A Study on Scalable and Trainable Abductive Reasoning for Natural Language Discourse Processing

Kazeto Yamamoto

Graduate School of Information Sciences

Tohoku University

A thesis submitted for the degree of

Doctor of Information Science

January 2016

Acknowledgements

主指導教官の乾健太郎教授には，本研究を遂行し学位論文をまとめるに当たり，多くの御支援と御指導をいただきました．また，学部3年次での研究室配属からの6年間，暖かく，かつ辛抱強く見守っていただき，研究生活全般にわたって多大なご支援・御助言をいただきました．深く感謝いたします．

審査委員として本論文を査読してくださいました田中和之教授，ならびに住井英二郎教授に深く感謝いたします．

岡崎直観准教授には，研究者として，ソフトウェア開発者として，多くのことを学ばせていただきました．深く感謝いたします．

現人工知能研究センター長の辻井潤一先生，ならびに現大阪大学の荒瀬由紀准教授には，Microsoft Research Asia でのインターンシップへの参加にあたって研究チームへ受け入れていただき，滞在期間中は終始暖かく御指導いただきました．初めての海外での長期滞在や，多くの優秀な研究者の中での研究活動など，5ヶ月という期間のなかで，いくつもの貴重な経験をさせていただきました．深く感謝いたします．

日本電気株式会社の渡邉陽太郎さんには，東北大学の助教として在籍されていた間，たびたび御指導をいただきました．これからまたお世話になります．どうぞ宜しくお願いいたします．

研究室の先輩でもある井之上直也助教には，研究テーマが近いこともあり，博士前期課程からの5年間は特に，多くの御助言・御指導をいただきました．博士前期課程に進学して研究テーマに悩んでいた私が今の研究テーマに携わるきっかけを下さったことや，研究に行き詰まるたびに相談させていただいたことなど，この5年間でどれだけ助けられたか分かりません．深く感謝いたします．

成田順子さん，菅原真由美さん，八巻智子さんには，研究活動および大学生活を暖かく支えていただきました．深く感謝いたします．

杉浦純さんは博士後期課程まで進学した同期の友人で，学部3年次での研究室配属から6年間，大学生活における様々な面で支えていただきました．深く感謝いたします．

また，研究を進めるにあたり，ご支援，ご協力をいただきながら，ここにお名前を記すことが出来なかった多くの方々に深く感謝します．

最後に，私が自分の思う道を進むことに対しても変わらず応援し，暖かく見守り，支えてくれた両親に心から感謝します．そして，博士後期課程の3年において，いつも暖かく応援し，心の支えとなってくれた佐野江里さんに心から感謝します．今後とも宜しくお願いします．

Abstract

Abduction is inference from a given set of observations to the best explanation about why those observed event happened. This mode of inference has long been considered a promising framework for *Discourse processing*, a subtask of natural language processing, which is the task of making implicit information in natural language texts explicit.

Formulating discourse understanding as abductive reasoning is expected to bring several distinct advantages: (1) it provides not only output of task but also human interpretable proof trees and hence shows us what was hypothesized and what knowledge was used to explain the observation, (2) it provides a uniform framework for integrating subtasks of multiple levels of abstraction and (3) the declarative nature of abduction allows us to abstract away from the procedural process of inferences.

In spite of these promising properties, however, in fact, the abduction-based approaches to subtasks of discourse processing, such as text/story understanding and plan/intention recognition, have never produced significant positive evidence that supports their effectiveness in real-life problems.

In this thesis, we address some of the problems which have hindered the success of these abduction-based approaches. Specifically, we address following issues: (i) abductive reasoning procedures were not efficient enough to use huge knowledge base, and (ii) the evaluation function, which is used to evaluate the goodness of explanation, is hand-tuned for each task.

As a solution to the first issue, we propose an efficient inference method of first-order abduction, which eliminates redundant explanations from the search space efficiently. Through the large-scale evaluation, we demonstrate that proposed method is far more efficient than the other existing abductive reasoners.

As a solution to the second issue, we propose a method to discriminatively learn parameters of the evaluation function. This method is applicable to an evaluation function if only the evaluation function is differentiable with respect to the parameters to tune. In our evaluation, we show that our learning procedure can reduce the value of loss function in each iteration, and learned parameters are also robust to unseen dataset.

Contents

List of Figures

Chapter 1

Introduction

Natural language texts contain various implicit information, which the writer omitted. For instance, given a sentence "John went to the bank. He got a loan", we humans can easily find various implicit information — "he" refers to "John", the purpose of John's going to the bank is to get a loan, and so on. Discourse processing, a subtask of natural language processing (NLP), is the task to make the implicit information like these in natural language texts explicit.

Abduction is inference from a given set of observations to the best explanation about why those observed events happened. This mode of inference has long been aplied to a range of AI tasks including text/story understanding and plan/intention recognition.

An epoch-making study in this line of research can be seen in a paper in *Artificial Intelligence* by Hobbs et al; they demonstrate that a wide range of subtasks in the understanding of natural language can be uniformly formulated as abductive reasoning (*Interpretation as Abduction*). For instance, given above sentence, "John went to the bank. He got a loan" as input, it is assumed that the model of Hobbs et al. semantically parses it to obtain a logical form, which consists of a flat conjunctive set of observed literals, as shown at the bottom of Figure 1.1.

This way of formulating intelligent inference has several distinct advantages:

1. It provides not only output of task but also human interpretable proof trees like Figure 1.1. Those explicitly show us what was hypothesized and what

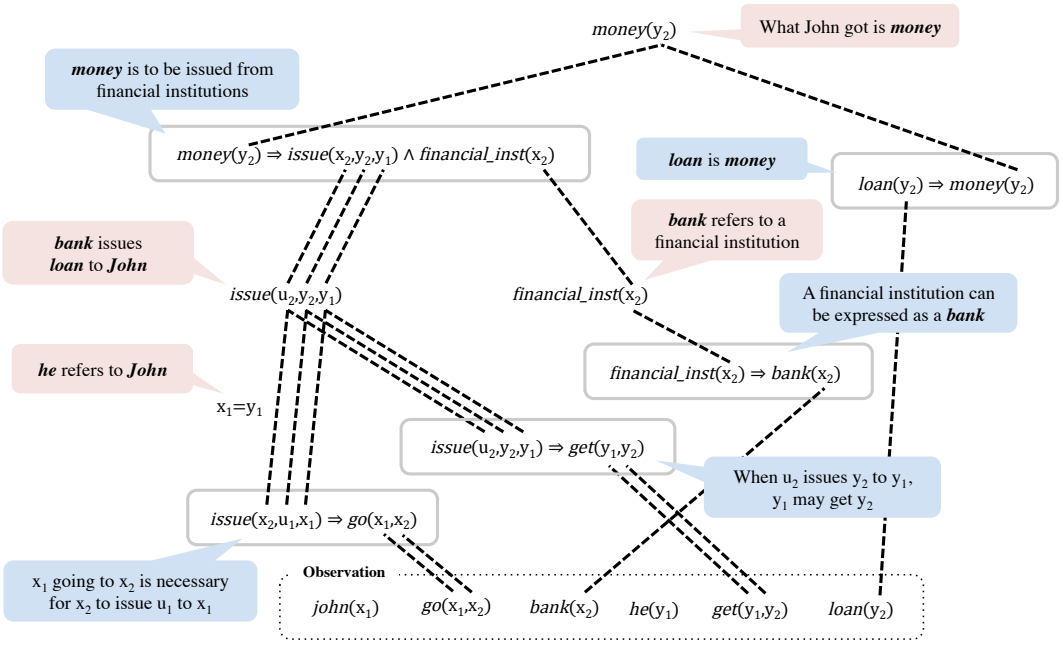

Figure 1.1: An example of discourse understanding with abductive reasoning.

knowledge was used to explain the observation.

2. It provides a uniform framework for integrating subtasks of multiple levels of abstraction; in the above example, finding the best explanation jointly resolves the coreference relation, the discourse relation, and the word-sense ambiguity.

3. The declarative nature of abduction allows us to abstract away from the procedural process of inferences. When multiple levels of interdependent subtasks are involved, it is often crucially difficult to predetermine the optimal order in which to solve the problems. This difficulty can be avoided by using joint inference.

In spite of these promising properties, however, the abduction-based approaches to text/story understanding and plan/intention recognition have never produced significant positive evidence that supports their effectiveness in real-life problems.

In this thesis, we aim to solve the problems which have hindered the success of these abduction-based approaches.

1.1 Research Issues

In this thesis, we try to construct discourse processing frameworks using abduction with large knowledge base. Specifically, we address following two issues:

Scalability Abduction on first-order logic (FOL) or similarly expressive languages is NP-hard and computationally expensive. Its search space grows exponentially with the size of the knowledge base.

Trainability Less attention has been paid to how to automatically learn score functions, which rank candidate explanations in order of their plausibility (henceforth, we call it the *evaluation function*). To apply abductive inference with large knowledge base to real-world problem, this non-trivial issue needs to be addressed because the criterion of plausibility is highly task-dependent.

1.2 Contributions

This thesis makes following contributions.

Scalable abduction framework for discourse processing We propose an efficient inference method of abductive reasoning on first-order logic. This method is based on ILP formulated Abduction.

Discriminative learning method of abduction We propose a method to discriminatively tune the parameters of the evaluation function in first-order abduction. This method is not task-specific nor model-specific and is therefore widely applicable. If only an evaluation function is differentiable with respect to its parameters to tune, it is tunable by this method.

The all-in-one software package for abduction We have implemented the proposed methods in one software package, which is called *Phillip*. The

software is an opensource software and publicly available at the Github[1]. This accomplishes much more efficient abduction than existing implementations and the supervised learning on various existing evaluation function. Phillip's implementation is flexible and then enables one to easily develop a new abductive inference model.

1.3 Thesis Overview

The rest of this thesis is structured as follows.

- **Chapter 2: Inference-based Approach for Discourse Processing** In this chapter, we introduce the theoretical background — first order logic, abductive inference and so on.
- **Chapter 3: Boosting Efficiency of Abduction** The problem of finding the best abductive explanation is an NP-hard problem. In this chapter, we propose an efficient inference method for first-ordered abduction.
- **Chapter 4: Boosting Efficiency of Abduction for Discourse Processing** In this chapter, we propose an efficient inference method for abductive inference-based frameworks for discourse processing.
- **Chapter 5: Discriminative Learning of Abduction** In this chapter, we propose an discriminative learning method for an evaluation function in first-ordered abduction.
- **Chapter 6: Scalable and Trainable Open Source Abductive Reasoner** We have implemented the proposed methods as an open sourse software, *Phillip*. In this chapter, we outline it and provide a basic way to use it.
- **Chapter 7: Conclusions** We summarize our discussion, and present our future direction.

[1] http://github.com/kazeto/phillip/

Chapter 2

Inference-based Approach for Discourse Processing

In this chapter, we introduce first-order abduction and some related works.

2.1 First-Order Logic

First-Order Logic (FOL) is a variety of predicate logic, which allows quantification of only variables and used as a language of meaning representation.

In FOL, the basic unit of meaning is called an *atom* or *atomic formula*. An atom is a form of $p(x_1, x_2, ..., x_N)$, where p is called *predicate* and $x_1, x_2, ..., x_n$ are called *terms*. A predicates is a symbol that represents property of objects, relation between objects and so on. A term represents some object in the world. For example, the atom $apple(x)$ means that an object x has property of *apple*, tha atom $love(John, Mary)$ means that there exists the relation of *love* between *John* and *Mary*. The number of terms in each atom is inherent in its predicate. The number of argument which a predicate takes is called *arity* and a predicate which takes N terms as arguments is called *N-ary predicate*. In this thesis, we denote a N-ary predicate p as p/n. For example, the predicate *love* of above example takes two argument and then is called 2-ary predicate and denoted as $love/2$. An atom whose all terms are constants is called a *grounded atom*.

Each atom can be negated. When an atom is negated, the truth value of an

atom is false. In FOL, the negation of an atom a is represented as $\neg a$. Negation operator can be recursively applied. The negation of a negated atom is equal to non-negated atom, $\neg(\neg a) = a$. A non-negated atom or a negated atom is called *literal*. An literal whose all terms are constants is called a *grounded literal*.

In order to represent a multiple fact consisting of prural literals, a *logical connection* can be used. The following are some typical logical connections; A *conjunction* $L_1 \wedge L_2$ is true iff both L_1 and L_2 are true. A *disjunction* $L_1 \vee L_2$ is true iff at least one of L_1 and L_2 is true. An *exclusive disjunction* $L_1 \oplus L_2$ is true iff one of L_1 and L_2 is true and another is false. An *implication* $L_1 \Rightarrow L_2$ is true iff L_1 is false or L_2 is true. Given an implication $L_1 \Rightarrow L_2$, L_1 is called *body* and L_2 is called *head*. An *equivalence* is true iff L_1 and L_2 have the same true value. A literal or literals connected by a logical connections are called *formula*. A logical connection can also connect formulas (e.g. $L_1 \wedge L_2 \Rightarrow L_3 \wedge L_4$).

Each variable in formulas can be quantified. In FOL, there are two types of quantification, *universal quantification* and *existential quantification*. Universal quantification is written as $\forall x_1, x_2, ..., x_n$. An universally quantified formula $\forall x F_x$ is true iff the formula F_x is true for any object x in the world. Existential quantification is written as $\exists x_1, x_2, ..., x_n$. An existential quantified formula $\exists x F_x$ is true iff the formula F_x is true for at least one object x in the world.

A formula is *satisfiable* iff it is possible to find the truth assignments of atoms in the formula which makes the formula true. When truth assignments makes a formula true, we say that the truth assignments *satisfy* the formula. When a formula F_1 is true in every truth assignments which satisfy another formula F_2, F_2 is said to be *entailed* by F_1 and this relation is denoted as $F_2 \models F_1$.

An equality between terms x and y, $x = y$ means that a variable x and a variable y refer same object. In this thesis, we deal with equalities between terms as literals. That is, they can be negated and connected by logical connections. For example, a formula $p(x) \wedge p(y) \wedge x = y$ has the same meaning as $p(x)$. We denote the negation of $x = y$ as $x \neq y$.

In this thesis, we consider that a set of literals and a conjunction of literals is interconvertible. For instance, a conjunction $p(x) \wedge p(y)$ can be written as a literal set $\{p(x), p(y)\}$, and $p(x) \in P$ implicates $P \models p(x)$.

