ILP-based Inference for Cost-based Abduction on First-order Predicate Logic

Naoya Inoue[†] and Kentaro Inui[†]

Abduction is desirable for many natural language processing (NLP) tasks. While recent advances in large-scale knowledge acquisition warrant applying abduction with large knowledge bases to real-life NLP problems, as of yet, no existing approach to abduction has achieved the efficiency necessary to be a practical solution for large-scale reasoning on real-life problems. In this paper, we propose an efficient solution for large-scale abduction. The contributions of our study are as follows: (i) we propose an efficient method of cost-based abduction in first-order predicate logic that avoids computationally expensive grounding procedures; (ii) we formulate the best-explanation search problem as an integer linear programming optimization problem, making our approach extensible; (iii) we show how cutting plane inference, which is an iterative optimization strategy developed in operations research, can be applied to make abduction in first-order logic tractable; and (iv) the abductive inference engine presented in this paper is made publicly available.

Key Words: Cost-based Abduction, Integer Linear Programming, Discourse Processing, Intepretation as Abduction

1 Introduction

Discovering implicit information from natural language discourse is essential to a wide range of NLP tasks, such as question answering, information extraction and recognizing textual entailment (RTE). Several NLP components for processing a variety of discourse phenomena (e.g. anaphora resolution) are exploited when inferring implicit information. In the field of computational linguistics, each NLP component has been studied extensively in recent decades; however, less attention has been paid to how to integrate them into a single inference framework.

In this paper, we explore *abduction*-based discourse processing as a framework for integrating NLP components. Abduction, defined here as inference to the best explanation, has long been studied in a wide range of contexts. For example, abduction has been viewed as a promising framework for describing the mechanism of human perception (Charniak and Goldman 1991; Hobbs, Stickel, Martin, and Edwards 1993; Shanahan 2005; Peraldi, Kaya, Melzer, Möller, and Wessel 2007). The idea is that the declarative nature of abduction enables us to infer the

[†]Tohoku University

most plausible, implicitly stated information combining several types of inference and pieces of explicitly observed information, as humans do.

Abduction-based discourse processing was studied intensively in the 1980s and 1990s; Hobbs et al. (1993) show that the lowest-cost abductive proof provides the solution to a broad range of natural language understanding problems, such as word sense disambiguation, anaphora, and metonymy resolution. As detailed in Sec. 2.1, the key advantage of using abduction for discourse processing is twofold:

- abduction-based discourse processing models interdependencies between NLP tasks and identifies the most coherent interpretation;
- it discovers plausible new information by combining heterogeneous inference rules and pieces of information observed from texts.

While the lack of world knowledge resources hampered applying abduction to real-life problems in the 1980s and 1990s, a number of techniques have been developed in the last decade (Fellbaum 1998; Chambers and Jurafsky 2009; Ruppenhofer, Ellsworth, Petruck, Johnson, and Scheffczyk 2010; Schoenmackers, Davis, Etzioni, and Weld 2010; Hovy, Zhang, Hovy, and Penas 2011). Consequently, several researchers started applying abduction to real-life problems, exploiting large knowledge bases. For instance, inspired by Hobbs et al. (1993), Ovchinnikova et al. (2011) proposed an abduction-based NLP framework using forty thousand axioms extracted from the popular ontological resources WordNet (Fellbaum 1998) and FrameNet (Ruppenhofer et al. 2010). They evaluate their approach on the real-life natural language processing task of RTE (Dagan, Dolan, Magnini, and Roth 2010).

However, in order to apply abduction to real-life problems with a large-scale knowledge base, we still need to address the following issue: *how to search for the best explanation efficiently.* In this paper, we adopt *first-order logic-based, cost-based abduction* (henceforth first-order cost-based abduction), where we use function-free first-order logic (FOL) as a representation language. In first-order cost-based abduction, an explanation is represented by a set of literals, and the plausibility of the explanation is evaluated through the sum of the costs defined on each literal. The best explanation is defined as the lowest-cost explanation. Finding the lowest-cost explanation can be reduced to a constrained combinatorial optimization problem with respect to the cost function, which is an NP-hard problem (Charniak and Goldman 1991); this hampers the application of abduction with large knowledge resources to real-life problems. In fact, Ovchinnikova et al. (2011) report that the Mini-TACITUS cost-based abduction system (Mulkar, Hobbs, and Hovy 2007) could not search the entire search space of explanations within 30 min in most of the RTE problems in their experiments.

In the literature, many researchers have tried to overcome cost-based abduction's inefficiency using a range of methods from approximation to exact inference (Poole 1993a; Santos 1994; Ishizuka and Matsuo 1998; Chivers, Tagliarini, and Abdelbar 2007). For example, Santos (1994) formulated cost-based abduction in propositional logic using integer linear programming (ILP), and showed its efficiency. However, to the best of our knowledge, most of the proposed methods are optimized for propositional logic. In order to employ these methods for first-order cost-based abduction, we need to transform knowledge bases and observations to propositional logic (henceforth, we call the transformation *grounding*). The process of grounding generates a huge search space and does not scale to larger problems, as discussed in Sec. 3.

In this paper, we provide a scalable solution to first-order cost-based abduction with the following contributions:

(i) we propose an efficient search technique for FOL abduction, which avoids expanding first-order logical formulae to propositional level;

(ii) we formulate the best-explanation finding problem as an ILP optimization problem to make our framework extensible, supporting definite clauses in background knowledge;

(iii) we describe how cutting plane inference (CPI), an iterative optimization strategy developed in operations research, can be exploited for making FOL abduction tractable, showing its efficiency by providing evaluation on a large, real-life dataset;

(iv) the abductive inference engine presented in this paper is made publicly available.[1]

Our paper is structured as follows. We start with a brief introduction of abduction-based discourse processing, taking Hobbs's interpretation as abduction framework (Hobbs et al. 1993) as a motivating example (Sec. 2). We then describe how to perform efficient searches in FOL abduction, and formulate the search problem as an ILP optimization problem (Secs. 3.1 and 3.2). We then show how CPI enables us to apply FOL abduction with large knowledge bases (Sec. 3.3). Finally, we evaluate the efficiency of our CPI-based framework on a large, real-life problem of NLP, RTE (Sec. 4), and give a comparison of our work to prior implementations of cost-based abduction (Sec. 5).

2 Background

2.1 Interpretation as Abduction

Hobbs et al. (1993) pioneered an abduction-based approach for natural language understand-

[1]https://github.com/naoya-i/henry-n700

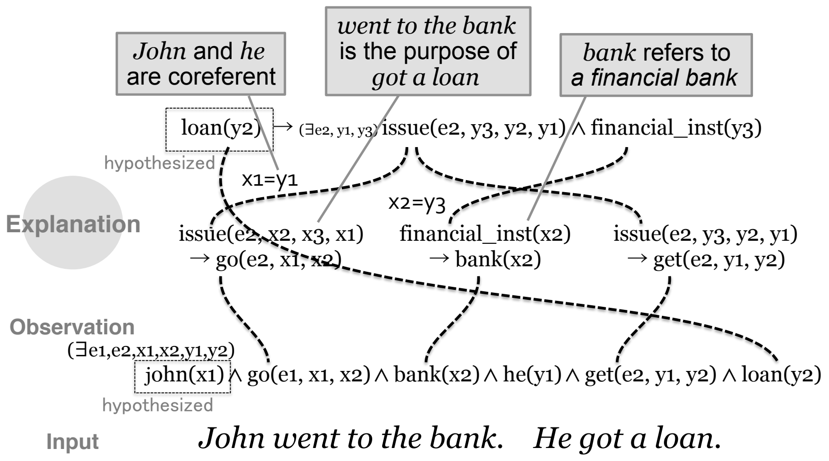

John and he are coreferent
went to the bank is the purpose of got a loan
bank refers to a financial bank
loan(y2) → (∃e2, y1, y3) issue(e2, y3, y2, y1) ∧ financial_inst(y3)
hypothesized
x1=y1
x2=y3
Explanation
issue(e2, x2, x3, x1) → go(e2, x1, x2)
financial_inst(x2) → bank(x2)
issue(e2, y3, y2, y1) → get(e2, y1, y2)
Observation
(∃e1,e2,x1,x2,y1,y2)
john(x1) ∧ go(e1, x1, x2) ∧ bank(x2) ∧ he(y1) ∧ get(e2, y1, y2) ∧ loan(y2)
hypothesized
Input
John went to the bank. He got a loan.

Fig. 1 Example of abductive interpretation.

ing. The key idea is that "*interpreting sentences is to prove the logical forms of sentences, allowing assumptions, merging redundancies where necessary.*" They demonstrated that a wide range of NLP tasks involved in discourse interpretation, including anaphora resolution and discourse relation recognition, can be cast as the problem of finding an explanation to the pieces of information observed from the discourse. Figure 1 shows an example taken from Hobbs et al. 1993. In this example, we solve three types of NLP tasks: (i) coreference resolution, e.g., the coreference relation between *John* and *he* ($x1 = y1$); (ii) intent recognition, e.g., the intention of *John* (backward inference on $go(e1, x1, x2)$ to $loan(y2)$); and (iii) word sense disambiguation, e.g., the meaning of *bank* is not a riverbank, but a financial institution (backward inference on *bank*($x2$) to *financial_inst*($x2$), where the inference rule means that "*a financial institution is expressed as bank in a text*").

