rooms for days 1 and 3 because $\mathsf{free}(D)@b' \leftarrow \mathsf{medium}\ D \in \{2\}\,||$ (originally in P_b) has a stronger constraint than $\mathsf{available}(D)@b \leftarrow \mathsf{weak}\ D \in \{1,2,3\}\,||$ (originally in P_r). Also, note that b sent askable atom $\mathsf{free}(D)@b'$ to its child agent b'.

Next, suppose that agent b returns rule $\mathsf{unavailable}(D)@b \leftarrow ||\,\mathsf{busy}(D)@b'$ by answering to the query $\mathsf{unavailable}(D)@b$ of r. Then r updates P and obtains the following solution (Figure 5(f)).

```
R=twin_room, L=[a,b], D=2
```

This is the same as the previous one because b does not have default knowledge about $\mathsf{busy}(D)@b'$. Note that b sent askable atom $\mathsf{busy}(D)@b'$ to b'.

Finally, suppose that agent b' returns $\mathsf{free}(D)@b' \leftarrow \mathsf{strong}\ D \in \{2,3\}\,||$ by answering the query $\mathsf{free}(D)@b'$ of b, which follows that b

© 2023 SCITEPRESS
ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.

further sends it to r. Then r updates P and obtains the following solution (Figure 5(g)).

```
R=twin_room, L=[a,b], D=2
R=single_room, L=[b], D=3
```

These reflect the own knowledge of b' meaning that this user is free on days 2 and 3.

7 DISCUSSION

Our main point in this work was to devise a simplified framework for speculative computation for MASs. As described in Section 1, the previous frameworks became complicated. It should be noted that this complication caused other problems. One obvious problem is that the resulting frameworks were difficult to implement. Another problem is that it was difficult to theoretically investigate properties of the resulting frameworks; in fact, the proof of the correctness of the algorithm associated with such a framework was a huge task (see, e.g., the appendix of (Hosobe et al., 2010), which provides the proofs of the soundness and the completeness of the given algorithm). Another problem is that it is to difficult to devise a more powerful framework by extending such a complicated framework. The complication occurred because the previous frameworks were achieved by performing speculative computation and default rule handling within the same mechanisms.

To solve the complication problem, we adopted a different approach. We separated speculative computation and default rule handling to devise our framework. For default rule handling, we based our framework on HCLP instead of CLP, which was used in several studies on speculative computation for MASs (Ceberio et al., 2006; Hosobe et al., 2007; Hosobe et al., 2010; Ma et al., 2010b; Satoh et al., 2003). By this, we realized default rule handling declaratively rather than operationally, which is a primary cause of the simplification. To enable speculative computation for MASs, we constructed a computational mechanism where agents in a hierarchical MAS progressively exchange necessary information to speculatively compute tentative answers. The resulting computational mechanism also is sufficiently simple, and it is easy to understand what operations the agents perform in the course of speculative computation. Also, it should be noted that our framework treats hierarchical MASs and default constraints in a similar way to the work (Hosobe et al., 2010), which was one of the most advanced (and complicated) frameworks for speculative computation for MASs.

We consider that the expressive power of our framework is higher than those of the previous CLP-based MAS frameworks. This is because our framework is based on HCLP, which allows us to more freely associate preferences with default constraints, unlike the previous CLP-based frameworks that embedded default constraint handling in their specialized computational mechanisms. However, we have not yet clarified how our framework is more powerful than the previous ones. To explore the expressive power of our framework, it is necessary to experimentally explore how preferences of constraints can be utilized to model and process various MASs. Also, it is necessary to theoretically clarify relations of our framework with the previous CLP-based one for speculative computation for MASs, which needs to formally present and prove theorems about the relations. In addition, it is necessary to investigate how our framework is related with other previous frameworks than the CLP-based ones.

8 CONCLUSIONS AND FUTURE WORK

We proposed an HCLP-based framework for speculative computation for MASs. While the previous frameworks were based on CLP and were complicated, we used HCLP with constraint hierarchies to express the default knowledge of agents and to realize speculative computation for MASs, which simplified our framework. We also presented the prototype implementation of our framework and illustrated the execution of an example based on our framework.

Our future work is to explore the expressive power of our framework. More specifically, we will experimentally investigate the modeling and processing of various and more complex MASs in our framework. Also, we will clarify relations of our framework with previous CLP-based frameworks for speculative computation for MASs, for which we will formally present and prove theorems about the relations. In addition, we will improve the speculative computation of our framework. For this purpose, by incorporating incremental computation as in previous related work, we will improve the efficiency of recomputing solutions after obtaining tentative ones.

ACKNOWLEDGMENT

This work was supported by JST AIP Trilateral AI Research Grant Number JPMJCR20G4.

© 2023 SCITEPRESS
ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.

REFERENCES

Borning, A., Freeman-Benson, B., and Wilson, M. (1992). Constraint hierarchies. *Lisp Symbolic Comput.*, 5(3):223–270.

Ceberio, M., Hosobe, H., and Satoh, K. (2006). Speculative constraint processing with iterative revision for disjunctive answers. In *Post-proc. CLIMA*, volume 3900 of *LNAI*, pages 340–357.

de Moura, L. and Bjørner, N. (2008). Z3: An efficient SMT solver. In *Proc. TACAS*, volume 4963 of *LNCS*, pages 337–340.

Dorri, A., Kanhere, S. S., and Jurdak, R. (2018). Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593.