2.2 Abduction

Abduction is inference to the best explanation. In this thesis, we adopt first-order logic as the meaning representation of logical abduction. Formally, logical abduction is defined as follows:

Given: Background knowledge B and observation O, where B is a set of first-order logical formulas, and O is a a first-order formula.

Find: A *hypothesis* (*explanation*) H such that $H \cup B \models O, H \cup B \not\models \bot$, where H is a first-order formula. $\bot$ is a logical constant denoting contradiction. We say that q is *hypothesized* if $H \cup B \models q$ and that q is *explained* by p if $(\exists p)\ p \Rightarrow q \in B$ and $H \cup B \models q$. What we call *equality assumption* is an equality between terms in a hypothesis, such as $x = y$.

Typically, there are several hypotheses H that explain O. We call these the *candidate hypotheses*, each literal in a candidate hypothesis is an *elemental hypothesis*, and the set of literals in all possible candidate hypotheses is called the *potential elemental hypotheses*. In this thesis, we denote potential elemental hypotheses as P. Since a candidate hypothesis is a subset of the potential elemental hypotheses, the potential elemental hypotheses provides the search space of the solution.

The goal of abduction is to find the best hypothesis $\hat{H}$ among the candidate hypotheses by using a specific evaluation measure. We call $\hat{H}$ the *solution hypothesis*. Formally, the solution hypothesis is defined as follows:

$$\hat{H} = \arg\max_{H \in \mathcal{H}} E(H) \tag{2.1}$$

where $\mathcal{H}$ is a set of possible candidate hypotheses, and E is a function $H \to \mathbb{R}$ that evaluates the plausibility of each candidate hypothesis. Here, we assume that $E(H)$ returns $-\infty$ if $H \cup B \models \bot$, and we call this the *evaluation function*. In the literature, several kinds of evaluation functions have been proposed [Charniak and Goldman, 1991; Hobbs et al., 1993; Raghavan and Mooney, 2010; Singla and Domingos, 2011, etc.].

2.2.1 Our Formulation

In this section, we define our formulation of abduction. This formulation is based on the formulation in Inoue and Inui [2011].

At first, let us confine the above definitions of abduction as follows;

- We use function-free first order logic as the meaning representation. A variable or a constant is permissible as a term of literal but a function is not permissible.

- Background knowledge B is a set of implications between conjunctions, where the body of each implication is universally quantified and the head of each implication is existentially quantified. Consequently, each implication can be formally represented as $\forall \mathbf{x}_1^n\ [\exists(\mathbf{y}_1^m \setminus \mathbf{x}_1^n)\ [p_1(\mathbf{x}_1) \wedge ... \wedge p_n(\mathbf{x}_n) \Rightarrow q_1(\mathbf{y}_1) \wedge ... \wedge q_m(\mathbf{y}_m)]]$, where $\mathbf{x}_i$ and $\mathbf{y}_j$ are term arrays.

- Observation O is a conjunction of first-order literals. We assume that all variables occurring in observation are existentially quantified.

- A hypothesis H is a conjunction of first-order literals. Like observation, we assume that all variables occurring in a hypothesis are existentially quantified.

In this thesis, we generally omit quantifiers in observations, background knowledge and hypotheses.

As noted, each candidate hypotheses can be regarded as a subset of the potential elemental hypotheses. Then the enumeration of possible candidate hypotheses can be formulated as the generation of the potential elemental hypotheses. In this thesis, the potential elemental hypotheses are generated by applying the limited number of the following two operations, starting with $P = O$;

Backward chaining: Assuming an axiom $p_1(x) \wedge p_2(x) \wedge ... \wedge p_n(x) \Rightarrow q(x) \in B$ and the potential elemental hypotheses P which contains a literal $q(a)$, this operation adds new literals $\{p_i(a)\}_{i=1}^n$ to the potential elemental hypotheses. For example, applying backward chaining with the axiom $p(x) \Rightarrow q(x)$ to $P = q(A)$, new literal $p(A)$ is added to P as a elemental hypothesis.

Unification: Assuming the potential elemental hypotheses which contains two literals that have the same predicate $p(\mathbf{x}), p(\mathbf{y})$, this operation adds equalities between each term of $\mathbf{x}$ and each term of $\mathbf{y}$. For example, given $P = p(x_1, x_2) \wedge p(y_1, y_2)$, the unification between $p(x_1, x_2)$ and $p(y_1, y_2)$ adds an equalities $x_1 = y_1$ and $x_2 = y_2$ to P.

See Figure 2.1 for an example of potential elemental hypotheses. Here, the observation is $O = animal(x) \wedge bark(e_1, y)$ and the knowledge base consists of following logical formulae:

$$\begin{aligned} poodle(x) &\Rightarrow dog(x) \\ dog(x) &\Rightarrow animal(x) \\ dog(x) &\Rightarrow bark(e, x) \\ cat(x) &\Rightarrow animal(x) \end{aligned}$$

As noted above, the potential elemental hypotheses are generated by applying operations of backward chaining and unification, starting with $P = O$. In this example, the potential elemental hypotheses is initialized as $P = \{animal(x), bark(e_1, y)\}$ and end up as $P = \{animal(x),\ bark(e_1, y),\ cat(x),\ dog(x),\ poodle(x),\ dog(y),\ x = y\}$.

Now, how should we decide the number of operations to apply? In this thesis, following Inoue and Inui [2011, 2012], we adopt the *depth* of a literal to limit the number of operations to apply. The depth of a literal means the number of backward chaining operations which are necessary to add the literal to the potential elemental hypotheses. For instance, the depth of a literal included in observation is 0. The depth of the literal $poodle(x)$ in Figure 2.1 is 2. Supposing d_{max} is the maximum depth, we restrict backward-chaining operations to be applied to literals whose depth exceeds d_{max}. This limitation is important particularly when knowledge base contains looping formulae (e.g. $a \Rightarrow b$ and $b \Rightarrow a$).

Being generated by this procedure, the potential elemental hypotheses consist of the limited number of literals and each literal is observable or hypothesized by the limited number of backward chainings. Consequently, the dicidability of $H \cup B \models O, H \cup B \not\models \bot$ for each candidate hypothesis is obviously guaranteed.

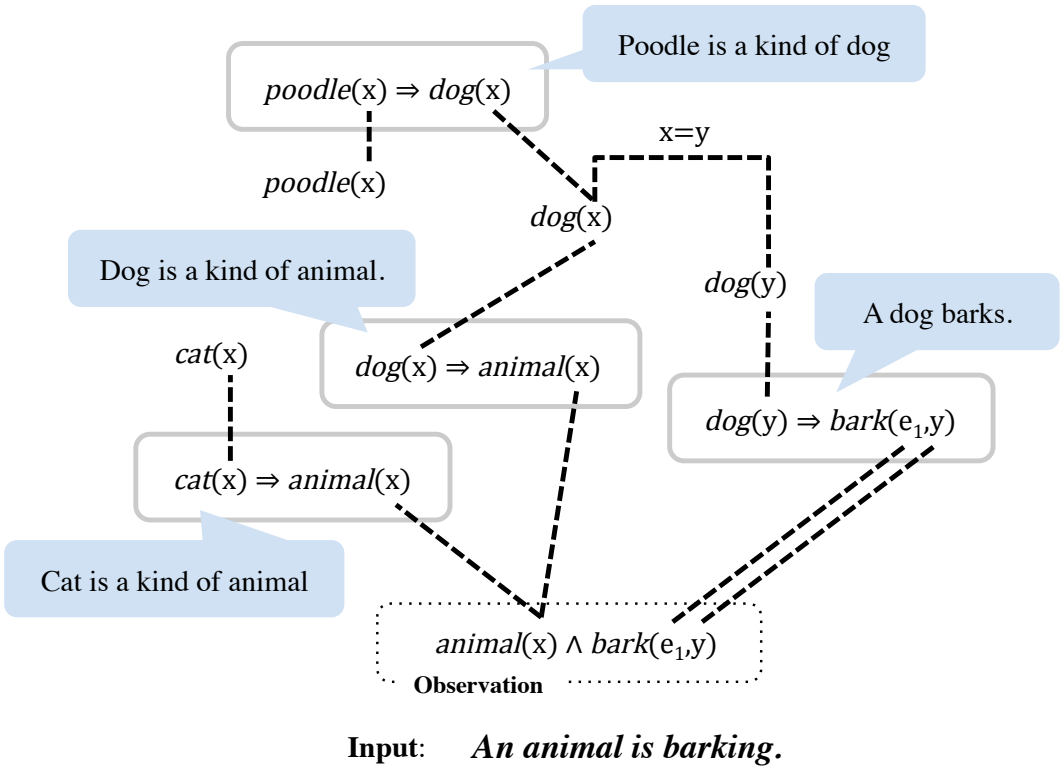

Figure 2.1: An example of elemental hypotheses set.

Moreover, it is guaranteed that the algorithm will halt without running forever.

2.3 Existing Frameworks of Abduction

In this section, we introduce some of existing frameworks of first-order abduction.

2.3.1 Weighted Abduction

Weighted Abduction is a abductive inference model proposed by Hobbs et al. [1993] and is the defacto standard model in the domain of abduction-based discourse processing.

In Weighted Abduction, background knowledge is a set of first-order logical Horn clause whose literals in its body are assigned positive real-valued *weights*, and each literal in an observaion or in a hypothesis has a positive real-valued *cost*.

We use a notation l^w to indicate "a literal l has the weight w" (e.g. $p(x)^{0.6} \wedge q(x)^{0.6} \Rightarrow r(x)$) and $l^{\$c}$ to denote "a literal l has the cost c" (e.g. $p(x)^{\$10} \wedge q(x)^{\$10}$).

In principle, the evaluation function of Weighted Abduction gives penalty for assuming specific and unreliable information but rewards for inferring the same information from different observations. Since a cost of each literal represents how the literal is specific and unreliable, a candidate hypothesis which consists of literals assigned low cost is considered to be plausible. More formally, the evaluation function is defined as the sum of all the costs of elemental hypotheses in it:

$$Eval(H) = -\sum_{h in P_H} c(h) \tag{2.2}$$

where P_H is a set of elemental hypotheses that are not explained nor unified, $c(h)$ is the cost which an elemental hypothesis h has.

Specificity and unreliablity of a literal h is evaluated based on two factors: (1) How the literal explained by h is specific and unreliable and (2) how the formulae used to hypothesize h are unreliable. More formally, given a weight vector θ, the cost of literal h is defined as the multiplication of the cost of the literal explained by h (we denote o_h) and the weight of logical formulae which are used to explain o_h from h.

$$c(h) = \left[\prod_{i \in chain(h)} \theta_i \right] c(o_h) \tag{2.3}$$

where $obs(h)$ is an observed literal that is back-chained on to hypothesize h, $chain(h)$ is a set of indices to a literal in axioms that are used for hypothesizing h from o_h. Henceforth, we refer to a weight vector θ as the parameter of the evaluation function of Weighted Abduction.

The special feature of this model is to be able to evaluate two types of plausibility of hypotheses simultaneously: *correctness* and *informativeness* [1]. Correctness represents how reliable the contents of information are. Informativeness is how specific the information is. This evaluation function is parametrized in a way that one can construct a evaluation function that favors more specific and thus more informative explanations, or less specific but more reliable explanations in

[1] These corresponds to what Thagard [1978] has called *simplicity* and *consilience*

terms of a specific task by altering the parameters.

2.3.2 Weighted Linear Abduction

Since the evaluation function in Weighted Abduction is non-linear, it is hard to compute the gradient of the weights directly. This property prevents one from adopting the gradient algorithm to learn the weight of Weighted Abduction.

Inoue et al. [2012] proposed the linear version of Weighted Abduction(we call **Weighted Linear Abduction**). In this model, instead of Equation 2.3, a cost of a literal h is defined as the sum of the cost of o_h and the weights:

$$c(h) = \left[\prod_{i \in chain(h)} \theta_i \right] c(o_h) \tag{2.4}$$

Unlike the evaluation function of Weighted Abduction, one of Weighted Linear Abduction is linear at their parameters. Therefore, it is relatively easy to tune the parameter of this model. Actually Inoue et al. [2012] shows that the parameters of Weighted Linear Abduction can be learned by instantiating Passive Aggressive algorithm [Crammer et al., 2006].

2.3.3 Markov Logic Network-based Abduction

Blythe et al. [2011] proposed a method that emulates abduction on Markov Logic Networks (MLNs) [Richardson and Domingos, 2006] (we call *MLN-based Abduction*). The evaluation function can be written as follows:

$$E(h) = \sum_{k \in B_h} w(k) \tag{2.5}$$

where B_h is a set of logical formula used to explain the observation from the hypothesis h and $w(k)$ is the weight assigned to a logical formula k. Each weight represents plausibility of backward chaining with the corresponding logical formula.

The advantage of this model is that it can be implemented on existing MLNs frameworks and then it can make use of efficient algorithms for the MLNs frame-

works. However, it is not very efficient because the grounding (i.e., the process that converts the knowledge base or observations in the first-order logic into propositional logic) causes the knowledge base to increase explosively.

2.4 Conclusion

In the formar part of this chapter, we introduced first-order logic and outlined the mechachism of abduction. Abduction framework takes observation and background knowledge as input and returns the best explanation to the observation (we call the solution hypothesis) as output. The goodness of each candidate is evaluated by an evaluation function.

In the rest of this chapter, we introduced several existing abduction frameworks. In following chapters, we basically adopt the evaluation function in Weighted Abduction.

Chapter 3

Boosting Efficiency of Abduction

Abductive inference is an NP-hard problem, and so its computational cost increases exponentially with increases in the knowledge base.

To archive discourse processing on abduction with large knowledge base, it is necessary to solve this big problem. Specifically, in this chapter, we aim to construct a framework of abduction that satisfies the following requirements:

Optimality Given enough time, it can infer the optimal solution in the current search space.

Scalability The length of time which is needed to infer the optimal solution is as short as possible.

Anytime inference Given not enough time to get the optimal solution, it search as good solution as possible — which contains as few contradictions as possible and has as good score on the evaluation function as possible.

3.1 Previous Work

3.1.1 Previous work on for efficient abduction

As noted above, the computational cost of abduction is a big problem. The studies that have addressed this issue can be classified roughly into two groups.