While the lack of knowledge was a serious problem in applying abduction to real-life problems in the 1980s and 1990s, the situation has changed. A number of techniques that acquire lexical resources useful for language processing have been developed (Fellbaum 1998; Ruppenhofer et al. 2010; Chambers and Jurafsky 2009; Schoenmackers et al. 2010, etc.). Given the recent advances in techniques, several researchers started pursuing abduction-based approaches for "real-life" problems, using large knowledge bases. For instance, Ovchinnikova et al. (2011) took on the popular, knowledge-intensive NLP task of RTE. A series of studies in machine reading projects (Etzioni, Banko, and Cafarella 2006), which discover implicit information from texts (Penas and Hovy 2010; Hovy et al. 2011), can also be viewed as an abductive interpretation problem.

2.2 Cost-based abduction

Abduction is inference to the best explanation. We use function-free first-order logic as the meaning representation of abduction in this paper. Formally, first-order logical abduction is defined as follows:[2]

- **Given:** Background knowledge B and observations O, where B is a set of first-order logical formulae and O is a set of literals or equalities.
- **Find:** An *explanation* (or *hypothesis*) H such that $H \cup B \models O, H \cup B \not\models \perp$, where H is a set of literals or equalities. Each element in H is called an *elemental explanation.*

Let us define some terminologies. We define *equality* to be the form $x = y$ (*positive* equality) or $x \neq y$ (*negative* equality), where x and y are either variables or constants. The equality $x = y$ means that referents of x and y are the same (i.e. $\{p(x), p(y), x = y\}$ has the same meaning as $\{p(x)\}$). We say that the literal p is the *logical consequence* of S if $S \models p$; p is (*explicitly*) *hypothesized* w.r.t. H if $p \in H$; p is *implicitly hypothesized* if $H \cup B \models p$ w.r.t. H and B (i.e. p is a logical consequence of H w.r.t. B); p is *explained* if $H \cup B \setminus \{p\} \models p$; otherwise p is *assumed.* We refer to the operation that we unify two or more literals in set S of literals, and apply the unifier to S as *factoring* of S.

In this paper, we assume that all variables occurring in a logical form of background knowledge are *universally* quantified with the widest possible scope, unless it is explicitly stated as existentially quantified. On the other hand, we assume that variables occurring in an explanation and observation are implicitly *existentially* quantified. We assume that the background knowledge has no cyclic dependencies between an explaining and an explained literal (e.g., $B = \{P(x) \rightarrow Q(x), Q(x) \rightarrow P(x)\}$ has a cyclic dependency). We call this assumption *knowledge recursion-free assumption.*

Typically, several explanations H explaining O exist. We call each of them a *candidate explanation*, and represent the set of candidate explanations of O given B as $\mathcal{H}_{O,B}$. The goal of abduction is to find the best explanation among candidate explanations by a specific evaluation measure. In this paper, we formulate abduction as the task of finding the minimum-cost explanation $\hat{H}$ among $\mathcal{H}_{O,B}$. Henceforth, we refer to abduction based on the minimum-cost explanation finding as *cost-based abduction (CBA)*. Formally, we find $\hat{H} = \arg\min_{H \in \mathcal{H}_{O,B}} cost(H)$, where *cost* is a function $\mathcal{H}_{O,B} \rightarrow \mathbb{R}$, which is called the *cost function.*

Let us describe the task of abduction with a toy example. Given $B = \{p(x, y) \wedge q(x) \rightarrow r(x), s(x) \rightarrow r(x)\}, O = \{r(z)\}$, we have four candidate explanations: $H_1 = \{r(z)\}$, $H_2 =$

[2]The same framework is used in *induction.* While induction finds a set of plausible rules from observations, abduction finds a set of plausible facts.

$\{p(z,w), q(z)\}$, $H_3 = \{s(z)\}$, and $H_4 = \{p(z,w), q(z), s(z)\}$. The task of abduction is to select the best explanation among them in terms of cost. Suppose $cost(H_1) = 5.5$, $cost(H_2) = 12.25$, $cost(H_3) = 10.8$, and $cost(H_4) = 7.13$. Then, the correct prediction is then H_1.

It is crucial to discuss the specificity of explanations. We say that an explanation H is more *specific* than another explanation H' if $H \cup B \models H'$. As discussed in Hobbs et al. 1993, we want to decide the appropriate specificity of an explanation because there is often little evidence (i.e., observation) to support specific explanations. Traditionally, two extreme modes of abduction have been considered. The first is *most-specific abduction*. In most-specific abduction, what we can explain from background knowledge is all detailed, which is suitable for diagnostic systems. In diagnostic systems, users might want to know, as much as possible, about what has caused the current situation. Some cost-based and probabilistic approaches fall into this group (Charniak and Goldman 1991; Raghavan and Mooney 2010). The second is *least-specific abduction*. Literally, in this mode, an explanation is obtained by just assuming observations. Using only least-specific abduction has little purpose, but as described below, it makes sense if it is combined with most-specific abduction.

In natural language understanding systems, we need both modes at the same time. Adopting only one of these levels is problematic. For example, if we adopt most-specific abduction, the system yields too specific an explanation, such as "*Bob took a gun because he would rob XYZ bank using a machine gun which he had bought three days ago*." Conversely, if we adopt least-specific abduction, the system assumes just observation, as in "*Bob took a gun because he took a gun*." We thus want to determine the suitable specificity during inference. To the best of our knowledge, the weighted abduction of Hobbs et al. (1993) is the only framework that concerns the appropriateness of explanation specificity. The cost function of weighted abduction naturally handles this by propagating costs of propositions and unification as described in Sec. 3.2.

2.3 Cost function

In the literature, several types of cost functions have been proposed, including cost- and probability-based functions (Charniak and Goldman 1991; Poole 1993b; Hobbs et al. 1993; Raghavan and Mooney 2010; Singla and Mooney 2011, etc.). In this paper, we adopt the cost function proposed by Hobbs et al. (1993). The cost function assumes that each elemental explanation $p \in H$ has the *non-negative* cost of hypothesizing p (intuitively, the plausibility of p being an explanation for given observations), and sums up the costs of *assumed* elemental explanations. Henceforth, we write $P(x)^{\$c}$ to denote $P(x)$ having a cost c.

During the construction of H, one can *factor* H to generate a new explanation at any time.

When H is factored, the following things happen: (i) the literal that has the smallest cost among a set of unified literals remains in H, and (ii) for the unifier $\{x_i/y_i\}_{i=1}^{n}$, a set of elemental explanations $\{x_i = y_i^{\$0}\}_{i=1}^{n}$ is added to H. For example, one can factor $H = \{R(a)^{\$20}, R(b)^{\$10}, Q(a)^{\$20}\}$ with the unifier $\{a/b\}$ to get $H' = \{R(b)^{\$10}, a = b^{\$0}, Q(b)^{\$20}\}$, where the smaller cost \$10 is assigned to $R(b)$. Formally, the cost function is defined as follows:

$$cost(H) = \sum_{h \in A(H)} cost(h), \tag{1}$$

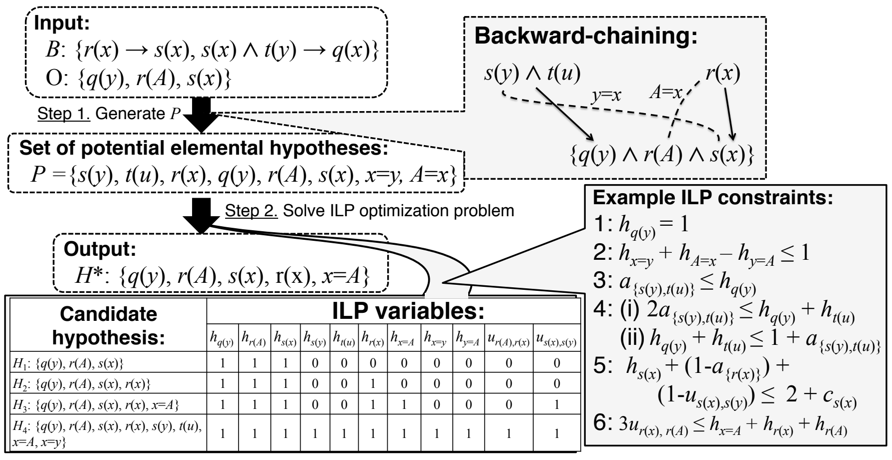
where $A(H)$ is a set of *assumed* literals in H.

In Hobbs et al. (1993), each cost is determined by two factors: (i) *weights*, the parameters assigned to axioms in background knowledge; and (ii) the costs of observations initially assigned. In this paper, we do not go into detail, and just assume that each cost is given in some way.

3 ILP-based inference for cost-based abduction

We now describe our strategy to find the best explanation in first-order CBA. The key idea is that we solve first-order CBA problems using the *lifted inference* technique, where each inference operation is directly performed on a first-order level, like resolution (Robinson 1965). In principle, this way of problem formulation gives us three benefits. First, we can reduce the search space of candidate explanations in comparison to a grounding approach, because we can avoid instantiating FOL formulae with all possible constants. Second, the best explanation finding problem can be reduced to the constrained combinatorial optimization problem of first-order literals and/or equalities, meaning that we can exploit several choices of combinatorial optimization technology developed in operations research. Specifically, our optimization problem can be naturally formulated as an ILP problem, which can be efficiently solved by existing ILP solvers. Third, the resulting framework is highly extensible; for example, we can easily incorporate linguistically motivated heuristics by simply adding some ILP variables and/or constraints to an optimization problem, keeping the overall framework unchanged.

In the rest of this section, we first formalize the best explanation finding in first-order CBA using the lifted inference technique, and then describe how to solve it as an ILP optimization problem.