Gomes, M., Oliveira, T., and Novais, P. (2016). A speculative computation approach for conflict styles assessment with incomplete information. In *Proc. IDC*, volume 678 of *SCI*, pages 163–172.

Hayashi, H., Cho, K., and Ohsuga, A. (2002). Speculative computation and action execution in multi-agent systems. In *Proc. CLIMA*, volume 70-5 of *ENTCS*, pages 153–166.

Hosobe, H., Satoh, K., and Codognet, P. (2007). Agent-based speculative constraint processing. *IEICE Trans. Inf. & Syst.*, E90-D(9):1354–1362.

Hosobe, H., Satoh, K., Ma, J., Russo, A., and Broda, K. (2010). Speculative constraint processing for hierarchical agents. *AI Comm.*, 23(4):373–388.

Inoue, K. and Iwanuma, K. (2004). Speculative computation through consequence-finding in multi-agent environments. *Ann. Math. Artif. Intell.*, 42(1–3):255–291.

Inoue, K., Kawaguchi, S., and Haneda, H. (2001). Controlling speculative computation in multi-agent environments. In *Proc. CLIMA*, pages 9–18.

Iwanuma, K. and Inoue, K. (2002). Conditional answer computation in SOL as speculative computation in multi-agent environments. In *Proc. CLIMA*, volume 70-5 of *ENTCS*, pages 167–182.

Jaffar, J. and Lassez, J.-L. (1987). Constraint logic programming. In *Proc. ACM POPL*, pages 111–119.

Jaffar, J. and Maher, M. (1994). Constraint logic programming: A survey. *J. Log. Program.*, 19–20(Supp. 1):503–581.

Kowalski, R. and Sadri, F. (1999). From logic programming towards multi-agent systems. *Ann. Math. Artif. Intell.*, 25:391–419.

Lam, H.-P., Governatori, G., Satoh, K., and Hosobe, H. (2012). Distributed defeasible speculative reasoning in ambient environment. In *Proc. CLIMA*, volume 7486 of *LNAI*, pages 43–60.

Lloyd, J. W. (1987). *Foundations of Logic Programming*. Springer, 2nd edition.

Ma, J., Broda, K., Goebel, R., Hosobe, H., Russo, A., and Satoh, K. (2010a). Speculative abductive reasoning for hierarchical agent systems. In *Proc. CLIMA*, volume 6245 of *LNAI*, pages 49–64.

Ma, J., Russo, A., Broda, K., Hosobe, H., and Satoh, K. (2010b). On the implementation of speculative constraint processing. In *Post-proc. CLIMA*, volume 6214 of *LNAI*, pages 178–195.

Oliveira, T., Neves, J., Novais, P., and Satoh, K. (2014). Applying speculative computation to guideline-based decision support systems. In *Proc. IEEE CBMS*, pages 42–47.

Oliveira, T., Satoh, K., Novais, P., Neves, J., and Hosobe, H. (2017). A dynamic default revision mechanism for speculative computation. *Auton. Agent Multi-Agent Syst.*, 31(3):656–695.

Oliveira, T., Satoh, K., Novais, P., Neves, J., Leão, P., and Hosobe, H. (2015). A reasoning module for distributed clinical decision support systems. In *Proc. IDC*, volume 616 of *SCI*, pages 387–397.

Ramos, J., Novais, P., Satoh, K., Oliveira, T., and Neves, J. (2015a). Speculative orientation and tracking system. *Int. J. Artif. Intell.*, 13(1):94–119.

Ramos, J., Oliveira, T., Novais, P., Neves, J., and Satoh, K. (2015b). An alert mechanism for orientation systems based on speculative computation. In *Proc. INISTA*, pages 1–8. IEEE.

Ramos, J., Oliveira, T., Satoh, K., Neves, J., and Novais, P. (2016). Orientation system based on speculative computation and trajectory mining. In *Proc. PPAMS Workshops*, volume 616 of *CCIS*, pages 250–261.

Ramos, J., Satoh, K., Novais, P., and Neves, J. (2014). Modelling an orientation system based on speculative computation. In *Proc. DCAI*, volume 290 of *AISC*, pages 319–326.

Sakama, C., Inoue, K., Iwanuma, K., and Satoh, K. (2000). A defeasible reasoning system in multi-agent environments. In *Proc. CLIMA*, pages 1–6.

Satoh, K. (2005). Speculative computation and abduction for an autonomous agent. *IEICE Trans. Inf. & Syst.*, E88-D(9):2031–2038.

Satoh, K., Codognet, P., and Hosobe, H. (2003). Speculative constraint processing in multi-agent systems. In *Proc. PRIMA*, volume 2891 of *LNAI*, pages 133–144.

Satoh, K., Inoue, K., Iwanuma, K., and Sakama, C. (2000). Speculative computation by abduction under incomplete communication environments. In *Proc. ICMAS*, pages 263–270.

Satoh, K. and Yamamoto, K. (2002). Speculative computation with multi-agent belief revision. In *Proc. AAMAS*, pages 897–904.

Sycara, K. P. (1998). Multiagent systems. *AI Mag.*, 19(2):79–92.

Vlahavas, I. (2002). MACLP: Multi agent constraint logic programming. *Information Science*, 144:127–142.

Wilson, M. and Borning, A. (1993). Hierarchical constraint logic programming. *J. Log. Program.*, 16(3–4):227–318.

© 2023 SCITEPRESS

ICAART2023—https://icaart.scitevents.org/?y=2023
This is the author's version. The final authenticated version is available online at https://doi.org/10.5220/0011698000003393.