The first includes those methods that emulate abduction by using a framework for deduction [Blythe et al., 2011; Raghavan and Mooney, 2010; Singla and Domingos, 2011, etc.]. For example, Singla and Domingos proposed a method that emulates abduction on Markov logic networks (MLNs) [Richardson and Domingos, 2006]. However, although these methods can make use of efficient algorithms for the target framework, they are not very efficient [Blythe et al., 2011]. The reason of this is that the grounding, i.e., the process that converts the knowledge base or observations in the first-order logic into propositional logic, causes the knowledge base to increase explosively.

The second includes those methods that formulate abduction as the problem of finding the best subset of the potential elemental hypotheses, and then uses another optimization algorithm to search the subset of potential elemental hypotheses that corresponds to the solution hypothesis. For example, Inoue and Inui proposed a method to formulate abductive reasoning as a problem of integer linear programming (ILP) without grounding [Inoue and Inui, 2011, 2012]. With this method, a drastic improvement was achieved by the efficiency of the lifted inference and by using an efficient optimization algorithm in an external ILP solver. Inoue and Inui [2012] reported that this approach is much faster than the MLN-based framework discussed above [Blythe et al., 2011], which had been the state of the art before being replaced by this method.

3.1.2 ILP Formulation of Abduction

In this section, we outline the method by Inoue and Inui, which formulate abduction as an Integer Linear Programming (ILP) problem [Inoue and Inui, 2011, 2012]. Their method can be divided into the three steps as follows:

Generation step: The potential elemental hypotheses are generated from given observation and knowledge base. As noted in section 2.2.1, the potential elemental hypotheses generation is done by applying operations of backward chaining and unification.

Conversion step: The potential elemental hypotheses generated are converted into an ILP problem. Here, whether the elemental hypothesis is included

in the solution hypothesis is expressed as 0-1 value of the corresponding variable in the ILP problem. Constraints in an ILP problem expresses various relations between elemental hypotheses — such as transitive property of equality, mutually exclusiveness between literals and so on. The evaluation function of abduction is expressed as the objective function in the ILP problem.

Optimization step: The solution hypothesis is obtained by optimizing the ILP problem. The optimization of the ILP problem is done by external ILP solver, such as LpSolve[1] and Gurobi Optimizer[2].

As noted above, transitivity constraints for equality assumptions are represented as ILP constraints in the ILP problem. The problem here is that the number of transitivity constraints is $\emptyset(n^3)$, where n is the number of equality assumptions in the potential elemental hypotheses. For this problem, Inoue and Inui [2012] proposed a method to boosting efficiency of ILP optimization by gradually optimizing and adding transitivity constraints if violated in an iterative manner. This is the current state-of-the-art abductive reasoner in terms of computational efficiency. Our method in this chapter is proposed as a extension of their method.

3.2 Basic Strategy

In this section, we discuss the basic strategy of our method.

We begin by discussing the optimality of the solution obtained by the abduction. In abductive reasoning, because the search space of the solution can increase without limit and the proof of global consistency for the negation requires a high computational cost, obtaining the global optimal solution by abductive reasoning is expensive. Therefore, in practice, it is the local, not the global, optimal solution that is sought; that is, we seek the best hypothesis within some limited search space and regard it as the best explanation. In the work of Inoue and Inui [2012], a parameter d_{max} was defined to be a natural number, and the potential elemental hypotheses consist of those elemental hypotheses that can be

[1] http://lpsolve.sourceforge.net/5.5/
[2] http://www.gurobi.com/

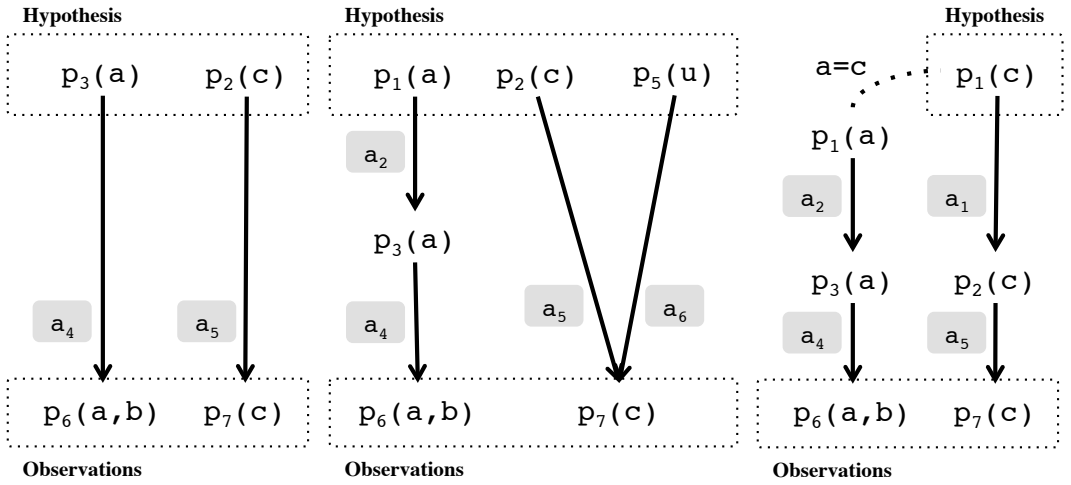

Figure 3.1: An example of the basic strategy.

ID	Axiom	ID	Axiom
a_1	$p_1(x) \Rightarrow p_2(x)$	a_5	$p_2(x) \Rightarrow p_7(x)$
a_2	$p_1(x) \Rightarrow p_3(x)$	a_6	$p_5(y) \Rightarrow p_7(x)$
a_3	$p_4(x, y) \Rightarrow p_5(y)$	a_7	$p_8(y) \Rightarrow p_6(x, y)$
a_4	$p_3(x) \Rightarrow p_6(x, y)$	a_8	$p_9(x) \Rightarrow p_8(x)$

Table 3.1: A knowledge base for an example.

hypothesized through less than d_{max} backward chainings. A larger d_{max} indicates a higher probability that the solution is a global optimum and a correspondingly higher computational cost. The optimality of the solution and its computational cost both depend on the size of the search space of the solution. In this paper, we aim to reduce the size of the search space (i.e., the number of potential elemental hypotheses) while maintaining the optimality of the solution.

3.3 Pruning Non-Reachable Literals

In abduction, the evaluation functions are generally defined so that the better a hypothesis is considered to be, the greater the probability of the assump-

tions included in the hypothesis and the more observations it explains. For example, given the knowledge base shown in Table 3.1 and an observation $O = \{p_6(a, b), p_7(c)\}$, let us consider the three hypotheses shown in Figure 3.1. Here, the hypothesis (b) is less optimal than hypothesis (a), because (b) includes more hypothesized literals than (a) but explains the same number of observations. On the other hand, since hypothesis (c) explains as many observations as (a) with fewer literals, (c) is considered to be better than (a). More formally, the evaluation functions E generally have the following properties:

1. Given a candidate hypothesis H and an operation of backward chaining c, $E(H) \geq E(H \cap c)$ is satisfied.

2. A candidate hypothesis H and an operation of unification u that satisfy $E(H) \leq E(H \cap u)$ can exist.

Supposing that the evaluation function that we employ has these properties, then we can reduce the number of potential elemental hypotheses by canceling the backward chainings that do not result in unification.

In order to estimate whether the backward chaining will result in unification, it is necessary to know which literals can be hypothesized from each observation and the plausibility of each literal. Here, we define the function $h^*(p, q)$, which provides the semantic relatedness between a literal p and a literal q. We call the return value of $h^*(p, q)$ the *heuristically estimated distance* (**HED**) between p and q.

The necessary conditions of $h^*(p, q)$ and HED are as follows. First, they must express the semantic relatedness between p and q. In other words, the more easily the relevance between two literals can be inferred, the higher the HED between them. Second, $h^*(p, q)$ must be admissible for use in an A* search, so that it can be employed as a heuristic for the cost, as in Section 3.4. Third, the computational cost for obtaining a return value from $h^*(p, q)$ should be as small as possible. For the third condition, we pre-estimate all of the HEDs and store them in a database. Thus, the function $h^*(p, q)$ only has to load values from memory. Since the size of the database of HEDs increases as the definition of $h^*(p, q)$ becomes more complex, we have to consider the balance between efficiency and the expressiveness of the HEDs.

Therefore, we define this function as the heuristic distance between the predicates of the literals with the abstraction of the conjunctions of the antecedents of each of the axioms. Formally, $h^*(p, q)$ is defined as follows:

$$h^*(p,q) = \min_{H \in \{H | H \cup B^* \models \{\rho(p), \rho(q)\}\}} \sum_{a \in A_H} \delta(a) \tag{3.1}$$

$$B^* = \bigcup_{p_1 \wedge \ldots \wedge p_n \Rightarrow q \in B} \left[\bigcup_{i=1}^{n} \rho(p_i) \Rightarrow \rho(q) \right] \tag{3.2}$$

where A_H is the set of axioms that are used in H, $\rho(L)$ is the function that returns the literal corresponding to the predicate of the first-order literal L (e.g., $\rho(john(x)) = john$). and $\delta(A)$ is the *distance function*, which returns the heuristic distance between the antecedents of the axiom A and the conclusions of A. For example, given the knowledge base in Table 3.1 and the distance function $\delta(A) = 1$, the value of $h^*(p_7(x), p_1(x))$ is $\delta(a_5) + \delta(a_1) = 2$.

In this paper, we define the distance function as $\delta(A) = 1$, for simplicity. In practice, it is necessary to select a proper distance function because the precision of the HEDs depends on the definition of the distance function. For example, in cost-based abduction [Inoue and Inui, 2012], the distance function better conforms to the evaluation function when using the cost assigned to each axiom for $\delta(A)$.

Since the HEDs depend only on the knowledge base, we can estimate these in advance. The computational cost of the estimation is $O(N_{pred}^2)$, where N_{pred} is the number of different predicates in the knowledge base.

3.4 Potential Elemental Hypotheses Creation with A* Search

In this section, we propose an algorithm that efficiently creates the potential elemental hypotheses. We apply an A* search to generate the potential elemental hypotheses and then trim without loss any that are included in the solution hypothesis. Although we employ the same evaluation function as used by *weighted abduction*, our method can be applied to other frameworks.

Now, our goal is to efficiently hypothesize the literals that can be combined.

Algorithm 1 A* search-based potential elemental hypothesis creation

Require: $B, O = \{o_1, o_2, ..., o_n\}, l_{max}, d_{max}$

$X \leftarrow \emptyset$ // The open set
$P \leftarrow \emptyset$ // The potential elemental hypotheses
for $i = 1$ to n **do**
  for $j = 1$ to $i - 1$ **do**
    $U \leftarrow getEqualityAssumption(o_i, o_j)$
    $P \leftarrow P \cup U$
    if $h^*(o_i, o_j) > 0$ **then**
      $X \leftarrow X \cup x,\ x.c = o_i \wedge x.s = o_i \wedge x.g = o_j$
      $X \leftarrow X \cup x,\ x.c = o_j \wedge x.s = o_j \wedge x.g = o_i$
    end if
  end for
end for
while $X \neq \emptyset$ **do**
  $\hat{x} \leftarrow \arg\min_{x \in X}\{d(x.s, x.c) + h^*(x.c, x.g)\}$
  for all $a = \{p_1 \wedge p_2 \wedge ... \wedge p_n \Rightarrow q\}$ in B **do**
    $R \leftarrow doBackwardChaining(\hat{x}.c, a)$
    $P \leftarrow P \cup R$
    for all r in R **do**
      for all x in $\{x | x \in X \wedge x.c = \hat{x}.c\}$ **do**
        if $d(x.s, x.c) + h^*(x.c, x.g) + \delta(a) \leq l_{max} \wedge depth(x.c) < d_{max}$ **then**
          $X \leftarrow X \cup \{y | y.s = x.s \wedge y.c = r \wedge y.g = x.g \wedge d(y.s, y.c) = d(x.s, x.c) + \delta(a)\}$
        end if
      end for
      for all p in $P \setminus r$ **do**
        $U \leftarrow getEqualityAssumption(r, p)$
        $P \leftarrow P \cup U$
        if $U \neq \emptyset$ **then**
          $X \leftarrow X \setminus \{x | x.c = r \wedge isExplanation(p, x.g)\}$
          $X \leftarrow X \setminus \{x | x.c = p \wedge isExplanation(r, x.g)\}$
        end if
      end for
    end for
  end for
  $X \leftarrow X \setminus \{x | x.c = \hat{x}.c\}$
end while
return P

Algorithm 2 *doBackwardChaining*(l, a)

Ensure: $a = \{p_1 \wedge p_2 \wedge ... \wedge p_n \Rightarrow q\}$

$P \leftarrow \emptyset$
if $\exists\theta,\ l\theta = q\theta$ **then**
  for $v \in notSubstitutedVars(\{p_1, p_2, ..., p_n\}, \theta)$ **do**
    $\theta \leftarrow \theta \cup \{v/u_i\};\ i \leftarrow i + 1$
  end for
  $P \leftarrow P \cup \{p_1, p_2, ..., p_n\}\theta$
end if
return P

Algorithm 3 *getEqualityAssumption*(p_1, p_2)

$P \leftarrow \emptyset$
if $\exists\theta,\ p_1\theta = p_2\theta$ **then**
  for all x/y in θ **do**
    $P \leftarrow P \cup \{x = y\}$
  end for
end if
return P

Since we cannot know exactly which axiom we should use in order to hypothesize those literals, we search for them by using the HEDs, as follows.

First, set positive values for l_{max} and d_{max}, which are hyperparameters that control the size of the search space and initialize the open set to be an empty set. We denote the distance of the path from a literal p to a literal q as $d(p, q)$ and the estimated distance between p and q as $d^*(p, q)$. We use the distance function $h^*(p, q)$ as the heuristic function that provides $d^*(p, q)$. In each step, the following operations are performed:

- Select the target literal $\hat{q}$, which is expected to result in the least expensive unification with the literals in the open set.
- Pop $\hat{q}$ off the open set. Enumerate the axioms whose descendant equals $\hat{q}$, and perform backward chaining with each of the axioms with the condition that at least one pair of a literal p_i in the antecedents of the axiom and a literal o in the observations satisfies the following conditions: (i) p_i is considered to be reachable by o (i.e., $h^*(p_i, o) \leq l_{max}$); (ii) there is no

possibility of unification between one of the descendants of p_i and one of the antecedents of o.

- If a literal in X and one in the potential elemental hypotheses are unifiable, insert the elemental hypotheses of equality between the terms resulting from the unification.

The search is over when the open set is empty.