Input:
B: $\{r(x) \rightarrow s(x), s(x) \wedge t(y) \rightarrow q(x)\}$
O: $\{q(y), r(A), s(x)\}$
Step 1. Generate P
Backward-chaining:
$s(y) \wedge t(u)$
$r(x)$
$y=x$
$A=x$
$\{q(y) \wedge r(A) \wedge s(x)\}$
Set of potential elemental hypotheses:
$P = \{s(y), t(u), r(x), q(y), r(A), s(x), x{=}y, A{=}x\}$
Step 2. Solve ILP optimization problem
Output:
H^*: $\{q(y), r(A), s(x), \mathrm{r(x)}, x{=}A\}$
Example ILP constraints:

1: $h_{q(y)} = 1$

2: $h_{x=y} + h_{A=x} - h_{y=A} \leq 1$

3: $a_{\{s(y),t(u)\}} \leq h_{q(y)}$

4: (i) $2a_{\{s(y),t(u)\}} \leq h_{q(y)} + h_{t(u)}$

(ii) $h_{q(y)} + h_{t(u)} \leq 1 + a_{\{s(y),t(u)\}}$

5: $h_{s(x)} + (1\text{-}a_{\{r(x)\}}) + (1\text{-}u_{s(x),s(y)}) \leq 2 + c_{s(x)}$

6: $3u_{r(x),\, r(A)} \leq h_{x=A} + h_{r(x)} + h_{r(A)}$

Candidate hypothesis:	ILP variables:										
	$h_{q(y)}$	$h_{r(A)}$	$h_{s(x)}$	$h_{s(y)}$	$h_{t(u)}$	$h_{r(x)}$	$h_{x=A}$	$h_{x=y}$	$h_{y=A}$	$u_{r(A),r(x)}$	$u_{s(x),s(y)}$
H_1: $\{q(y), r(A), s(x)\}$	1	1	1	0	0	0	0	0	0	0	0
H_2: $\{q(y), r(A), s(x), r(x)\}$	1	1	1	0	0	1	0	0	0	0	0
H_3: $\{q(y), r(A), s(x), r(x), x{=}A\}$	1	1	1	0	0	1	1	0	0	0	1
H_4: $\{q(y), r(A), s(x), r(x), s(y), t(u), x{=}A, x{=}y\}$	1	1	1	1	1	1	1	1	1	1	1

Fig. 2 Summary of the ILP-based approach.

3.1 Lifted first-order inference for CBA

First-order logic inherits all the theoretical properties of propositional logic, and hence a sound and complete inference can be performed on propositional logic. However, performing first-order logical inference on a propositional level has severe overhead, because we need *grounding*, which generates the ground instances of first-order logical formulae in knowledge bases and observations (i.e., instantiating them with all possible constants). The grounding procedure generates a lot of formulae when a domain is large. In this paper, we thus propose to perform cost-based abduction on a first-order level. The approach is in the spirit of resolution (Robinson 1965), but is applied to the best explanation finding problems. In the rest of this section, we show how to solve the abductive inference problem on first-order level.

Figure 2 summarizes our approach. In principle, our approach takes two steps: (i) Step 1: *search-space generation*, and (ii) Step 2: *best-explanation search*. In search-space generation, we first construct a set of all possible literals and/or equalities that are potentially included in H. For example, given the toy problem in Sec. 2.2, we construct the following set: $\{r(z), p(z, w), q(z), s(z)\}$. In the best-explanation search step, we find the best explanation for O by finding the best combination of literals or equalities among the set of literals constructed in the search-space generation step, according to the cost function. The problem is solved in the form of constrained boolean optimization problem, which is the problem of finding the truth assignment to boolean variables that maximizes or minimizes an objective function satisfying the given constraints.

Now we move on to the detail of our approach, in which we use the following format for background knowledge, observations, and explanation:

- Background knowledge: a set of first-order definite clauses (i.e. $p_1 \wedge p_2 \wedge ... \wedge p_n \rightarrow q$, where $p_1, p_2, ..., p_n$, and q are atoms);
- Observations: a set of positive literals or positive equalities;
- Explanation: a set of positive literals or positive equalities.

Henceforth, we call each clause in background knowledge an *axiom*, the right-hand side the *head*, and the left-hand side the *body*.

We give the overall algorithm in Algorithm 1. Given background knowledge B and observations O, we first create the set P of literals or equalities that are potentially included as constituents of the best explanation of O (line 2–10). We refer to the literal or equality $p \in P$ as the *potential elemental explanation*, which we enumerate by first initializing P with O. We then iteratively apply *backward inference* to each $p \in P$ (line 2–6). Algorithm 2 depicts the backward-inference operation in detail (lines 5–8). We define backward inference as the following operation:

- **Input:** a clause in the form $p_1 \wedge p_2 \wedge ... \wedge p_n \rightarrow q$ and the literal l, where there must exist the most general unifier θ such that $l\theta = q\theta$.
- **Output:** $\{p_1, p_2, ..., p_n\}\theta$, where the variables that are not substituted by θ (*notSubstitutedVars*$(\{p_1, p_2, ..., p_n\}, \theta)$ in Algorithm 2) are replaced with existentially quantified variables not appearing in P so far.

For example, given the axiom $p(x, y) \wedge q(x, y, z) \rightarrow r(x)$ and $r(a)$, it derives $\{p(a, u_1), q(a, u_1, u_2)\}$, where u_1 and u_2 are existentially quantified variables not appearing in P. Note that P is not equivalent to a set of resolvents that are generated by a particular proof procedure. The goal of proof procedure is to check whether a logical formula is implied by a set of logical formulae. Therefore, the derived proof might not contain a set of *all literals* that can explain observations. For example, SLD resolution (Kowalski 1974), is a backward-inference-based proof procedure that works on definite clauses, where literals resolved upon are selected by a particular computation rule (e.g., leftmost), and the resolution procedure terminates when the proof is found to be a failure or a success. However, what we want to enumerate is the set of all literals that can explain observations. Since we have the knowledge-recursion-free assumption, lines 2–11 terminate in finite time (i.e., until no more backward inference can be applied).

In lines 6–10, we search for the pairs of unifiable literals in P in order to represent the application of factoring operation to H. For each pair of unifiable literals, we add the equalities that are potentially hypothesized by the unifier (see Sec. 2.2). We do *not* unify such literals in P here because we want to allow that the factoring operations are also defeasible, "*possibly*" true operations. We use the cost function to determine whether they should be factored.

Algorithm 1 liftedFirstOrderCBA(Background knowledge B, Observation O, Cost function *cost*)

```
1: $P \leftarrow O$, $S \leftarrow O$
2: while $S \neq \phi$ do
3:   $S \leftarrow getPotentialElementalExplanations(B, S)$
4:   $P \leftarrow P \cup S$
5: end while
6: for $p_1, p_2 \in P$ do
7:   if $\exists \theta p_1 \theta = p_2 \theta$ then
8:     for $x/y \in \theta$ do $P \leftarrow P \cup \{x = y\}$
9:   end if
10: end for
11: return $findBestExplanation(P, cost)$
```

Algorithm 2 getPotentialElementalExplanations(Background knowledge B, set S of literals)

```
1: $R \leftarrow \{\}$
2: for $l \in S$ do
3:   for $p_1 \wedge p_2 \wedge ... \wedge p_n \rightarrow q \in B$ do
4:     if $\exists \theta l \theta = q \theta$ then
5:       for $v \in notSubstitutedVars(\{p_1, p_2, ..., p_n\}, \theta)$ do
6:         $\theta \leftarrow \theta \cup \{v/u_i\}$; $i \leftarrow i + 1$
7:       end for
8:       $R \leftarrow R \cup \{p_1, p_2, ..., p_n\}\theta$
9:     end if
10:   end for
11: end for
12: return $R$
```

In line 11, we find the best explanation. Given P, the problem of best explanation finding can be reduced to a constrained combinatorial optimization problem. Note that the number of candidate explanations grows exponentially (i.e. $O(2^{|P|})$), because each explanation is represented by the combination of potential elemental explanations. We immediately see that a simple approach that finds a minimal explanation by evaluating all candidate explanations is intractable. To improve efficiency, we formulate the best explanation finding as the 0-1 ILP optimization problem to exploit the state-of-the-art search strategy of combinatorial optimization problems. The formulation is described in the next section.

3.2 ILP Formulation

We formulate the best-explanation finding problem as an ILP optimization problem where the search space is represented as ILP variables and constraints, and the cost function is used as the ILP objective. Intuitively, for each $p \in P$, we introduce some 0-1 state variable that represents whether the potential elemental explanation p is (explicitly or implicitly) hypothesized. Then, every possible $H \in \mathcal{H}_{\mathcal{O},\mathcal{B}}$ can be expressed as a combination of value assignments to these state variables.

We elaborate on two types of ILP variables and ILP constraints in the optimization problem: (i) for representing the search space of candidate explanations, and (ii) for implementing the cost function.

Formulation for CBA search space: To represent whether the literal or equality $p \in P$ is hypothesized (including *implicitly* hypothesized), we introduce an ILP variable $h \in \{0, 1\}$ as follows:

$$\text{for each } p \in P : h_p = \begin{cases} 1 & \text{iff } H \cup B \models p; \\ 0 & \text{otherwise.} \end{cases}$$

For example, H_2 in Figure 2 holds $h_{r(x)} = 1$, where $r(x)$ is hypothesized in H_2. We also use h to represent equalities. In H_3, the variable $h_{x=A}$ is set to 1 because $x = A$ is assumed. Note that h variables do not represent the truth values of p (i.e. $h_p = 0$ does not mean $H \cup B \models \neg p$). Once a value assignment to h is determined, we construct H on the basis of the assignment as follows:

Definition 3.1 Given a particular value assignment to h variables, we generate an explanation H as follows:

- $p \in H \Leftrightarrow h_p = 1$ for each $p \in P$;
- $p \notin H \Leftrightarrow h_p = 0$ for each $p \in P$.