For example, given the knowledge base shown in Table 3.1 and an observation $O = \{p_2(a), p_6(b, c), p_7(d)\}$, the first step of the search is performed as shown in Figure 3.2; the edges drawn with a solid line represent backward chaining, and those drawn with a dotted line are unifications. The numbers in the balloons connected to the nodes in the open set indicate the estimated distance. In the initial step, since the shortest path is expected to be the one between $p_7(d)$ and $p_2(a)$, the literals $p_2(d)$ and $p_5(u_1)$ are inserted into the open set as the results of backward chainings.

The procedure is shown in Algorithm 3.4, X is the open set for the search. Each element $x \in X$ is a candidate for the search and has three possible designations: $x.s$ is the start node, $x.c$ is the current node, and $x.g$ is the goal node. The function $isExplanationOf(x, y)$ is the binary function that indicates which the literal x explains the literal y (i.e., if x is an antecedent of y), and the function $depth(p)$ returns the number of backward chainings that are needed to hypothesize the literal p from the observations.

Next, we summarize the advantages of this algorithm. First, since this algorithm does not add literals that cannot be included in the solution hypothesis to the potential elemental hypotheses, it can reduce the size of the search space. We believe that this may lead to a more efficient optimization.

Second, this algorithm prevents redundant unifications. For example, given the knowledge base shown in Table 3.1 and the observation $O = \{p_7(a), p_7(b)\}$, let us consider how to generate the potential elemental hypotheses P. In Inoue and Inui [2011], the potential elemental hypotheses generated are $P = \{p_2(a), p_2(b), p_1(a), p_2(b)\}$. However, according to Section 3.2, the evaluation of the candidate hypothesis $H = \{a = b\}$ must be better than the evaluation of $H = \{p_2(a), p_2(b), a = b\}$ or $H = \{p_1(a), p_1(b), p_2(a), p_2(b)$. We have no need to consider backward chainings

from observations in this case. Our algorithm can deal with such a heuristic.

Third, this algorithm adds elemental hypotheses to the potential ones in the order of their probability of being included in the solution. Therefore, if the generation of potential elemental hypotheses is interrupted, such as by timing out, a better suboptimal solution is provided. This property is expected to be much more useful in practice.

3.5 Parallelization

In the domain of the efficiency of other frameworks for inference, some researchers have adopted the approach of parallelizing the inference by splitting the input into independent subproblems [Gonzalez et al., 2009; Jojic et al., 2010; Niu et al., 2012; Urbani et al., 2009]. In this section, we propose a similar method to parallelize abductive inference by using HEDs, which were proposed in the previous section.

First, we consider the condition that two subproblems o_i and o_j are independent. This condition is defined by the particular evaluation function that is used. For instance, in weighted abduction, the conditions can be defined as follows:

1. There is no elemental hypothesis that explains both the literals $p \in o_i$ and $q \in o_j$ (i.e. $\min\{h^*(p,q), p \in o_i \wedge q \in o_j\} = \infty$).

2. Equalities between any two terms cannot be hypothesized from o_i and o_j together. In other words, o_i and o_j can share no more than one logical variable.

Given observations O, the inference is parallelized via the following process:

1. Split the observations O into independent subproblems $o_1, o_2, ..., o_n$.

2. Compute in parallel the solution hypothesis for each subproblem.

3. Merge the solution hypotheses of the subproblems, and then output the solution hypothesis of O.

As mentioned, the computational cost of abduction grows exponentially with the number of observations. Therefore, dividing the observations into subproblems not only reaps the benefits of parallel computing, but it is also expected to reduce the total computational cost.

3.6 Evaluation

In this section, we reported the results of two experiments to evaluate the efficiency of our methods.

3.6.1 Common Setting

Dataset We used the same dataset as the one used by Inoue and Inui [2012]; it consists of sets of observations and a knowledge base. The observation sets were created by converting the development dataset of RTE-2[1], the task of Textual Entailment Recognition, with the Boxer semantic parser[2]; it consists of 777 observation sets. The average number of literals in each observation set was 29.6.

The knowledge base consists of 289,655 axioms that were extracted from WordNet [Fellbaum, 1998a], and 7,558 that were extracted from FrameNet [Ruppenhofer et al., 2010]. The number of different predicates in this knowledge base is 269,725.

Evaluation Function We employed Weighted Abduction [Hobbs et al., 1993] as the evaluation function. We manually assigned the weights to each axiom.

Machine and ILP solver For our experiments, we used a 12-Core Opteron 6174 (2.2 GHz) 128 GB RAM machine. We used a Gurobi optimizer[3], which is an efficient ILP solver. It is a commercial product but is freely available with an academic license.

[1] http://pascallin.ecs.soton.ac.uk/Challenges/RTE2/
[2] http://svn.ask.it.usyd.edu.au/trac/candc
[3] http://www.gurobi.com/

	Baseline ($d_{max} = 3$)	**A*-single** ($l_{max} = 6$)
# of literals	1059	233
# of chains	1013	189
# of unifications	395	114
Time (P-Gen)	0.2	0.8
Time (ILP-Conv)	0.2	0.04
Time (ILP-Solve)	15.6	3.8
Time (ALL)	15.9	3.8
# of timeout	48	16

Table 3.2: The result of the comparison between our methods and the baseline.

3.6.2 Evaluation of efficiency

3.6.2.1 Setting

On this experiment, we compared the solving times when using our models and when using that of Inoue and Inui [2012], which is currently the state of the art. We will denote their model as **Baseline** and ours as **A*-single** and **A*-parallel**. **A*-based** will be used to refer to both of A*-single and A*-parallel. We also compared the computational costs for pre-estimating the HEDs with various l_{max}.

In the experiment, the parameter d_{max} was 3, and the parameter l_{max} of A*-based was 6. We employed weighted abduction [Hobbs et al., 1993] as the evaluation function. We defined the distance function $\delta(a) = 1$ for simplicity, and so that the search space on A*-based was equal to that of Baseline.

For our experiments, we used a 12-Core Opteron 6174 (2.2 GHz) 128 GB RAM machine. We used a Gurobi optimizer[1], which is an efficient ILP solver. It is a commercial product but is freely available with an academic license. We excluded from the results those observations in which the optimization took more than 3600 seconds in at least one setting.

3.6.2.2 Results

The results of the first experiment are shown in Table 3.2. The row **# of literals** shows the average number of literals in the potential elemental hypotheses, the

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

	Time	**File Size**
$l_{max} = 4$	106	0.8 GB
$l_{max} = 6$	1514	5.8 GB
$l_{max} = 8$	7841	28 GB

Table 3.3: The computational cost of pre-estimating the HEDs.

row **# of chains** shows the average number of backward chainings in the potential elemental hypotheses, and the row **# of unifications** shows the average number of unifications in the potential elemental hypotheses.

Time (P-Gen) shows the average time (seconds) required to generate the elemental hypotheses, **Time (ILP-Conv)** shows the average time (seconds) required to convert the elemental hypotheses into an ILP problem, **Time** shows the average time (seconds) required to optimize the ILP problem, and **# of timeout** shows the number of problems that timed out.

From Table 3.2, we can observe that there were fewer potential elemental hypotheses in our A* search-based system than in the baseline system, and the time for optimization was shorter.

We compare the results of the optimization times for A*-single and A*-parallel in Figure 3.3; the x-axis is the inference time (seconds) for the Baseline system, and the y-axis is the inference time (seconds) for our system (A*-Single or A*-Parallel).

We can see from Figure 3.3 that, for complex problems, A*-parallel is more efficient than A*-single. On the other hand, for simple problems, A*-parallel is less efficient. We assume that this is because there is overhead required to split the input into subproblems and to initiate the parallel threads.

The costs for pre-estimating the HEDs are compared in Table 3.3. We see that the computational cost and the size of the database increase sharply as l_{max} increases. However, in practice, it is sufficient if l_{max} is in the range of 4 to 8, and so we believe that this cost may not be a bottleneck.

3.6.3 Evaluation of capability for anytime inference

In this section, we show that A* algorithm improve the optimality of the solution hypothesis under the condition that the inference time is limited. Specifically we show the following two things: (1) it improves evaluation value of the solution hypothesis to control the order of the backward chaining operations with using A* algorithm and (2) it improves evaluation value of the solution hypothesis to use a distance function conforming to the evaluation function.

3.6.3.1 Setting

In this experiment, we applied Weighted Abduction to the dataset and compared the result using different distance functions in A*-based Abduction. Here we limited the number of hypothesized literals in the potential elemental hypotheses to $[10, 20, 30, 40, 50, 60, 70, 80]$. The potential elemental hypotheses generation was interrupted when the limit was exceeded.

We used the following distance functions:

NO-SEARCH We use $\delta(a) = 0$ as the distance function. Here, a system considers only whether a literal pair is reachable and does not control the order of backward chaining operations.

CONST We use $\delta(a) = 1$ as the distance function. Each heuristic distance corresponds to the number of backward chaining operations necessary to connect corresponding literals.

WEIGHT We use $\delta(a) = \sum_{w \in W(a)} w$ as the distance function, where $W(a)$ is a sequence of weights for Weighted Abduction assigned to the literals in the body of logical formula a.

3.6.3.2 Results

The result of the experiment is shown in Figure 3.4. The horizontal axis represents the limit of the number of hypothesized literals in the potential elemental hypotheses, and the vertical axis the average of evaluation value of the solution

hypotheses. From this result, we can observe two things: (1) it improves evaluation value of the solution hypothesis to control the order of the backward chaining operations with A* algorithm and (2) it improves evaluation value of the solution hypothesis to use a distance function onforming to the evaluation function.

3.7 Conclusion

While abduction has long been considered to be a promising framework for making explicit the implicit information in sentences, its computational complexity has hindered the application of abduction to practical NLP problems. In this paper, we proposed a method that is an improvement over the method of Inoue and Inui [2012], which is the current state-of-the-art system. Specifically, we proposed a method that eliminates the redundant literals from the potential elemental hypotheses by using an A* search; we then showed that this improves the efficiency of the system. We also proposed a method that splits the input into subproblems and then uses parallel abductive inference; we presented results confirming the efficiency of parallelization.

In our future work, since our methods have a strong dependence on the precision of the pre-estimates, we will refine the definition of the HEDs. We note that currently the estimation is imprecise when a predicate does not have a concrete meaning and tends to occur with other literals in axioms; for example, this happens with the literals for functional verbs. This problem occurs because an axiom $p_1 \wedge p_2 \Rightarrow q$ in the knowledge base is split into the axioms $p_1 \Rightarrow q$ and $p_2 \Rightarrow q$ during the pre-estimation. Therefore, it is important to determine how to enrich the functionality of the pre-estimation without causing the computational cost to explode.

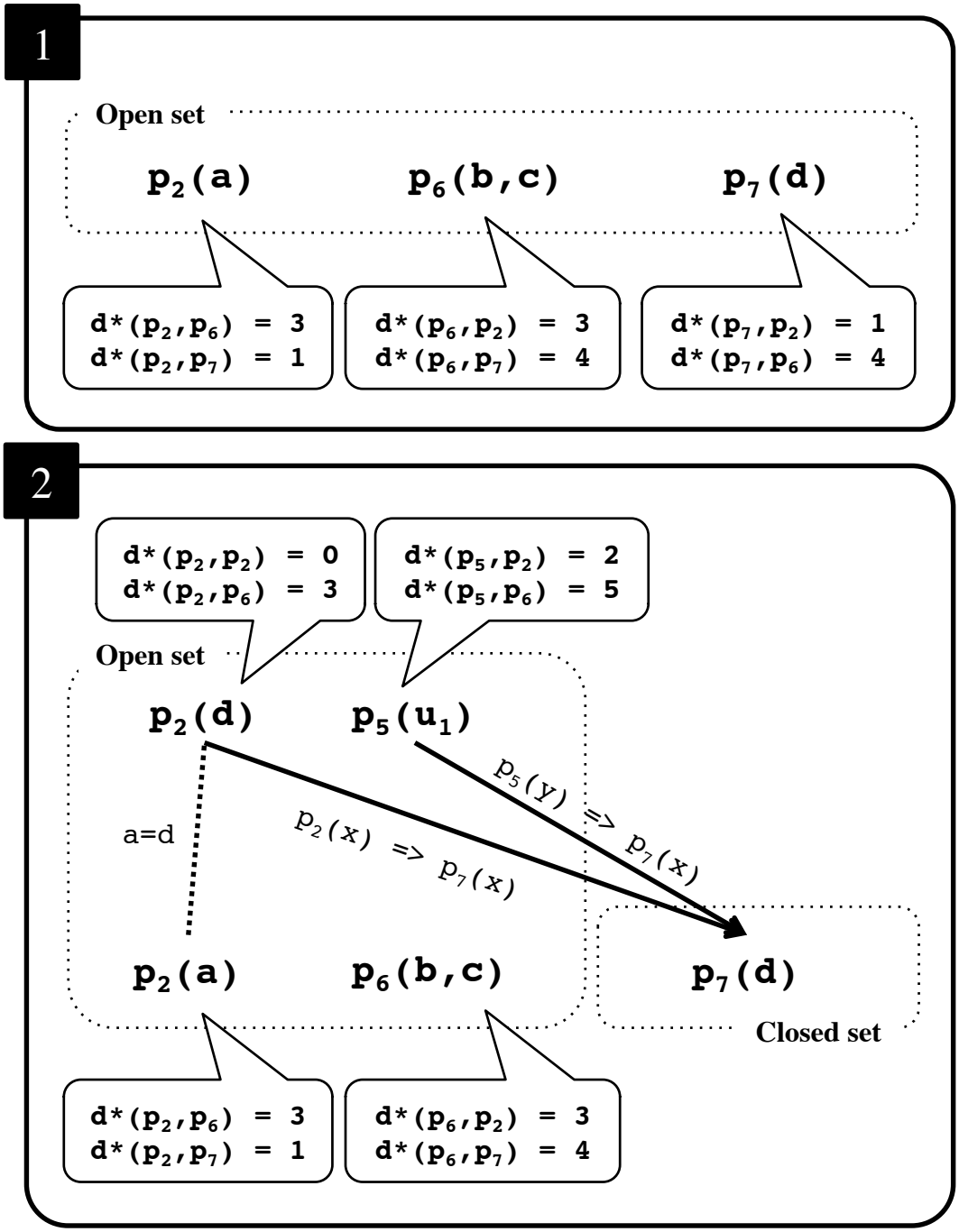

Figure 3.2: An example of the creation of potential elemental hypotheses based on an A* search.

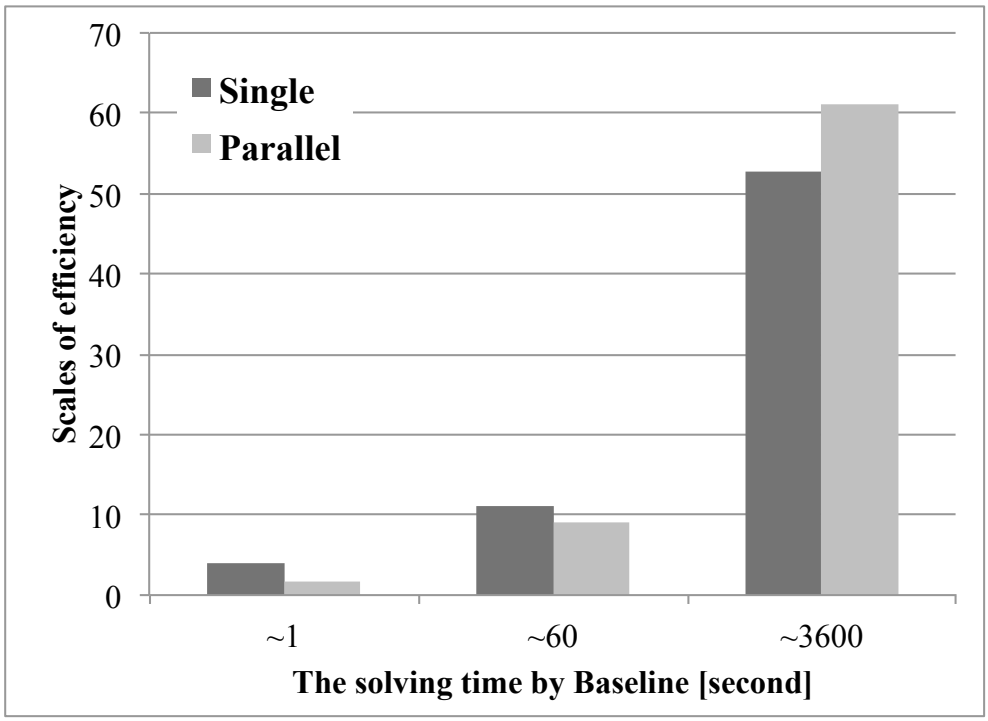

Figure 3.3: The comparison between the single thread system and the parallel thread system.

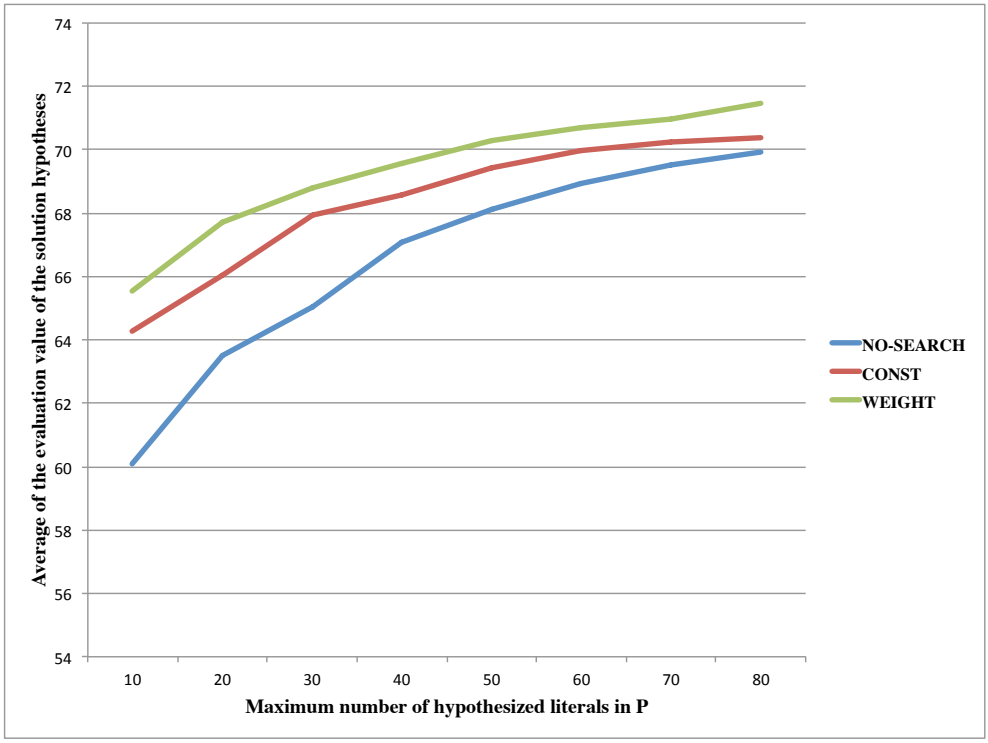

Figure 3.4: The comparison amon the goodness of the solution hypotheses by three distance functions.

Chapter 4

Boosting Efficiency of Abduction for Discourse Processing

We proposed a method to make efficient general first-order abduction in Chapter 3. In this chapter, we propose a method to make efficient abductive inference-based frameworks for discourse processing based on the method in Chapter 3.

4.1 Computational Inefficiency Caused by Literals of Dependency

In this section, we introduce the problem which abductive inference-based frameworks for discourse processing on existing implementations [Inoue and Inui, 2011; Inoue et al., 2012] have.

4.1.1 Preliminary

At first we introduce the meaning representation and the evaluation function which we suppose.

As stated in Chapter 2.2, evaluation functions of abduction are the function to evaluate the goodness of each candidate hypothesis. This goodness is considered to be decided from two factors at least; (1) the goodness of what the hypothesis expresses and (2) the well-formedness — whether the meaning rep-

resentation expressed by the hypothesis is syntactically correct. Our method is to make abduction efficient only if certain presupossitions for these factors are satisfied. In this section, we outline the presuppositions we have for the meaning representation and the evaluation functions.

(1) The Meaning Representation What logical representation to express the information extracted from natural language expressions is an important issue. Specially, the logical representation of thematic roles have been discussed controversially [Copestake et al., 2005; Davidson, 1980; Hobbs, 1985; McCord, 1990; Parsons, 1990, etc.]. The representative methods among them are *Davidsonian* [Davidson, 1980] and *Neo-Davidsonian* [Parsons, 1990].

In Davidsoninan, thematic roles are represented as a term of the literal of the event. For example, let us consider the sentence "*Brutus stabbed Caesar with a knife.*" This sentence may be expressed as $stab(e, Brutus, Caesar) \wedge with(e, knife)$ in Davidsoninan, where the first term of $stab(e, Brutus, Caesar)$ corresponds to the event variable (i.e., the variable e refers the event of "*stab*" itself), the second term corresponds to the agent of the event (i.e., the variable $Brutus$ is the agent of the event of "*stab*") and the third term corresponds to the object of the event (i.e., the variable $Caesar$ is the object of the event of "*stab*").

On the other hand, in Neo-Davidsonian, all of themantic roles are represented as a individual literal. For example, the sentence above may be expressed as $stab(e) \wedge nsubj(e, Brutus) \wedge dobj(e, Caesar) \wedge with(e, knife)$ in Neo-Davidsoninan, where $nsubj(e, x)$ is a literal to mean that the nominal subject of the event e and $dobj(e, x)$ is a literal to mean that the direct object of the event e. In this thesis, which we call **functional literal** is a literal to express syntactic dependency between words such as $nsubj(e, x)$, and which we call **functional predicate** is the predicate of a functional literal. On the other hand, **content predicate** mean predicates which is not a funcational predicate and **content literal** mean a literal with content predicate, such as $stab(e)$. We say a content literal l_c is the parent of a functional literal l_f iff l_c contains the governor of the dependency represented by l_f as its argument. For example, in above logical formula in Neo-Davidsonian, $stab(e)$ is the parent of $nsubj(e, x)$.

Compared with Davidosnian, Neo-Davidsoninan is considered to have several

advantage as follows:

- It can express partial reasoning for an event. For example, knowledge that a *police* can be the nominal subject of *arrest* can be expressed as $police(x) \Rightarrow arrest(e) \wedge nsubj(e, x)$.
- There is no need to determine whether each role is essential or optional. On the other hand, in Davidsonian, since the expression of essential roles differ from of optional roles (i.e., the essential roles are expressed as terms of the literal of the event and optional roles are expressed as individual literals), one must determine which roles are essential.

Since verbs in natural language vary in essential roles, Neo-Davidsonian is considered to be more suitable to deal with real-world sentences than Davidsonian. Consequently, in this thesis, we suppose the meaning representation to be based on Neo-Davidsonian.

As noted above, functional literals represent dependencies between words in natural language. Therefore, the functional literals which have no parent are considered to be syntactically invalid. In this chapter, we presuppose that all observations satisfy the following condition:

Condition 1. None of observation contain a functional literal which has no parent.

We consider observations which does not satisfy this condition to be syntactically invalid and believe that such observation is not given as input.

(2) Evaluation functions Firstly, we suppose that an evaluation function is able to evaluate the validness of equality assumptions in a candidate hypothesis. As stated Section 2.2.1, equality assumptions are generated by operations of backward chaining and unification in the process to generate the potential elemental hypotheses. Here, this process can generate the candidate hypothesis which claims the equality between unequal entities, such as $smart(e_1) \wedge foolish(e_2) \wedge e_1 = e_2$. we call such equality assumptions **invalid** and denote invalid equality assumption between e_1 and e_2 as $e_1 =^* e_2$. In this chapter, we presuppose that an evaluation function satisfies following condition:

Condition 2. An evaluation function does not choice a candidate hypothesis which contains invalid equality assumptions as the solution hypothesis.

where we adopt the Closed World Assumptions for equalities between variables in order to determine the validity of equality assumptions. Therefore, an equality assumption $a = b$ is invalid ($a =^* b$) iff the terms a, b cannot have the same type in potential elemental hypotheses P — iff P contains no pair of content literals which can be unified and introduce $a = b$.

Secondly, we suppose that an evaluation function is able to evaluate whether logical formulae in a candidate hypothesis are syntactically valid. Since functional literals with no parents are syntactically invalid as noted above, we presuppose that an evaluation function satisfies following condition:

Condition 3. An evaluation function does not choice a candidate hypothesis which contains functional literal with no parents as the solution hypothesis.

In the following section, what we call the **Validity Condition** is the set of condition 1, 2 and 3.

4.1.2 Computational inefficiency caused by functional literals

One problem in first-order abdutive inference-based discourse processing is that the operations of backward chaining and unification for functional literals can introduce invalid equality assumptions and then cause the computational inefficiency.

We show two example in Figure 4.1 and Figure 4.2. In Figure 4.1, applying unification to two observable functional literals $nsubj(e_3, j)$ and $nsubj(e_4, t)$, equality assumptions $e_3 = e_4$ (which means that *John* and *Tom* are coreferent) and $j = t$ (which means that *smart* and *foolish* are coreferent) are added to the potential elemental hypotheses. In Figure 4.2, the equality assumption $e_1 = e_2$ is assumed in order to apply the backward chaining to $smart(e_1)$ and $nsubj(e_2, t)$.

Although these hypotheses are logically valid, as noted in Section 4.1.1, they cannot be the best explanation. Wrong operations like these are generated by the combination of literals with same predicate (e.g. the number of literals with *smart*

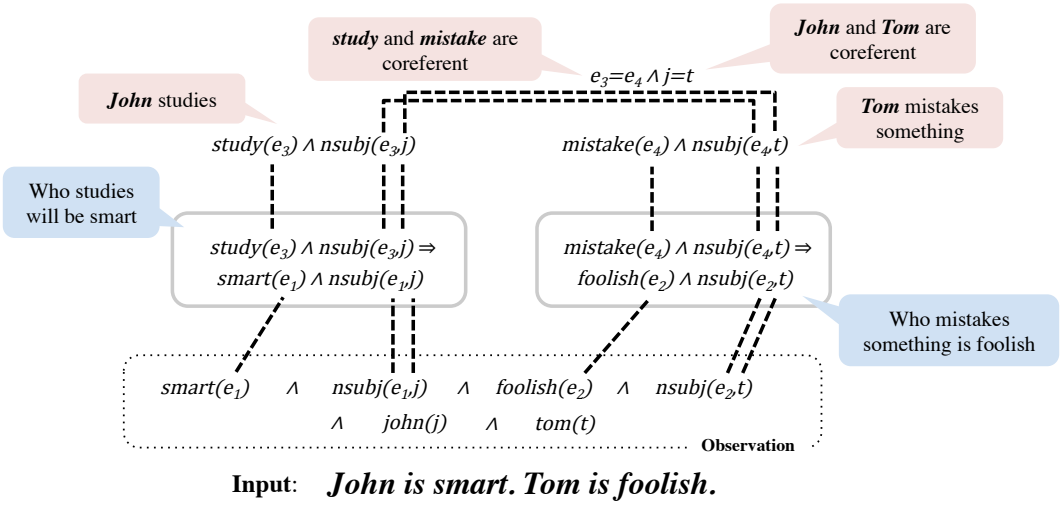

Figure 4.1: An example of problematic unification.

$\times$ the number of literals with *nsubj*) and exponentially increases the number of candidate hypotheses. It is considered to be critical for the computational cost of abduction.

This problem is essentially caused because the semantic validity of equality assumptions are not taken into account on the generation of the potential elemental hypotheses (noted in Section 2.2.1). For instance, in Figure 4.1, it is not considered whether the equality assumption $e_3 = e_4$ is valid (i.e., whether the hypothesis that *study* and *mistake* are coreferent is feasible), and then $e_3 = e_4$ will be added to the potential elemental hypotheses even though it is invalid.

One may consider this problem to be peculiar in Neo-Davidsoninan — in the meaning representation which expresses each thematic role as an individual literal. However, this problem can occur not only in Neo-Davidsoninan but also other meaning representations. For example, in the meaning representation of Hobbs et al. [1993], the semantic relation between nouns (e.g. part-of relation) and syntactic relation (e.g. the dependency between nouns which consists of a noun phrase) are expressed as $part\_of(x, y)$ and $nn(x, y)$. These literals cause similar problem to above problem but is necessary to represent information from natural language in first-order logic. Consequenty, the problem discussed in this section is considered to be important in the domain of IA.