That is, *all* logical consequences of $H \cup B$ are considered to be an explanation (i.e., the generated explanation includes *implicitly*, as well as *explicitly*, hypothesized literals).

Note that not all value assignments to ILP variables h are allowed. By the definition of candidate explanation in Sec. 2.2, for example, it is not allowed to output the assignment that there exists $p \in O$ s.t. $H \cup B \not\models p$. To ensure that the search space includes only *valid* candidate explanations (i.e., H satisfies $H \cup B \models O$ and $H \cup B \not\models \perp$), we impose several constraints on the value assignments of h. We denote T to represent a set of logical atomic terms in P.

Constraint 1: From the definition of explanation, observations must be the logical consequences of $H \cup B$ (i.e., $H \cup B \models O$).

$$\text{for each } p \in O : h_p = 1 \tag{2}$$

Since we assume that P includes only positive literals, it is not necessary to ensure the consistency of $H \cup B$. In Figure 2, the constraint $h_{q(y)} = 1$ is generated.

Constraint 2: From the equality axiom in first-order logic, equality relations must be reflexive (i.e., for all $x \in T$, $h_{x=x} = 1$), symmetric (i.e., for all $x, y \in T$, $h_{x=y} = 1 \Rightarrow h_{y=x} = 1$), and transitive (i.e., for all $x, y, z \in T$, $h_{x=y} = 1 \wedge h_{y=z} = 1 \Rightarrow h_{x=z} = 1$). To satisfy these three properties, we introduce the following constraints:

$$\text{for each } x \in T : h_{x=x} = 1 \tag{3}$$

$$\text{for each } x, y \in T : h_{x=y} = h_{y=x} \tag{4}$$

$$\text{for each } x, y, z \in T : h_{x=y} + h_{y=z} - h_{x=z} \leq 1 \tag{5}$$

In Figure 2, the constraint $h_{x=y} + h_{A=x} - h_{y=A} \leq 1$ is generated as an instance of inequality (5).

Constraint 3: From the definition of h variables, h_p must be 1 if there exists set Q of literals such that Q implies p and each literal in Q is hypothesized (i.e., for all $Q \subseteq P$, $(Q \cup B \models p \wedge H \cup B \models Q) \Rightarrow H \cup B \models p$). To represent that each literal in the set Q of elemental explanations is hypothesized, we introduce a new ILP variable $a_Q \in \{0, 1\}$ s.t. $a_Q = 1$ iff all literals in Q are logical consequences of $H \cup B$; $a_Q = 0$ otherwise. Using a_Q, the constraint $\forall Q \subseteq P[(H \cup B \models Q \wedge Q \cup B \models p) \Rightarrow h_p = 1]$ can be expressed as follows:

$$\text{for each } p \in P : \sum_{Q \in \mathcal{E}(p)} a_Q \leq |\mathcal{E}(p)| \cdot h_p, \tag{6}$$

where $\mathcal{E}(p)$ is a family of sets of potential elemental explanations that explain p. For example, in Figure 2, the constraint $a_{\{s(y),t(u)\}} \leq h_{q(y)}$ is generated since $q(y)$ is explained by $s(y) \wedge t(u)$.

Constraint 4: From the definition of an ILP variable a, for all $Q \subseteq P$, a_Q can be set to 1 if and only if Q is a logical consequence of $H \cup B$ (for all $q \in Q$, $h_q = 1$). This can be expressed as follows:

$$\text{for each } p \in P, Q \in \mathcal{E}(p) : |Q| a_Q \leq \sum_{q \in Q} h_q \tag{7}$$

$$\text{for each } p \in P, Q \in \mathcal{E}(p) : \sum_{q \in Q} h_q \leq |Q| - 1 + a_Q \tag{8}$$

In Figure 2, two constraints are generated: (i) $2a_{\{s(y),t(u)\}} \leq h_{q(y)} + h_{t(u)}$, which ensures that $a_{\{s(y)t(u)\}} = 1 \Rightarrow h_{q(y)} = 1 \wedge h_{t(u)} = 1$; and (ii) $h_{q(y)} + h_{t(u)} \leq 1 + a_{\{s(y),t(u)\}}$, which ensures that $h_{q(y)} = 1 \wedge h_{t(u)} = 1 \Rightarrow a_{\{s(y)t(u)\}} = 1$.

In this formulation, we generate $O(n^3)$ ILP constraints for Constraint 2, where n is the number of logical atomic terms appearing in P. As the reader will see in Sec. 4, this makes inference intractable in large-scale processing. We show how this drawback can be overcome by exploiting CPI in Sec. 3.3.

Formulation for implementing the cost function: As mentioned in Sec. 2.2, we adopt the cost function proposed by Hobbs et al. (1993). For convenience, we repeat the cost function:

$$\mathit{cost}(H) = \sum_{h \in A(H)} \mathit{cost}(h), \tag{9}$$

where $A(H)$ is a set of *assumed* literals in H. This means that the cost of H is calculated from the subset of hypothesized literals. To represent the set of literals counted in the cost function, we first introduce ILP variables $c \in \{0, 1\}$ as follows:

$$\text{for each } p \in P : c_p = \begin{cases} 1 & \text{if } p \text{ pays its cost;} \\ 0 & \text{otherwise.} \end{cases}$$

In Figure 2, $c_{s(x)}$ is set to 0 in H_2 since $s(x)$ does not pay the cost (i.e. $s(x)$ is explained by $r(x)$).

Using c variables, the objective function of the ILP problem is given by

$$\text{minimize } \mathit{cost}(H) = \sum_{p \in P} c_p \cdot \mathit{cost}(p) \tag{10}$$

Note that it is easy to incorporate other criteria into the cost function. For instance, one can consider the plausibility of coreference relations between two mentions in a text. Assuming that mentions are represented by variables (e.g., $\mathit{cat}(x)$ means that mention x, whose linguistic expression is *cat*, appears in a text), one can add $\sum_{x,y \in T} \mathit{cost}(x, y, O) \cdot h_{x=y}$, where the cost is calculated by the information mentioned in O. For example, one could design the cost function that returns a higher cost if two contradictory properties are mentioned in O (e.g., $cat(x)$ and $dog(y)$ occur in O).

Again, from the definition of c variables, not all value assignments to c are allowed. Accordingly, we introduce several constraints on c as follows.

Constraint 5: From the definition of the cost function in Sec. 2.3, c_p is set to 1 if and only if (i) p is *not* explained (i.e., assumed); and (ii) p is *not* unified with any other literal that has a smaller cost by factoring H. To represent the second case, we introduce a new ILP variable $u_{p_1,p_2} \in \{0,1\}$ for the pair (p_1,p_2) of unifiable literals s.t. $u_{p_1,p_2} = 1$ iff p_1 is unified with p_2 by factoring H; $u_{p_1,p_2} = 0$ otherwise. Using u, the condition can be expressed as follows:

$$\text{for each } p \in P: \quad h_p + \sum_{Q \in \mathcal{E}(p)} (1 - a_Q) + \sum_{p' \in U^-(p)} (1 - u_{p,p'}) \leq |\mathcal{E}(p)| + |U^-(p)| + c_p \tag{11}$$

where $U^-(p)$ is a set of literals that (i) are unifiable with p, and (ii) have a cost smaller than $cost(p)$. For example, in Figure 2, we introduce $h_{s(x)} + (1 - a_{\{r(x)\}}) + (1 - u_{s(x),s(y)}) \leq 2 + c_{s(x)}$ as an instance of inequality (11).

Finally, we impose a constraint on u_{p_1,p_2} so that the value of u_{p_1,p_2} is allowed to be 1 only if (i) there exist equalities that make p_1 and p_2 equivalent in H, and (ii) p_1 and p_2 are hypothesized.[3]

Constraint 6: By the definition of an ILP variable u, $u_{p_1(\mathbf{x}),p_2(\mathbf{y})}$ can be set to 1 if and only if (i) two literals $p_1(\mathbf{x}) \equiv p_1(x_1, x_2, ..., x_n)$ and $p_2(\mathbf{y}) \equiv p_2(y_1, y_2, ..., y_n)$ are unified (i.e. the substitution $\{x_i/y_i\}_{i=1}^n$ occurs, namely $h_{x_i=y_i} = 1$ for all $i \in \{1, 2, ..., n\}$), and (ii) both $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$ are hypothesized.

$$(n+2) \cdot u_{p_1(\mathbf{x}),p_2(\mathbf{y})} \leq \sum_{i=1}^{n} h_{x_i=y_i} + h_{p_1(\mathbf{x})} + h_{p_2(\mathbf{y})} \tag{12}$$

$$\sum_{i=1}^{n} h_{x_i=y_i} + h_{p_1(\mathbf{x})} + h_{p_2(\mathbf{y})} \leq (n+2) - 1 + u_{p_1(\mathbf{x}),p_2(\mathbf{y})} \tag{13}$$

In Figure 2, the constraint $(1+2) \cdot u_{r(x),r(A)} \leq h_{x=A} + h_{r(x)} + h_{r(A)}$ is generated. Finally,

[3] We can consider the unification of two literals $p(x_1, x_2, \ldots, x_n), p(y_1, y_2, \ldots, y_n)$ as "$p(x_1, x_2, \ldots, x_n)$ is explained by $p(y_1, y_2, \ldots, y_n) \wedge x_1 = y_1 \wedge x_2 = y_2 \wedge \ldots \wedge x_n = y_n$" where $cost(p(x_1, x_2, \ldots, x_n)) > cost(p(y_1, y_2, \ldots, y_n))$. In this formulation, we do not need to introduce ILP variable u and Constraint 6, which can simplify our formulation. However, we intentionally leave the formulation with u variables in order to keep the comprehensibility of our paper.

in order to avoid the case where we hypothesize a single literal that (i) does not explain anything, but (ii) is unified with the other literal, we impose the following constraint:

$$\text{for each } p \in P: \quad h_p \leq \sum_{Q \in C(p)} a_Q, \tag{14}$$

where $C(p)$ is a family of sets of literals with which p co-occurs to explain the other literal. In Figure 2, since $s(y)$ co-occurs with $t(u)$ to explain $q(y)$ (i.e. $C(s(y)) = \{\{s(y), t(u)\}\}$), we introduce $h_{s(y)} \leq a_{s(y),t(u)}$. If we do not have this constraint, we can hypothesize $s(y)$ without hypothesizing $t(u)$ to reduce the cost of $s(x)$. Since such a hypothesis cannot be generated through backward inference, we need to prohibit it.