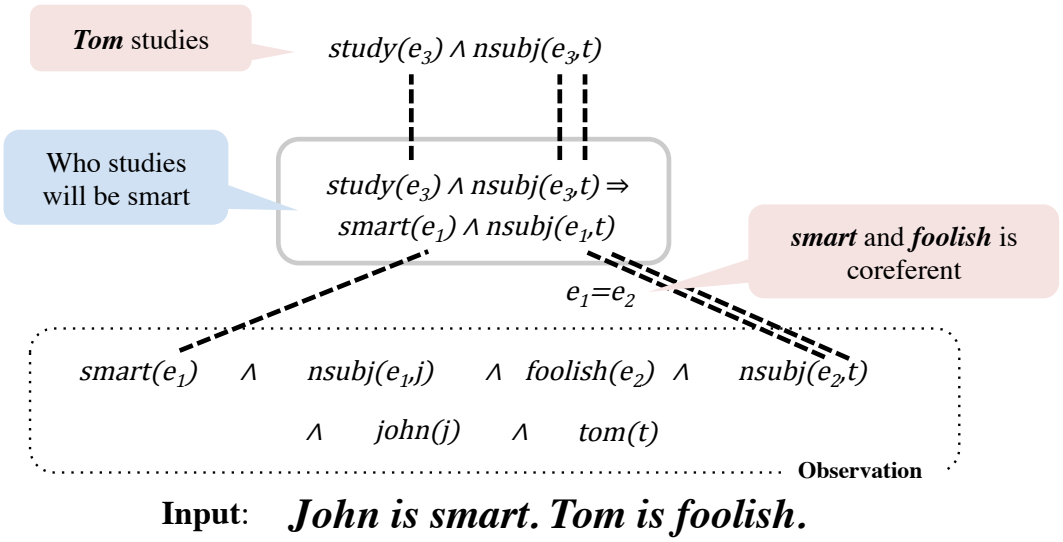

Figure 4.2: An example of problematic backward-chaining.

In this chapter, we propose a method to solve this problem, which prohibits the operations to cause invalid equality assumptions on the potential elemental hypotheses generation and then excludes invalid equality assumptions from the potential elemental hypotheses. For instance, before applying unification to $nsubj(e_3, j)$ and $nsubj(e_4, t)$ in Figure 4.1, we check the validity of $e_3 = e_4$ with using the criteria noted in Section 4.1.1. If we knew that *study* and *mistake* cannot be coreferent, the equality assumption $e_3 = e_4$ is invalid and cannot be contained in the solution hypothesis and therefore this unification operation will be canceled.

In Section 4.2, we extend the procedure for generation of the potential elemental hypotheses in Section 2.2.1 to improve the computational efficiency. In Section 4.3, we propose a method based on the above idea to improve the computational efficiency of A*-based Abduction noted in Chapter 3.

In the following section, we assume that all functional literals have the following format for convenience:

- All functional predicates takes 2 arguments.
- The first argument of a functional literal l_f corresponds the governor of the

dependency which l_f expresses.

- The second argument of a functinal literal l_f corresponds the dependent of the dependency which l_f expresses.

Dependency in natural language is generally expressed as a binary relation and a multi-relation can be generalize to a combination of binary relations. Therefore these assumptions are considered to maintain generality of the meaning representation.

4.2 Boosting Efficiency by Requirement about Equality Assumptions

In this section, we propose the method to exclude invalid equality assumptions from the potential elemental hypotheses by imposing a condition on the operations of backward chaining and unification in Section 2.2.1.

4.2.1 Requirement for unification for functional literals

As noted in Section 2.2.1, the operations of unification in existing framworks [Inoue and Inui, 2011, 2012] are applied to all literal pairs sharing a same predicate. However, if the meaning representation contains functional literals, this procedure may generates invalid equality assumptions as elemental hypotheses. Supposing that the Validity Condition is satisfied, a candidate hypothesis which contains invalid equality assumptions cannot be the solution hypothesis and therefore it cause computational inefficiency to include such a candidate hypothesis in the search space.

In order to address this problem, we propose to allow an operation of unification for a pair of functional literals only if the potential elemental hypothesese have already contain a valid equality assumption between variables corresponding to their governor (i.e., the first term of each literal). For example, application of the unification to $nsubj(e_3, j)$ and $nsubj(e_4, t)$ is allowed only if the equality assumption $e_3 = e_4$ is contained in the potential elemental hypotheses. This requirement prevents functional literals whose parents cannot be coreferent from

being unified and then prevents invalid equality assumptions from being generated as elemental hypotheses.

More formally, given the potential elemental hypotheses P and a functional predicate d, the unification between functional literals $d(x_1, y_1)$ and $d(x_2, y_2)$ is allowed only if x_1 and x_2 are identical or the equality assumption $x_1 = x_2$ is contained in P. In Section 4.2.3, we discuss the implementation of this requirement.

4.2.2 Extension of the requirement to backward chaining

In this section, we consider to impose a requirement like one proposed in the previous section to the operations of backward chaining. For instance, in Figure 4.2, we propose that application of the backward chaining with the logical formula $study(e_3) \wedge nsubj(e_3, t) \Rightarrow smart(e_1) \wedge nsubj(e_1, t)$ to $smart(e_1) \wedge nsubj(e_2, t)$ is allowed only if the potential elemental hypotheses contain the valid equality assumption $e_1 = e_2$. This requirement can exclude backward chaining to introduce invalid equality assumptions from the search space of the potential elemental hypotheses generation.

What should be noted here is that, if one imposes the above requirement on all of backward chaining operations, it can prune candidate hypotheses containing no invalid equality assumptions even though they should not be pruned. We show an example in Figure 4.3. If one imposes the above requirement on the backward chaining operation in Figure 4.3, the equality assumption $x_1 = x_3$ is necessary to perform the backward chaining. Reversely, the backward chaining is necessary to introduce $x_1 = x_3$ and therefore this backward chaining cannot be performed. However the candidate hypothesis shown in Figure 4.3 does not contain any invalid equality assumption and should not be pruned. To address this problem, we exclude backward chaining to cause a problematic case — a candidate hypothesis containing none of invalid equality assumption is pruned — from the target of the requirement. More specifically, what can cause a problematic case is the backward chaining with the logical formula in which a content literal in its body is the parent of a functional literal in its head. For example, the logical formula used in Figure 4.3 has a content literal $student(x_1)$ in its body and a functional literal $in(x_1, x_2)$ in its head. Since the parent of $in(x_1, x_2)$ is

student(x_1), this backward chaining can cause a problematic case and then will be excluded from the target of the requirement.

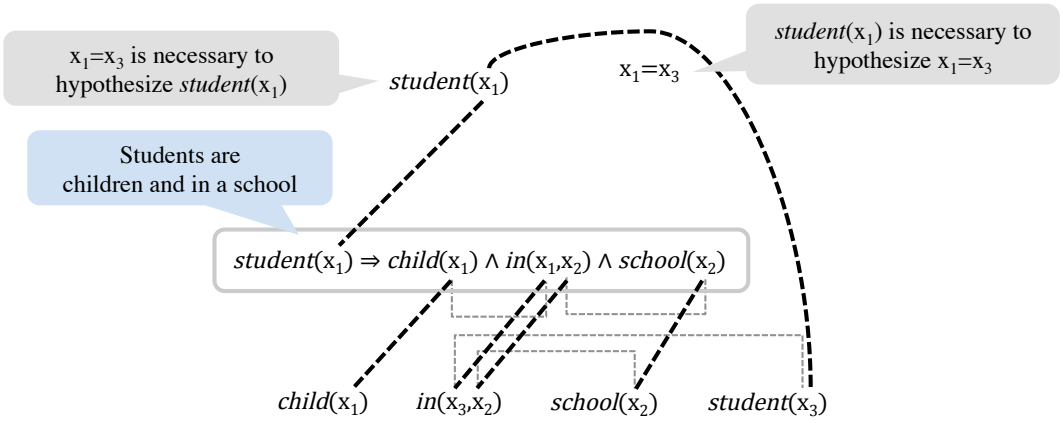

Figure 4.3: An example of backward-chaining which should not be pruned but can be pruned.

Let us descrive the above idea more formally. Given the potential elemental hypotheses P, in order to apply the backward chaining with an implicational rule $\bigwedge_{i=1}^{n} p_i(\mathbf{x}_i) \Rightarrow \bigwedge_{j=1}^{m} q_j(\mathbf{y}_j)$ to a conjunction $\bigwedge_{j=1}^{m} q_j(\mathbf{z}_j)$ in P, at least one of following conditions must be satisfied for each functional literal in the conjunction (we denote the index of target literal as f):

1. Supposing that the c-th literal in the head of the implicational rule (i.e., $q_c(\mathbf{y}_c)$) is a content literal and any of its terms (we denote $\mathbf{y}_c^i$) is identical to the first term of $q_f(\mathbf{y}_f)$, the variable pair $\mathbf{z}_c^i$ and $\mathbf{z}_f^1$ are identical or P contains the equality assumption $\mathbf{z}_c^i = \mathbf{z}_f^1$.

2. The first term of $q_f(\mathbf{y}_f)$ is included in $\bigwedge_{i=1}^{n} \mathbf{x}_i$.

If a backward chain operation cannot satisfy this condition, it introduces invalid equality assumptions and then can be excluded from the search space. In Section 4.2.3, we discuss the implementation of this requirement.

4.2.3 The extension of the potential elemental hypotheses generation

In Section 4.2.1 and Section 4.2.2, we proposed the methods to excludes the operations to introduce invalid equality assumptions from the search space. In this section, we discuss what algorithm to implement this requirement as.

Satisfiability of the requirement on each operation and the state of the potential elemental hypotheses depend on each other (i.e., satisfiability of the requirement is decided from the state of the potential elemental hypotheses, and the constituents of the potential elemental hypotheses depend on satisfiability of the requirement on each operation). Even if an operation cannot satisfy the requirement at certain point of time, other operations may enable it to satisfy the requirement after that. Therefore, satisfiability checking must be done so that all of operations not performed to the last are guaranteed to be unable to satisfy the requirement.

Based on the above idea, we extend the procedure of the potential elemental hypotheses generation in Section 2.2.1 as follows:

1. Initialize the potential elemental hypotheses P to O.

2. Enumerates the operations of backward chaining and unification which is applicable to P and perform them comprehensively with controling their order based on A*-based Abduction. However the operations not to satisfy the requirements of Section 4.2.1 and Section 4.2.2 are memorized in a buffer S instead of being performed.

3. Finish the potential elemental hypotheses generation iff the buffer S is empty.

4. Check satisfiability of the requirement on each operation in S and perform it iff it satisfies the requirement.

5. Finish the potential elemental hypotheses generation iff none of operations is performed in the previous step.

6. Go back to the second step.

Adopting this implementation, it is guaranteed that an operation not performed to the last cannot satisfy the requirement. As discussed in Section 4.2.1 and Section 4.2.2, an operation not to satisfy the requirement introduces invalid equality assumptions and then cannot be included in the original solution hypothesis. Consequently, it is guaranteed that these requirements preserve the solution hypotheses.

4.3 Improvement of A*-based Abduction

A*-based Abduction proposed in Chapter 3 has the problem that its computational efficiency get worse on applied to the meaning representation containing functional literals. In this section, in order to address this problem, we propose the method to change the way of estimation of the predicate distance and to improve the computational efficiency of A*-based Abduction.

4.3.1 Preliminary

In this section, we define some terms for the following discussion.

Given unifiable literals $\{l_1, l_2\}$ and supposing that l_1 explains an observable literal o_1 and that l_2 explains an observable literal o_2, what we call the **operation path** between o_1 and o_2 is the sequence of operations consisting of the unification for $\{l_1, l_2\}$ and the backward chaining operations needed to hypothesize l_1 or l_2 from o_1 or o_2. Here we call each of o_1 and o_2 the **anchor** of the operation path. For example, the operation path between $go(x_1, x_2)$ and $get(y_1, y_2)$ in Figure 1.1 consists of following three operations; (1) the backward chaining from $go(x_1, x_2)$ to $issue(x_2, u_1, x_1)$, (2) the backward chaining from $get(y_1, y_2)$ to $issue(u_2, y_2, y_1)$ and (3) the unification between $issue(x_2, u_1, x_1)$ and $issue(u_2, y_2, y_1)$. It should be noted here that the number of unification operations included an operation path must be just one [1]. What we call the **evidence** of an operation path is the set of observable literals which take part in the operation path (i.e., observable literals which the hypothesis by the operation path explains). An evidence of an operation path includes its anchors.

[1] This fact is necessary in the proof of safety of pruning the reachability graph

As noted Chapter 3, an evaluation function in A*-based Abduction must following conditions: (1) Since A*-based Abduction is based on ILP-formulated Abduction [Inoue and Inui, 2011, 2012], an evaluation function must be able to be represented as an ILP problem. (2) An evaluation function does not choice a redundant hypothesis as the solution hypothesis. In other words, it must be guaranteed that a literal not to contribute any unification is contained in the solution hypothesis).

Now, let us consider to extend the second condition to Neo-Davidsonian. For example, the hypothesis shown in Figure 4.4 (a) is semantically equal to one shown in Figure 4.4 (b) and then it is considered to be redundant and cannot be the best explanation. Consequently, an operation path whose evidence consists of functional literals sharing same parent and their parent, like one show in Figure 4.4 (a), can be pruned from the search space.

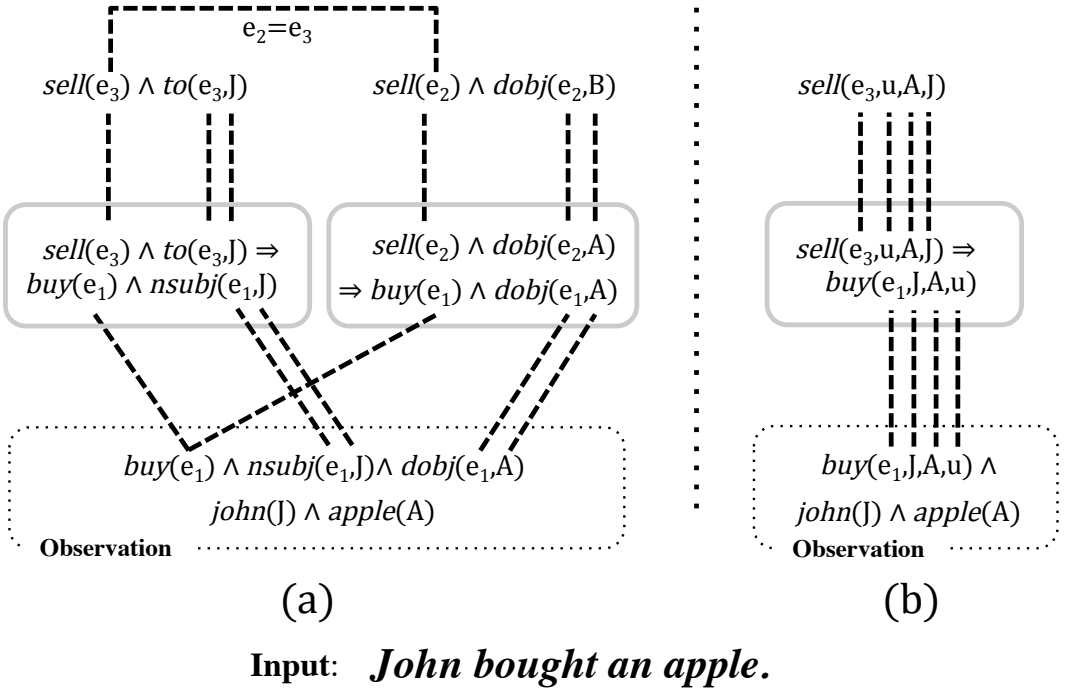

Figure 4.4: An example of redundant hypotheses in Neo-Davidsonian and Davidsonian.

In the following section, we suppose that an evaluation function can be ex-

pressed as an ILP problem equally and satisfy the following conditions:

Condition 4. An evaluation function does not choice a candidate hypothesis containing a literal not to contribute any unification as the solution hypothesis.

Condition 5. An evaluation function does not choice a candidate hypothesis containing an operation path whose evidence consists of functional literals sharing same parent and their parent as the solution hypothesis.

We call these Conditions **Simplicity Condition**.

4.3.2 Performance deterioration of A*-based Abduction

As noted in Chapter 3, a system of A*-based Abduction estimates the heuristic distance between predicates (Heuristic Estimated Distance, HED) and prunes the search space of elemental hypotheses with using HED. The problem here is that it makes the computational efficiency of A*-based Abduction significantly worse to use the meaning representation including the functional literals.

This inefficiency occurs because a functional literal behaves like a hub in HED. For example, let us consider the HED for the knowledge base used in Figure 4.1:

$$mistake(e_1) \wedge nsubj(e_1, x) \Rightarrow foolish(e_2) \wedge nsubj(e_2, x)$$
$$study(e_1) \wedge nsubj(e_1, x) \Rightarrow smart(e_2) \wedge nsubj(e_2, x)$$

The HEDs for this knowledge base can be expressed as the directed graph shown in Figure 4.5, where each heuristic distance between predicates corresponds to distance between the nodes of the predicates. Here $nsubj/2$ behaves like a hub and then all content predicate pairs are estimated to be reachable each other. However, in fact, some of them always introduce invalid equality assumptions and cannot be reachable in the solution hypothesis. For instance, although $foolish/1$ and $smart/1$ are estimated to be reachable but, as we see in Section 4.1.2, the operation path between $foolish/1$ and $smart/1$ introduce invalid equality assumptions and then cannot be included in the solution hypothesis.

Consequently, the HEDs cannot consider whether the inference between the predicates introduce invalid equality assumptions. This cause the computational inefficiency of A*-based Abduction.

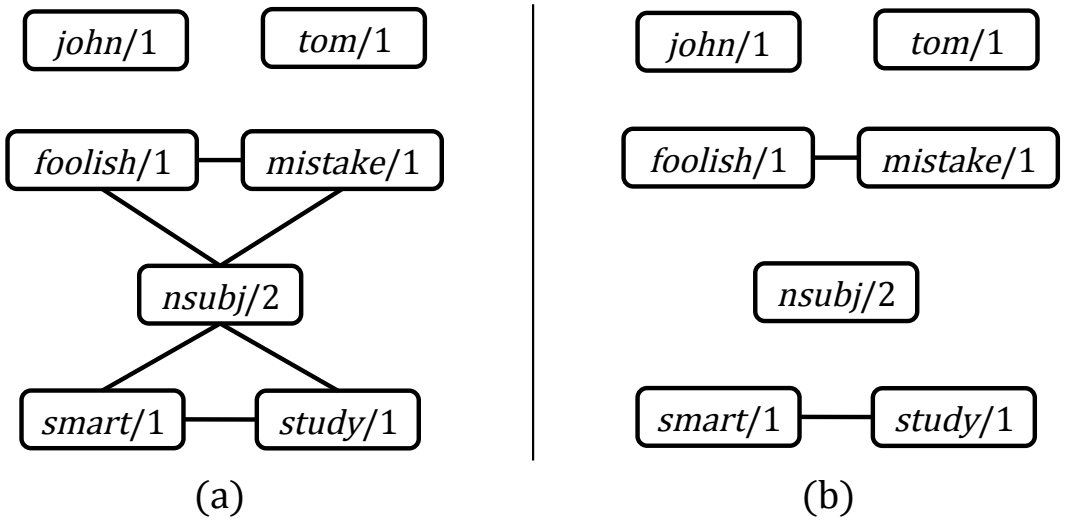

Figure 4.5: The HEDs of knowledge base in Figure 4.1 as a graph.

4.3.3 Pruning the heuristic estimated distance

In this section, to address the problem noted in Section 4.3.2, we propose to prune the connections from the HED. In other words, supposing that all of the possible inference between a pair of literals introduce invalid equality assumption (such as a pair of $foolish(e_2)$ and $smart(e_1)$ in Figure 4.1), we modify the HEDs so that the distance between the literals is infinity. As result, the inference to introduce invalid equality assumptions is excluded from the search space and then it is expected to improve the computational efficiency of A*-based Abduction.

We introduce the specific procedure. We add two extentions to A*-based Abduction as follows:

- Do not consider the distance to a functional literal iff its parent is included in the same side of implication rule. For example, supposing a logical formula $study(e_1) \wedge nsubj(e_1, x) \Rightarrow smart(e_2) \wedge nsubj(e_2, x)$, a system takes into account only the connection between *study* and *smart*, where literals of *nsubj* are ignored because their parent (i.e., $study(e_1)$ and $smart(e_2)$) accompany them.

- Given a functional literal l_f, its parent l_c and another literal l_x, use the

heuristic distance between l_c and l_x for the heuristic distance between l_f and l_x on the potential elemental hypothesis generation (i.e., $h^*(l_f, l_x) = h^*(l_c, l_x)$). For example, since the parent of $nsubj(e_1, j)$ is $smart(e_1)$ in Figure 4.1, a system use the heuristic distance between $smart(e_1)$ and $foolish(e_2)$ for the heuristic distance between $nsubj(e_1, j)$ and $foolish(e_2)$. If a functional literal has several parent, a system use the one of the minimum distance among them.

These extensions excludes the inference to introduce invalid equality assumptions from the search space. For example, the HEDs shown in Figure 4.5 (a) are modified by the extensions and result in the HEDs shown in Figure 4.5 (b). As result, the heuristic distance between *smart* and *foolish* is estimated as infinity and therefore the inference between $smart(e_1)$ and $foolish(e_2)$ in Figure 4.1, which introduce invalid equality assumptions, is pruned.

If only the Validity Condition and Simplicity Condition are satisfied, it is guaranteed that this method preserve the solution hypothesis — the solution hypothesis before applying this method is not pruned by this method. See the Appendix for the detail of the proof.

4.4 Experiments

4.4.1 Common Setting

Dataset For our experiments, we converted each of problems in the development dataset of Winograd Schema Challenge [Levesque, 2011] by Rahman and Ng [2012] into first-order logical formula and used them for the observations. Specifically, we parsed the problems with The Stanford CoreNLP[1] and converted words and dependencies in the sentences into literals. The observation set consisted of 1,305 observations and the average number of literals in each observation is 28. An example of observation is shown as O in Table 4.4.1, which is converted from a sentence "*Tony helped Jeff because the wanted to help.*" We have verified that each of observation satisfies the Validity Condition. We denote this observation set O_{wsc}.

[1] http://nlp.stanford.edu/software/corenlp.shtml

The Winograd Schema Challenge (WSC) is a test of machine intelligence proposed by Levesque [2011]. This is a set of Winograd Schemas, which is a pair of sentence that differ in only once or two words and that contain an ambiguity of correference relation to requires the use of world knowledge and reasoning for its resolution. The following shows an example:

1. *Tony helped Jeff because* ***he*** *wanted to help.*
2. *Tony helped Jeff because* ***he*** *needed help.*

Here, "he" in the former sentence refers "Tony" and "he" in the latter sentence refers "Jeff". These correference relations are easily disambiguated by the human reader, but are not solvable by typical NLP techniques such as selectional preference. In our experiments, we regarded an Windograd Schema as two problems of coreference resolution and converted it into two observation.

Knowledge base For our experiments, we extracted knowledge of causality from ClueWeb12[1] and used for knowledge base. Specifically, we firstly parsed the sentences in ClueWeb12 with the Stanford CoreNLP and extracted 50 millions of pair of events which have a shared argument. For instance, we parsed a sentence "*Tom helped Mary yesterday, so Mary thanked to Tom.*" and extracted ⟨*Tom help Mary yesterday*, *Mary thank to Tom*⟩, which share the argument "*Mary*". Next, we generalized the extracted event pairs using statistical criteria and converted them into first-order logical formulae. For specifically, we generalized each event pair at all possible abstraction levels (e.g. the example of event pair above is generalized into ⟨*Tom help Mary yesterday*, *Mary thank to Tom*⟩ を ⟨*Tom help X*, *X thank to Tom*⟩, ⟨*help X*, *X thank*⟩, and so on), counted their frequency and discarded generalized event pairs with low frequencies. Finally we converted the rest of them into first-order logical formulae and obtained 278,802 implicational logical formulae. We denote this set B_{ep}. B_{ep} shown in Table 4.4.1 is an example of the logical formula extracted from ClueWeb12.

Additionaly, we converted synonyms and hypernyms in WordNet [Fellbaum, 1998b] into logical formulae and obtained 235,706 implicational logical formulae.

[1] http://lemurproject.org/clueweb12/

Table 4.1: Examples of observation and knowledge base used in the experiment.

$$
\begin{aligned}
O = \quad & tony\_nn(E_1) \wedge help\_vb(E_2) \wedge jeff\_nn(E_3) \wedge because\_in(E_4) \wedge he\_pr(e5) \wedge \\
& want\_vb(E_6) \wedge to\_to(E_7) \wedge help\_vb(E_8) \wedge nsubj(E_2, E_1) \wedge \\
& dobj(E_2, E_3) \wedge mark(E_6, E_4) \wedge nsubj(E_6, e_5) \wedge advcl(E_2, E_6) \wedge \\
& aux(E_8, E_7) \wedge xcomp(E_6, E_8) \wedge nsubj(E_8, e_5) \\
B_{ep} = \quad & meet\_vb(e_1) \wedge nsubj(e_1, x) \\
& \Rightarrow have\_vb(e_2) \wedge nsubj(e_2, x) \wedge dobj(e_2, y) \wedge interest\_nn(y), \\
& graduate\_vb(e_1) \wedge nsubj(e_1, x) \\
& \Rightarrow give\_vb(e_2) \wedge iobj(e_2, x) \wedge dobj(e_2, y) \wedge job\_nn(y) \\
& get\_vb(e_1) \wedge nsubj(e_1, x) \wedge dobj(e_1, y) \wedge discount\_nn(y) \\
& \Rightarrow buy\_vb(e_2) \wedge nsubj(e_2, x), \ ... \\
B_{wn} = \quad & play\_nn(x) \Rightarrow action\_nn(x) \\
& attack\_vb(e) \Rightarrow affect\_vb(e)
\end{aligned}
$$

We denote this set B_{wn}. B_{wn} shown in Table 4.4.1 is an example of the logical formula obtained from WordNet.

Evaluation function We used the evaluation function which is based on Weighted Abduction [Hobbs et al., 1993] and satisfies the Validity Condition and the Simplicity Condition. Specifically, we added an constraint that the solution hypothesis must satisfy the Validity Condition and the Simplicity Condition to the legacy Weighted Abduction.

4.4.2 Comparison of computational efficiency

On this experiment, we compared the solving time when using our proposed methods and when A*-based Abduction proposed in Chapter 3. We used the following settings to compare:

BASELINE This setting uses A*-based Abduction for the method of the potential elemental hypotheses generation.

ALL This setting uses A*-based Abduction and the methods proposed in Section 4.2 and Section 4.3.

ABLATION-1 This setting uses A*-based Abduction and the method proposed in Section 4.3. The method in Section 4.2 is ablated.

ABLATION-2 This setting uses A*-based Abduction and the method proposed in Section 4.2. The method in Section 4.3 is ablated.

In this experiment, We used so small setting that BASELINE can find the optimal solution in the search space. Specifically, we used all observations in O_{wsc} and used 187,732 logical formulae, which are part of B_{ep}. We set $d_{max} = 1$ to limit the search spece of the elemental hypotheses set, where backward chaining operations can be applied only to observable literals. The timeout limit was set to 5 minutes.

The results of this experiment are shown in Figure 4.6, Figure 4.7 and Figure 4.8. Each plot point in the figures represents the solving time for the corresponding observation. The horizontal axis represents the solving time on the setting of ALL and the vertical axis represents the solving time on another setting to be compared with ALL. In these figures, 210 observations were discarded due to timeout in all settings. A green line represents $y = x$ and then a plot point over the line means that the proposed methods improved the solving time for the corresponding observation.

The result of comparison between ALL and BASELINE is shown in Figure 4.6. From Figure 4.6, we can observe that the proposed methods was much more efficient for all the observations than A*-based Abduction.

The result of the ablation test of the method in Section 4.2 is shown in Figure 4.7. From Figure 4.7, we can observe that the method in Section 4.2 imploved solving time for all the observations excluding several observations. The possible reasons why solving time for the several observations got worse are that it taken some time to check the requirement satisfaction and that invalid equality assumptions to be pruned are few or nothing.

The result of the ablation test of the method in Section 4.3 is shown in Figure 4.8. From Figure 4.8, we can observe this result is similar to the result in Figure 4.6. This is because the elemental hypotheses pruned by the method in Section 4.3 include ones pruned by the method in Section 4.2.