As mentioned in Sec. 1, the most similar work to ours is Santos's ILP-based formulation of propositional logic-based CBA (1994), but our approach is different in two ways.

First, we are capable of evaluating the specificity of explanations, which is an important feature for abduction-based NLP, as discussed in Sec. 2.2. The Santos approach amounts to performing most-specific abduction, and he finds a truth assignment to all propositions. Let us describe how the appropriate level of specificity is controlled in our approach. Suppose $O = \{p(a), q(a)\}$ and $B = \{r(x) \rightarrow p(x)\}$. We then have two candidate explanations. The first explanation is $H_1 = \{p(a), q(a)\}$, which simply assumes observations, and the cost is $\mathit{cost}(p(a)) + \mathit{cost}(q(a))$ (i.e. $c_{p(a)} = 1$, $c_{q(a)} = 1$). Backward chaining on $p(a)$ yields the second explanation $H_2 = \{q(a), r(a)\}$, which is more specific than H_1. The cost of H_2 is $\mathit{cost}(q(a)) + \mathit{cost}(r(a))$ ($c_{p(a)} = 0$, $c_{q(a)} = 1$, $c_{r(a)} = 1$). Note that we do not count $p(a)$ because $p(a)$ is *not* assumed anymore. Therefore, for this problem, if $\mathit{cost}(r(a)) < \mathit{cost}(p(a))$, then a more specific explanation H_1 is selected as the best explanation; otherwise, the less specific explanation H_2 is selected. This is controlled by the ILP variables c and Constraints 5 and 6, which are not introduced in the Santos approach. To summarize, our approach can decide which specificity of explanation is appropriate for the current observation and knowledge base, on the basis of how well the explanation is supported by observations.

Second, our approach directly models first-order CBA, while Santos approach formulates propositional-logic abduction. We could employ his approach for first-order CBA since it is well known that FOL formulae can be represented by propositional logic formulae through the application of grounding procedure (i.e., generate logical formulae, replacing variables with all possible constants). However, abductive inference over propositional level will make inference intractable when existentially quantified variables are included in observations or background knowledge. For example, suppose that $B = \{q(x, y) \rightarrow p(x, y), r(x, y, z) \rightarrow q(x, y)\}, O = \{p(x, y)\}$ and all possible constants are $\mathcal{C} = \{C_1, C_2, ..., C_n\}$. To ground this observation, we need to generate a disjunctive clause for $p(x, y)$, replacing x and y with all possible combinations from $\mathcal{C}$, i.e. $p(C_1, C_1) \lor p(C_1, C_2) \lor ... \lor p(C_n, C_n)$. The extension of the expressivity of observation is not difficult, but the problem arises in the search-space generation process: we get $O(n^2)$ potential elemental explanations (i.e. $q(C_i, C_j)$ for all $i, j \in \{1, 2, ..., n\}$) to explain each disjunct with the axiom $q(x, y) \rightarrow p(x, y)$. In addition, backchaining on each $q(C_i, C_j)$ with $r(x, y, z) \rightarrow q(x, y)$ yields $O(n)$ potential elemental explanations (i.e. $r(C_i, C_j, C_k)$ for all $k \in \{1, 2, ..., n\}$). In contrast, the search-space generation in our approach yields $\{p(x, y), q(x, y), r(x, y, u)\}$. As the readers can see, our approach seems to be more robust to the size of domain. In discourse processing, this robustness is important because literals usually have more than two or three arguments to represent an event with its participants.

3.3 CPI for CBA

One major drawback of ILP formulation is that it needs to generate $O(n^3)$ transitivity constraints, where n is the number of logical atomic terms, because we perform inference over FOL-based representation. That makes inference intractable (see Sec. 4 for empirical evidence) because it generates an ILP optimization problem that has quite a large number of constraints.

How do we overcome this drawback? The idea is that "all the transitivity constraints may not be violated all at once; so we gradually optimize and add transitivity constraints *if violated* in an iterative manner." More formally, we propose applying CPI to the CBA problems that was originally developed for solving large linear programming (LP) problems in operations research (Dantzig, Fulkerson, and Johnson 1954). CPI has been successfully applied to a wide range of constrained optimization problems where constraints are very large (Riedel and Clarke 2006; Riedel 2008; Joachims, Finley, and Yu 2009; Berant, Aviv, and Goldberger 2008), from probabilistic deductive inference problems (Riedel 2008) to machine learning problems (Joachims et al. 2009). To the best of our knowledge, however, our work is the first successful application of CPI to abductive inference tasks. In principle, CPI solves optimization problem in an iterative manner as follows: it solves an optimization problem without constraints, and then adds violated constraints to the optimization problem. When the iteration terminates, it guarantees solutions to be optimal. The proposed algorithm, called *CPI4CBA*, is also an exact inference framework.

How do we apply the technique of CPI to cost-based abduction problems? Intuitively, we iterate the following two steps: (i) solving an abduction problem without enforcing transitivity on logical atomic terms, and (ii) generating transitivity constraints dynamically when transitiveness of unification is violated (e.g., $H \cup B \models x = y \land y = z$, and $H \cup B \not\models x = z$). The iteration

Algorithm 3 cpiForLiftedFirstOrderCBA(Background Knowledge **B**, Observation **O**)

```
1: $(\Psi, I) \leftarrow createBaseILP(B, O)$
2: repeat
3:   $sol \leftarrow solveILP(\Psi, I)$; $V \leftarrow \phi$
4:   for $x, y \in \{t_1, t_2 \mid t_1 \in T, t_2 \in T, sol(h_{t_1=t_2}) = 1\}$ do
5:     for $z \in termsUnifiableWith(x, y)$ do
6:       // $H \cup B \models x = y \wedge y = z$, and $H \cup B \not\models x = z$
7:       if $sol(h_{y=z}) = 1$ and $sol(h_{x=z}) = 0$ then $V \leftarrow V \cup \{h_{x=y} + h_{y=z} - h_{x=z} \leq 1\}$
8:       // $H \cup B \models x = y \wedge x = z$, and $H \cup B \not\models y = z$
9:       if $sol(h_{y=z}) = 0$ and $sol(h_{x=z}) = 1$ then $V \leftarrow V \cup \{h_{x=y} + h_{x=z} - h_{y=z} \leq 1\}$
10:    end for
11:  end for
12:  $I \leftarrow I \cup V$
13: until $V = \phi$
```

terminates if there is no violated unification transitivity. The pseudocode is given in Algorithm 3. In line 1, we first create an ILP optimization problem described in Sec. 3.2, but without transitivity constraints (i.e., Constraint 2), where Ψ denotes a set of ILP variables, and I denotes a set of ILP constraints. In lines 2–13, we repeat checking consistency of unification transitiveness, adding constraints for violated transitiveness, and re-optimizing. In line 3, we find the solution *sol* for the current ILP optimization problem. Then, for each pair (x, y) of logical atomic terms unified in the solution *sol* (line 4), we find the logical term z which is unifiable with x and y (line 5). If the transitive relation x, y with respect to z is violated (i.e., $h_{x=z} = 0 \wedge h_{y=z} = 1$ or $h_{x=z} = 1 \wedge h_{y=z} = 0$), then we generate constraints for preventing this violation, and keep it in set V of constraints (lines 6–9). Finally, we again perform an ILP optimization with newly generated constraints (lines 12 and 3). The iteration ends when there is no violated transitiveness (line 13).

The key advantage of CPI4CBA is that it can reduce the time of search-space generation, and it is also expected to reduce the time of ILP optimization. CPI4CBA does not generate all transitivity constraints before optimization, which saves search-space generation time. In addition, optimization problems that we solve would become smaller than the original problem in most cases, because not all transitivity constraints need to be considered. In the worst case, we need to solve the optimization problem that is same as the original one; however, in most cases, we found this unnecessary. We will show its empirical evidence through large-scale evaluation in

Sec. 4.

4 Runtime Evaluation

How much does CPI improve the *runtime* of an ILP-based reasoner? Does CPI scale to larger real-life problems? To answer these questions, we evaluated the CPI4CBA algorithm in two settings: (i) **STORY**, the task of plan recognition; and (ii) **RTE**, the popular, knowledge-intensive, real-life natural language processing task of RTE. While most of the existing abductive inference systems are evaluated on rather small, and/or artificial datasets (Kate and Mooney 2009; Raghavan and Mooney 2010; Singla and Mooney 2011, etc.), our evaluation takes real-life, much larger datasets (see Sec. 4.1). In our experiments, we compare our system with systems (Kate and Mooney 2009; Singla and Mooney 2011; Blythe, Hobbs, Domingos, Kate, and Mooney 2011) based on Markov logic networks (MLNs) (Richardson and Domingos 2006). For our experiments, we have used a 12-Core Opteron 6174 (2.2GHz) 128 GB RAM machine, and assigned 8 CPU cores for each run. For an ILP solver, we used a Gurobi optimizer.[4] It is commercial, but an academic license is freely available. In our experiments, we use Constraint 6 *without* enumerating potential logical consequences (i.e. Algorithm 1 is used). The empirical evaluation with the enumeration of potential logical consequences is our future work.

4.1 Settings

STORY: For this setting, we have used Ng and Mooney's story understanding dataset (1992),[5] which is widely used for the evaluation of abductive plan recognition systems (Kate and Mooney 2009; Raghavan and Mooney 2010; Singla and Mooney 2011). In this task, we need to abductively infer the top-level plans of characters from actions. We follow Singla and Mooney's setting to define top-level plan predicates. These include 10 types of literals, such as *shopping*.[6] The dataset consists of 50 plan recognition problems represented by a set of ground atoms (e.g., $\{getting\_off(Getoff16), agent\_get\_off(Getoff16, Fred16), name(Fred16, Fred)\}$) and 107 background definite clauses (e.g., $go\_step(r, g) \wedge going(g) \rightarrow robbing(r)$). The dataset contains on average 12.6 literals in the logical forms of actions. We additionally generated 73 ILP constraints to prevent one variable/constant from having distinct top-level plan predicates in a hypothesis (e.g. $H = \{robbing(R), shopping(\mathrm{R})\}$). For example, we generated $h_{robbing(x)} + h_{shopping(y)} + h_{x=y} \leq 2$

[4]http://www.gurobi.com/

[5]ftp://ftp.cs.utexas.edu/pub/mooney/accel

[6]The complete list of top-level plan predicates is as follows: *shopping, robbing, traveling, rest_dining, drinking, paying, jogging*, and *partying*.

to represent that *robbing* and *shopping* cannot be hypothesized for the same variable/constant. Note that $H = \{robbing(R), shopping(S)\}$ is still possible.

To assign a cost to each literal (i.e., $cost(h)$ in equation (10)), we followed Hobbs et al.'s weighted abduction theory (Hobbs et al. 1993). In the theory, as mentioned in Sec. 2.2, each literal in the left-hand side of the axioms has a set of *weights*, which is expressed as $p_1^{w_1} \wedge p_2^{w_2} \wedge \ldots \wedge p_n^{w_n} \rightarrow q$. During backward chaining, each weight is multiplied with the cost of literal that is backchained on. For example, given $p(x)^{0.6} \wedge q(x)^{0.6} \rightarrow r(x)$ and $r(a)^{\$10}$, the theory derives $\{p(a)^{\$6}, q(a)^{\$6}\}$. Because the background knowledge of Ng and Mooney's dataset does not have weights, we assigned weights to the axioms so that the sum of the weights is 1.2 (e.g., $p^{0.4} \wedge q^{0.4} \wedge r^{0.4} \rightarrow s$). This assignment means that backward inference always increases the cost of explanation, and unification is the only way to reduce the cost. That is, it is almost equivalent to performing pure logic-based abduction, where the number of literals in an explanation is used as the plausibility of explanation.

RTE: For observations (input), we employed the second challenge of the RTE dataset.[7] In this, we need to determine correctly whether one text (called *text*, or T) entails another (called *hypothesis*, or H). The dataset consists of a development set and a test set, each of which includes 800 natural language text hypothesis pairs. We used all 800 texts from the test set. We converted texts into logical forms presented in Hobbs (1985) using the Boxer semantic parser (Bos 2008). The number of literals in observations is 29.6 literals on average. For background knowledge, we extracted 289,655 axioms[8] from WordNet 3.0 (Fellbaum 1998), and 7,558 axioms from FrameNet 1.5 (Ruppenhofer et al. 2010), following Ovchinnikova et al. (2011). In principle, the WordNet knowledge base contains various lexical relations between words, such as IS-A, ontological relations (e.g., $dog(x) \rightarrow animal(x)$). FrameNet knowledge bases contain lexeme-to-frame mappings, frame-frame relations, etc. For example, the mapping from surface realization "x_1 give x_2 x_3" to a frame "Giving" is given by $Giving(e_1, x_1, x_2, x_3) \wedge donor(e_1, x_1) \wedge recipient(e_1, x_2) \wedge theme(e_1, x_3) \rightarrow give(e_1, x_1, x_2, x_3)$. We again followed Hobbs's weighted abduction theory for calculating the cost of explanation. We assigned the weights to axioms by following Ovchinnikova et al. (2011) in this setting.

4.2 Results and discussion

The reasoner was given a 2-min time limit for each inference step (i.e., search-space generation and best-explanation search). In Table 1, we show the results of each setting for two inference

[7]http://pascallin.ecs.soton.ac.uk/Challenges/RTE2/

[8]The extracted relations include word-to-synset mapping, hypernym-hyponym, cause-effect, entailment, derivational, instance-of relations.

Setting	Method	Depth	Generation [sec.] (timeout = 120)	ILP inf [sec.] (timeout = 120)	# of ILP cnstr
STORY	IAICBA	1	0.02 (100.0 %)	0.60 (100.0 %)	3,708
		2	0.12 (100.0 %)	5.34 (100.0 %)	23,543
		3	0.33 (100.0 %)	8.11 (100.0 %)	50,667
		∞	0.35 (100.0 %)	9.00 (100.0 %)	61,122
	CPI4CBA	1	0.01 (100.0 %)	0.34 (100.0 %)	784 (Δ 451)
		2	0.07 (100.0 %)	4.15 (100.0 %)	7,393 (Δ 922)
		3	0.16 (100.0 %)	3.36 (100.0 %)	16,959 (Δ 495)
		∞	**0.22 (100.0 %)**	**5.95 (100.0 %)**	24,759 (Δ 522)
RTE	IAICBA	1	0.01 (100.0 %)	0.25 (99.7 %)	1,104
		2	0.08 (100.0 %)	2.15 (98.1 %)	5,185
		3	0.56 (99.9 %)	5.66 (93.0 %)	16,992
		∞	4.78 (90.7 %)	15.40 (60.7 %)	36,773
	CPI4CBA	1	0.01 (100.0 %)	0.05 (100.0 %)	269 (Δ 62)
		2	0.04 (100.0 %)	0.35 (99.6 %)	1,228 (Δ 151)
		3	0.09 (100.0 %)	1.66 (99.0 %)	2,705 (Δ 216)
		∞	**0.84 (98.4 %)**	**11.73 (76.9 %)**	10,060 (Δ 137)

Table 1 The results of averaged inference time in **STORY** and **RTE**.

methods: (i) *IAICBA*: the inference method without CPI, and (ii) *CPI4CBA*: inference method with CPI.

In order to investigate the relationship between the size of search space and the runtime, we show the results for each depth, which we used for limiting the length of backward chaining. In the "Generation" column, we show the runtime, which is taken for search-space generation in seconds, averaged over all problems whose search-space generation is finished within 2 min. In the parenthesis, we show the percentage of those problems whose search-space generation is finished within 2 min. In the column "ILP inf," we show the runtime of ILP optimization averaged on only problems such that both search-space generation and ILP optimization are finished within 2 min as well as the percentage of those problems (e.g., 80% means "for 80% of all the problems, search-space generation, as well as ILP inference, was finished within 2 min."). In the "# of ILP cnstr" column, we show the averaged number of generated ILP constraints. Concerning CPI4CBA, the number denotes the averaged number of constraints considered in the end, including the constraints added by CPI. The number marked by Δ indicates the averaged number of constraints that are added during CPI (i.e., how many times are the constraints added by line 7 or 9 in Algorithm 3).

Overall, the runtimes in both search-space generation and ILP inference are dramatically improved from IAICBA to CPI4CBA in both settings, as shown in Table 1. In addition, CPI4CBA can find optimal solutions in ILP inference for more than 90% of the problems, even for depth ∞. This indicates that CPI4CBA scales to larger problems. The results of IAICBA in **RTE** settings indicate the significant bottleneck of IAICBA in large-scale reasoning: the time of search-space generation. Search-space generation could be performed within 2 min for only 90.7% of the problems. CPI4CBA successfully overcomes this bottleneck. CPI4CBA is clearly advantageous in search-space generation because it is not necessary to generate transitivity constraints, an operation that grows cubically before optimization.

In addition, CPI4CBA also reduces the time of ILP inference significantly. In ILP inference, CPI did not guarantee the reduction of inference time in theory; *however*, as shown in Table 1, we found that the number of ILP constraints actually used is much less than the original problem. Therefore, CPI4CBA successfully reduces the complexity of the ILP optimization problems in practice. This is also supported by the fact that CPI4CBA keeps 76.9% in "ILP inf" for Depth = ∞ because it solves very large ILP optimization problems that fail to be generated in IAICBA. In order to see how CPI contributes to the improvement in ILP inference time, we show how the runtime of IAICBA is affected by the CPI4CBA method for each problem in Figure 3. Each data point corresponds to one problem in **STORY** and **RTE** settings. We show the data points for problems for which we found optimal solutions in ILP inference for Depth = ∞. Overall, the runtime of CPI4CBA is smaller than that of IAICBA in most problems. In particular, CPI4CBA successfully reduces the time of ILP inference for larger problems by exploiting the iterative optimization technique. In the larger domain of **RTE** setting, we found that performance was improved in 81.7% of the problems.

Finally, we compare CPI4CBA with the existing MLN-based systems (Kate and Mooney 2009; Singla and Mooney 2011; Blythe et al. 2011). In summary, our system is comparable or slightly less efficient in the **STORY** setting; *however*, our system is more efficient in the **RTE** setting.

For the **STORY** setting, Singla and Mooney (2011) reported the averaged inference time of two existing MLN-based systems using CPI (Riedel 2008) *on the test set*: (i) Kate and Mooney (2009)'s approach: 2.93 s, and (ii) Singla and Mooney (2011)'s approach: 0.93 s.[9] To make the

[9]This is the result of MLN-HC in (Singla and Mooney 2011). MLN-HCAM cannot be directly compared with our results, since the search space is different from our experiments because they unify some assumptions in

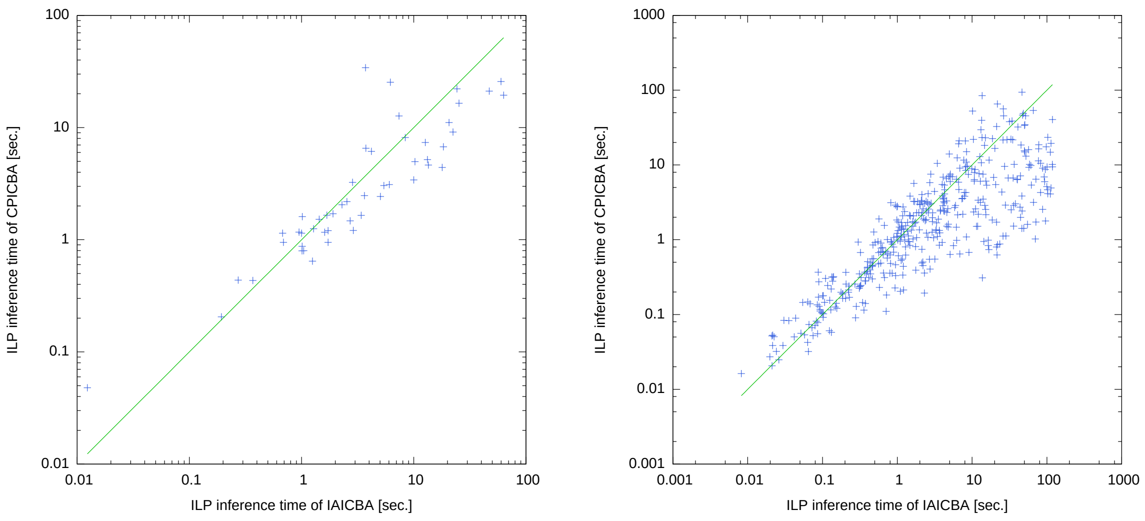


Fig. 3 Runtime comparison between IAICBA and CPI4CBA (logarithmic scale). The left figure shows the results of STORY dataset, and the right figure shows the results of RTE datasets.

comparison fair, we evaluated our approach with a single CPU core *on the test set*. It took 2.36 s on average to process all the problems (optimal solutions were found for all of them). MLN-based approaches seem to be reasonably efficient for small datasets.

However, it does not scale to larger problems; for the **RTE** setting, Blythe et al. (2011) reported that only 28 from 100 selected RTE-2 problems could be run to completion with only the FrameNet knowledge bases. The processing time was 7.5 min on average (personal communication[10]). However, our method solves 76.9% of all the problems, with suboptimal solutions still available for the remaining 21.5%, and it takes only 0.84 s for search-space generation, and 11.73 s for ILP inference. As mentioned in Sec. 5, our framework is more scalable because it does not need to generate the axioms explicitly to emulate an *explaining away* effect (i.e., inferring one cause makes another cause less probable) and needs no grounding (Sec. 3.2).

advance to reduce the search space.

[10]They used 56,000 FrameNet axioms in the experiments, while we used 289,655 WordNet axioms and 7,558 FrameNet axioms. To make the comparison fairer, we run our experiment on the same but smaller dataset, as in Blythe et al. (2011). The dataset contains 65 background axioms and 14 abductive interpretation problems from Hobbs et al. (1993). As reported in Blythe et al. (2011), it took 5.6 s for their system to process 12 problems (2 problems cannot be solved), while it took 1.0 s for our system to process all the problems with a single CPU core (Depth = 8).

5 Related work

The computational aspect of abduction has been studied extensively in the contexts of logic programming and statistical relational learning. In the context of logic programming, abduction has been introduced as the extension of logic programming (Stickel 1991; Kakas, Kowalski, and Toni 1992), where the extended framework is often called abductive logic programming (ALP). Since abduction and induction share the basic framework (see Sec. 2.2 for detail), abduction has also been studied in the area of inductive logic programming, the logic programming framework for induction (Inoue 2004; Tamaddoni-Nezhad, Chaleil, Kakas, and Muggleton 2006). In the context of ALP, Stickel (1991) showed how to formulate minimum-cost explanation finding in Prolog, a popular implementation of logic programming. Stickel allowed the system to *assume* literals during SLD resolution when definite clause rules or facts unifiable with the targeted literal are not found. In this system, the cost of explanation is calculated by the sum of the costs of elemental explanations, and the costs of axioms used for constructing the proof. However, Stickel did not show how to implement it efficiently.

In the following years, a number of methods attempting to find the minimum-cost explanation efficiently were proposed (Santos 1994; Ishizuka and Matsuo 1998; Prendinger and Ishizuka 1999; Abdelbar and Hefny 2005; Chivers et al. 2007; Guinn, Shipman, and Addison 2008); for example, Santos (1994) formulated cost-based abduction in propositional logic using ILP, and showed its efficiency. However, most of them focus on improving the efficiency of propositional logic-based abduction. As discussed in Sec. 3, one could use such a framework through propositionalization techniques for first-order CBA; however, the propositionalization will produce a huge number of ground instances of background knowledge axioms and literals in observation. Hence, they would not scale to larger problems with sizeable knowledge bases.

In the context of statistical relational learning, abduction has also been widely studied. One of the prominent formalisms is PRISM (Sato and Kameya 2008), which is a general logic-based probabilistic modeling language. In the past two decades, a number of techniques for efficient inference or learning have been studied extensively (see Sato and Kameya (2008) for overview). Concerning inference, in principle PRISM achieves the best explanation finding in a polynomial time through a tabled search technique for logic programs (Tamaki and Sato 1986). However, this technique exploits the local information computed so far, and hence is incompatible with the factoring of explanation, which is a global operation (personal communication). It is a nontrivial issue to incorporate the factoring process into the search without loss of efficiency.

Another important stream is the series of studies (Kate and Mooney 2009; Blythe et al. 2011;

Singla and Mooney 2011), where abduction has been emulated through MLNs (Richardson and Domingos 2006), a probabilistic deductive inference framework. MLNs provide full support of first-order predicate logic and software packages of inference and learning; however, MLN-based approaches have severe overheads of inference: (i) they require special procedures to convert abduction problems into deduction problems because of the deductive nature of MLNs, and (ii) they need grounding for inference. To emulate abduction in the deductive framework, the pioneering work of MLN-based abduction (Kate and Mooney 2009) exploits the reverse implication of the original axioms, and uses the additional axioms to emulate the *explaining away* effect (i.e., inferring one cause makes another cause less probable). For example, suppose $B = \{p_1 \rightarrow q, p_2 \rightarrow q, p_3 \rightarrow q\}$. Then, B is not used in MLN background knowledge base as it is: B is converted into the following set of logical formulae: $\{q \rightarrow p_1 \vee p_2 \vee p_3,\ q \rightarrow \neg p_1 \vee \neg p_2,\ q \rightarrow \neg p_1 \vee \neg p_3\}$. As the readers can imagine, an MLN-based approach suffers from the inefficiency of inference because of the increase in converted axioms. In addition, to the best of our knowledge, most of the existing approaches for maximum-a-posterior (MAP) inference for MLN (Singla and Domingos 2006; Riedel 2008, etc.) need (partial) grounding of axioms, which makes inference prohibitively slow.

In terms of the applications, there are multiple researches that exploit abduction in many fields. For example, in systems biology, abduction is used for discovering scientific knowledge, such as causal relationships from genotype to phenotype, or modeling inhibition in metabolic networks (Doncescu, Inoue, and Sato 2008; Tamaddoni-Nezhad et al. 2006).

6 Conclusion

We have proposed an ILP-based lifted formulation for cost-based abduction on FOL. Although abductive reasoning on FOL is computationally expensive, we demonstrated that the lifted inference and CPI techniques bring us a significant boost to the efficiency of FOL-based reasoning. We have evaluated our method on two datasets, including real-life problems (i.e., RTE dataset with axioms generated from WordNet and FrameNet). Our evaluation revealed that our inference method CPI4CBA was more efficient than other existing systems on a large KB. The abductive inference engine presented in this paper is made publicly available.[11]

In future work, we plan to apply CPI to both search-space generation and ILP inference, repeating the generation of potential elemental explanations and ILP optimization interactively, as in Cutting Plane MAP inference in MLNs (Riedel 2008).

[11] https://github.com/naoya-i/henry-n700

We also plan to extend the expressivity of our approach, i.e., to support negations in background knowledge. The capability of handling negations is crucial for a wide range of abductive inference systems. For example, in abduction-based natural language interpretation, one can easily imagine that it needs to handle negated expressions, such as "*I don't like ice cream*", or "*Tweety is not a bird.*" Our future direction also includes providing the formal proof of completeness and soundness of our approach.

Acknowledgment

I thank the reviewers of this paper for their helpful comments. This work was partially supported by Grant-in-Aid for JSPS Fellows (22-9719) and Grant-in-Aid for Scientific Research (23240018). The authors would like to thank Enago (www.enago.jp) for the English language review.

Reference

Abdelbar, A. M. and Hefny, M. (2005). "An efficient LP-based admissible heuristic for cost-based abduction." *Journal of Experimental and Theoretical Artificial Intelligence*, **17** (3), pp. 297–303.

Berant, J., Aviv, T., and Goldberger, J. (2008). "Global Learning of Typed Entailment Rules." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 610–619.

Blythe, J., Hobbs, J. R., Domingos, P., Kate, R. J., and Mooney, R. J. (2011). "Implementing Weighted Abduction in Markov Logic." In *Proceedings of the International Conference on Computational Semantics*, pp. 55–64.

Bos, J. (2008). "Wide-Coverage Semantic Analysis with Boxer." In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pp. 277–286.

Chambers, N. and Jurafsky, D. (2009). "Unsupervised Learning of Narrative Schemas and their Participants." In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 602–610.

Charniak, E. and Goldman, R. P. (1991). "A Probabilistic Model of Plan Recognition." In *Proceedings of the 9th National Conference on Artificial Intelligence*, pp. 160–165.

Chivers, S. T., Tagliarini, G. A., and Abdelbar, A. M. (2007). "An Evolutionary Optimization

Approach to Cost-Based Abduction, with Comparison to PSO." In *Proceedings 2007 IEEE International Joint Conference in Neural Networks*, pp. 2926–2930.

Dagan, I., Dolan, B., Magnini, B., and Roth, D. (2010). "Recognizing textual entailment: Rational, evaluation and approaches - Erratum." *Natural Language Engineering*, **16** (1), p. 105.

Dantzig, G. B., Fulkerson, R., and Johnson, S. M. (1954). "Solution of a large-scale traveling salesman problem." *Operations Research*, **2** (4), pp. 393–410.

Doncescu, A., Inoue, K., and Sato, T. (2008). "Hypothesis-Finding in Systems Biology." *ALP Newsletter*, **21**, pp. 2–3.

Etzioni, O., Banko, M., and Cafarella, M. J. (2006). "Machine reading." In *Proceedings of the 21st National Conference on Artificial Intelligence.*

Fellbaum, C. (Ed.) (1998). *WordNet: an electronic lexical database.* MIT Press.

Guinn, C., Shipman, W., and Addison, E. (2008). "The Parallelization of Membrane Computers to Find Near Optimal Solutions to Cost-Based Abduction." In *Proceedings of the 2008 International Conference on Genetic and Evolutionary Methods*, pp. 241–2–47.

Hobbs, J. R. (1985). "Ontological Promiscuity." In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pp. 61–69 Chicago, Illinois.

Hobbs, J. R., Stickel, M., Martin, P., and Edwards, D. (1993). "Interpretation as Abduction." *Artificial Intelligence*, **63**, pp. 69–142.

Hovy, D., Zhang, C., Hovy, E., and Penas, A. (2011). "Unsupervised Discovery of Domain-Specific Knowledge from Text." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1466–1475.

Inoue, K. (2004). "Induction as consequence finding." *Machine Learning*, **55** (2), pp. 109–135.

Ishizuka, M. and Matsuo, Y. (1998). "SL Method for Computing a Near-optimal Solution using Linear and Non-linear Programming in Cost-based Hypothetical Reasoning." In *Proceedings of the 5th Pacific Rim International Conference on Artificial Intelligence: Topics in Artificial Intelligence*, pp. 611–625.

Joachims, T., Finley, T., and Yu, C. J. (2009). "Cutting-plane training of structural SVMs." In *Machine Learning*, pp. 27–59.

Kakas, A., Kowalski, R., and Toni, F. (1992). "Abductive logic programming." *Journal of Logic and Computation*, **2** (6), pp. 719–770.

Kate, R. J. and Mooney, R. J. (2009). "Probabilistic Abduction using Markov Logic Networks." In *Proceedings of the IJCAI-09 Workshop on Plan, Activity, and Intent Recognition.*

Kowalski, R. (1974). "Predicate Logic as a Programming Language." In *IFIP Congress*, pp. 569–574.

Mulkar, R., Hobbs, J., and Hovy, E. (2007). "Learning from Reading Syntactically Complex Biology Texts." In *Proceedings of the 8th International Symposium on Logical Formalizations of Commonsense Reasoning.* Palo Alto.

Ng, H. T. and Mooney, R. J. (1992). "Abductive Plan Recognition and Diagnosis: A Comprehensive Empirical Evaluation." In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pp. 499–508.

Ovchinnikova, E., Montazeri, N., Alexandrov, T., Hobbs, J. R., McCord, M., and Mulkar-Mehta, R. (2011). "Abductive Reasoning with a Large Knowledge Base for Discourse Processing." In *Proceedings of the International Conference on Computational Semantics*, pp. 225–234.

Penas, A. and Hovy, E. (2010). "Filling knowledge gaps in text for machine reading." In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pp. 979–987.

Peraldi, S. E., Kaya, A., Melzer, S., Möller, R., and Wessel, M. (2007). "Multimedia Interpretation as Abduction." In *International Workshop on Description Logics.*

Poole, D. (1993a). "Logic Programming, Abduction and Probability: a top-down anytime algorithm for estimating prior and posterior probabilities." *New Generation Computing*, **11(3-4)**, pp. 377–400.

Poole, D. (1993b). "Probabilistic Horn abduction and Bayesian networks." *Artificial Intelligence*, **64 (1)**, pp. 81–129.

Prendinger, H. and Ishizuka, M. (1999). "First-Order Diagnosis by Propositional Reasoning: A Representation-Based Approach." In *Proceedings of the 10th International Workshop on Principles of Diagnosis*, pp. 220–225.

Raghavan, S. and Mooney, R. J. (2010). "Bayesian Abductive Logic Programs." In *Proceedings of the AAAI-10 Workshop on Statistical Relational AI*, pp. 82–87.

Richardson, M. and Domingos, P. (2006). "Markov logic networks." *Machine Learning*, pp. 107–136.

Riedel, S. (2008). "Improving the Accuracy and Efficiency of MAP Inference for Markov Logic." In *Proceedings of the 24th Annual Conference on Uncertainty in AI*, pp. 468–475.

Riedel, S. and Clarke, J. (2006). "Incremental Integer Linear Programming for Non-projective Dependency Parsing." In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 129–137.

Robinson, J. A. (1965). "A Machine-Oriented Logic Based on the Resolution Principle." *Journal of the ACM*, **12**, pp. 23–41.

Ruppenhofer, J., Ellsworth, M., Petruck, M., Johnson, C., and Scheffczyk, J. (2010). "FrameNet

II: Extended Theory and Practice." Tech. rep., Berkeley, USA.

Santos, E. (1994). "A linear constraint satisfaction approach to cost-based abduction." *Artificial Intelligence*, **65 (1)**, pp. 1–27.

Sato, T. and Kameya, Y. (2008). "New Advances in Logic-Based Probabilistic Modeling by PRISM." In Raedt, L., Frasconi, P., Kersting, K., and Muggleton, S. (Eds.), *Probabilistic Inductive Logic Programming*, Vol. 4911 of *Lecture Notes in Computer Science*, pp. 118–155. Springer Berlin Heidelberg.

Schoenmackers, S., Davis, J., Etzioni, O., and Weld, D. (2010). "Learning First-order Horn Clauses from Web Text." In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1088–1098.

Shanahan, M. (2005). "Perception as Abduction: Turning Sensor Data Into Meaningful Representation." *Cognitive Science*, **29** (1), pp. 103–134.

Singla, P. and Domingos, P. (2006). "Memory-Efficient Inference for Relational Domains." In *Proceedings of the 21st National Conference on Artificial Intelligence*, pp. 488–493.

Singla, P. and Mooney, R. J. (2011). "Abductive Markov Logic for Plan Recognition." In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 1069–1075.

Stickel, M. E. (1991). "A prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation." *Annals of Mathematics and Artificial Intelligence*, **4** (1), pp. 89–105.

Tamaddoni-Nezhad, A., Chaleil, R., Kakas, A., and Muggleton, S. (2006). "Application of abductive ILP to learning metabolic network inhibition from temporal data." *Machine Learning*, **64** (1-3), pp. 209–230.

Tamaki, H. and Sato, T. (1986). "OLD resolution with tabulation." In Shapiro, E. (Ed.), *Third International Conference on Logic Programming*, Vol. 225 of *Lecture Notes in Computer Science*, pp. 84–98. Springer Berlin Heidelberg.

Naoya Inoue: He received his M.S. degree of engineering from Nara Institute of Science and Technology in 2010 and his Ph.D. degree in information science from Tohoku University in 2013. He is currently a post-doc researcher at Tohoku University and works as a researcher for DENSO Research Laboratries. His research interests are in inference-based discourse processing and language grounding problems.

Kentaro Inui: He received his doctorate degree of engineering from Tokyo Institute of Technology in 1995. Having experienced Assistant Professor at Tokyo Institute of Technology and Associate Professor at Kyushu Institute of Technology and Nara Institute of Science and Technology, he has been Professor of Graduate School of Information Sciences at Tohoku University since 2010. His research interests include natural language understanding and knowledge processing. He currently serves as IPSJ Director and ANLP Director.