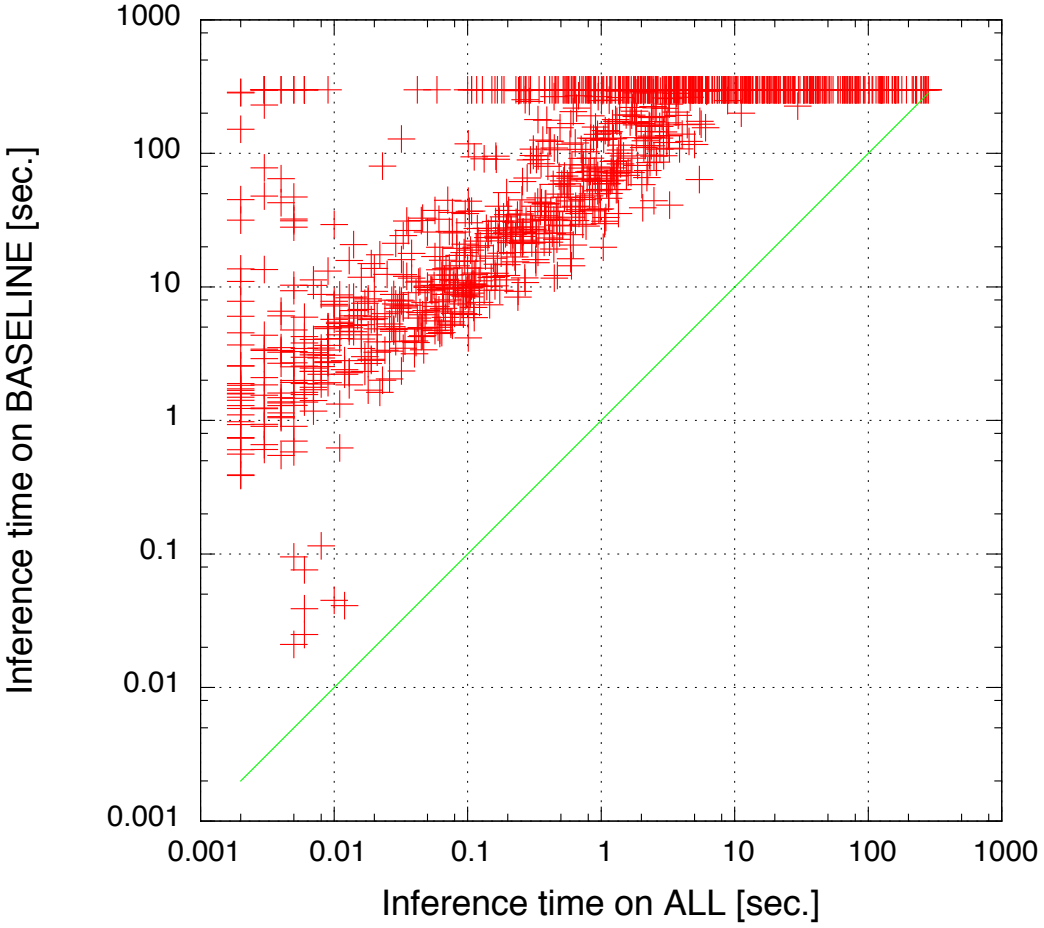

Figure 4.6: The comparison between ALL and BASELINE.

For all the problem, we observed that the value of evaluation function for the solution hypothesis does not change no matter what the method used for the potential elemental hypotheses generation is. From this, it was shown empirically that our method preserve optimality of the solution hypothesis.

4.4.3 Experiment on larger search space

In the experiment reported in the previous section, it was found that a system based on the proposed methods is much more efficient than A*-based Abduction. However, this experiment (SMALL) was strongly scaled down and is considered to be alienated from the task in real world. In this section, for reference, we

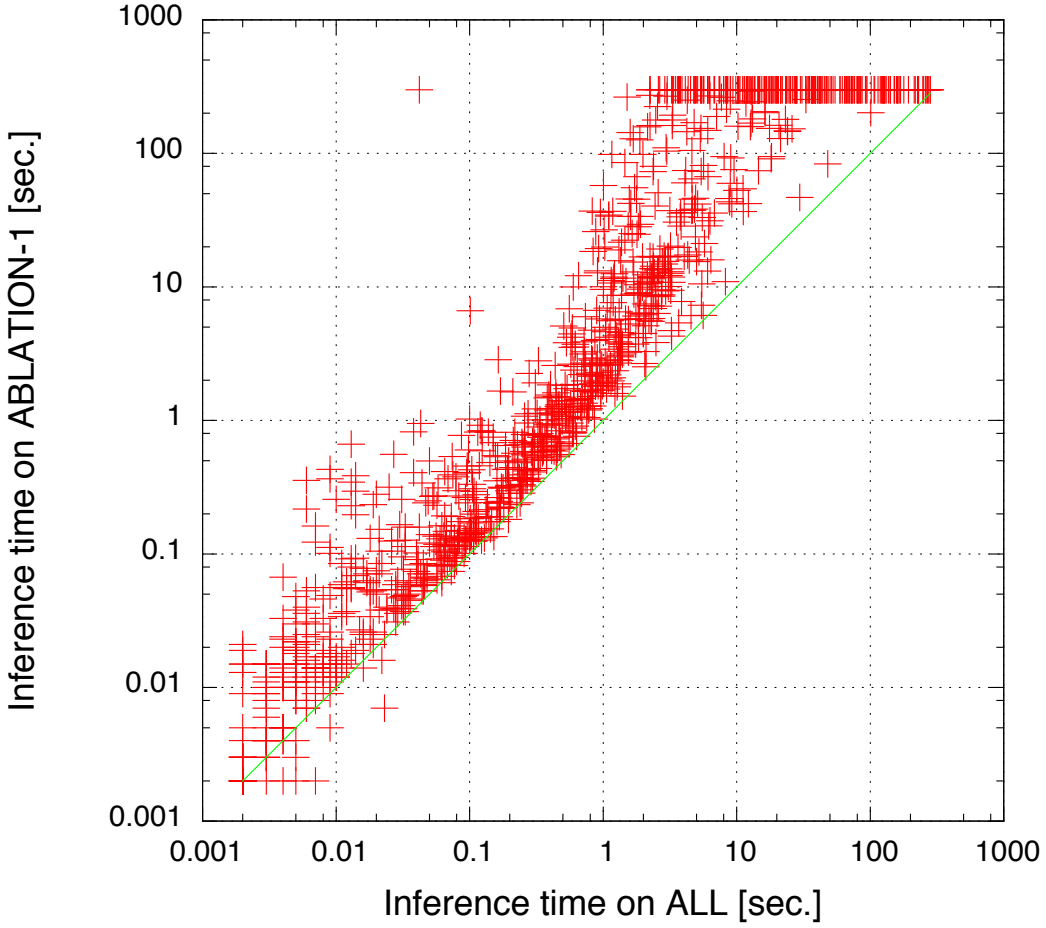

Figure 4.7: The comparison between ALL and ABLATION-1.

performed an experiment on the setting which conforms more to the task in real world. From this experiment, we show that explanations inferred by our system were semantically appropriate.

The dataset includes the problems which our knowledge base can never solve — a problem including negation or contradictory conjunctions, a problem which needs knowledge about specific proper noun to solve, a problem to deal with numerical expressions and so on. Therefore, we randomly extracted 100 problems from O_{wsc} and classified them based on the kind of knowledge needed to solve them. We used 32 out of 100 problems for observations, which is considered to be solved by only knowledge of causality relation. We used all the implicational

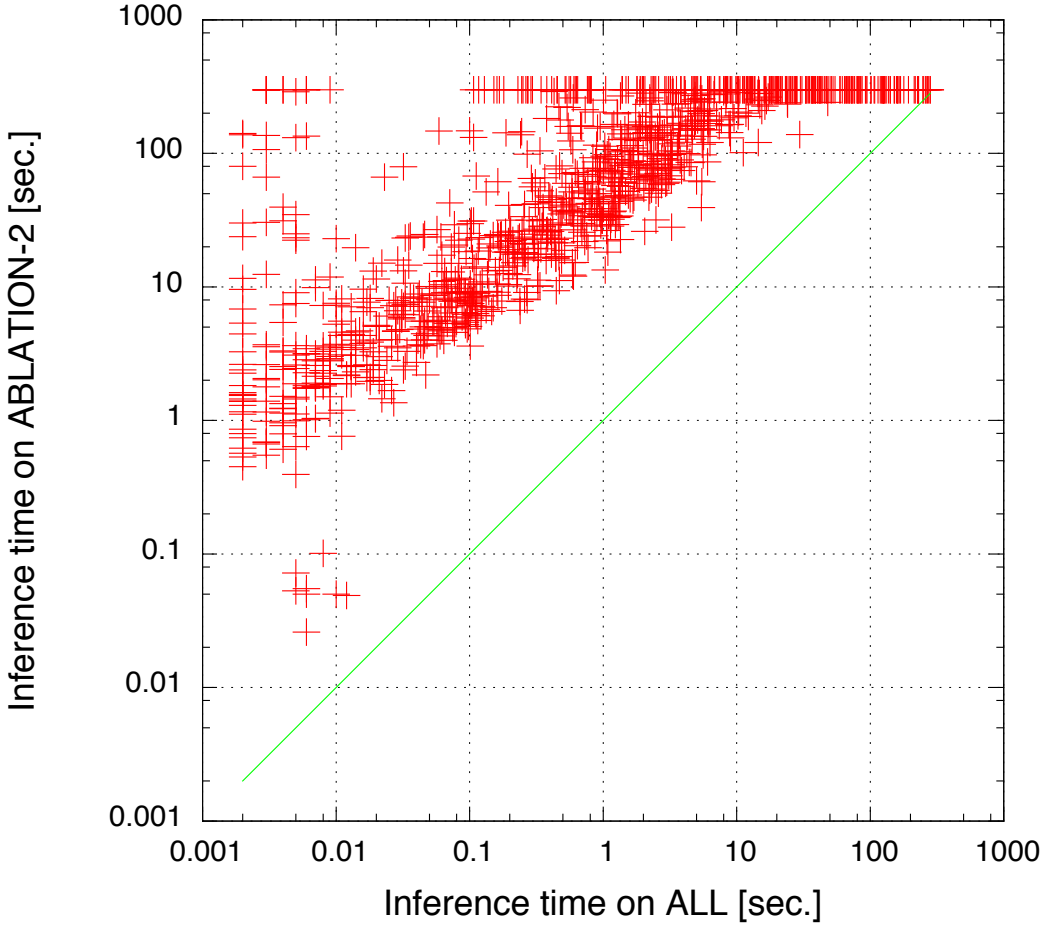

Figure 4.8: The comparison between ALL and ABLATION-2.

rules in B_{ep} and B_{wn} for background knowledge. This set consists of 514,508 rules. We set $d_{max} = 2$ from our empirical knowledge that an explanation using three or more causality relation tends to be semantically inappropriate. The timeout limit for the potential elemental hypothesis generation was set to 10 seconds. We used BASELINE and ALL noted in the previous section for the methods for the potential elemental hypotheses generation.

The result is shown in Table 4.4.3. From this result, we can observe that the baseline system (A*-based Abduction) could hardly find the solution due to the computational cost but a system based on the proposed methods could overcome that.

Table 4.2: The result of coreference resolution in WSC with an abduction-based system.

	BASELINE	ALL
Correct	1	15
Wrong	0	6
No Decision	31	11
Precision	-	0.71
Recall	0.03	0.46

From the resulting precision on the setting of ALL, we can observe that each explanation is semantically appropriate in its own way. On the other hand, the recall was very low and we need to improve the coverage of background knowledge in future. Although the resulting accuracy was not so good, abductive reasoning with real scale knowledge base became feasible, due to the proposed methods. This is considered to be important contribution for further research on inference-based discourse processing.

4.5 Conclusion

While abduction has long been considered to be a promising framework for discourse processing, its computational complexity has hindered the application of abduction to practical NLP problem.

In this chapter, we focused on the problem of computational cost caused by literals representing thematic roles and proposed a method that eliminates the redundant operations related with such literals from the search space. We then empirically showed that a system based on this method is far more efficient than A*-based Abduction.

In our future work, we will consider to prune the search space with using property of each dependency. Each relationship has various properties — vertical relationship is transitive, one event cannot have several arguments with a same thematic role, contiguity is symmetrical and so on. We expect these properties to be useful for pruning the search space of the potential elemental hypotheses generation. For example, given an observation $go(e_1) \wedge go(e_2) \wedge nsubj(e_1, x_1) \wedge$

$nsubj(e_2, x_2)$, we know that the hypothesis $e_1 = e_2 \wedge x_1 \neq x_2$ is not feasible because one *go* event cannot have two argument with nominal subject role. We expect to improve the efficiency of abduction by pruning such hypotheses from the search space.

In addition, we will consider to parallelize the potential elemental hypotheses generation and the optimization of the ILP problem. In existing implementation, these procedures are processed sequentially. Hence, before the optimization step, it is not clear how good the solution hypothesis in the current potential elemental hypotheses is. For this problem, we consider to generate the potential elemental hypotheses while seeking the solution hypothesis in the current potential elemental hypotheses and then stop the generation according to the result of the optimization.

Futhermore, we will focus on application abduction to real world problems. Our system made abduction with real scale knowledge base feasible. This enables to evaluate accuracy of abduction-based system for real world task and to compare that system with other existing frameworks empirically. We will consider to develop large and accurate knowledge base for abduction and to develop an evaluation function suitable for discourse processing.

Chapter 5

Discriminative Learning of Abduction

While the lack of world knowledge resources hampered applying abduction to real-life problems in the 1980s and 1990s, a number of techniques for acquiring world knowledge resources have been developed in the last decade [Chambers and Jurafsky, 2009; Fellbaum, 1998a; Hovy et al., 2011; Ruppenhofer et al., 2010; Schoenmackers et al., 2010, etc.]. In addition, the development of an efficient inference technique of abduction warrant the application of abduction with large knowledge bases to real-life problems [Inoue and Inui, 2011]. Consequently, several researchers have started applying abduction to real-life problems exploiting large knowledge bases. For instance, inspired by Hobbs et al. [1993], Ovchinnikova et al. [2011] propose an abduction-based natural language processing framework using forty thousands axioms extracted from the popular ontological resources, WordNet [Fellbaum, 1998a] and FrameNet [Ruppenhofer et al., 2010]. They evaluate their approach on the real-life natural language processing task of textual entailment recognition [Dagan et al., 2010].

Although discourse processing with abductive reasoning has been studied from the 1980s, less attention has been paid to how to automatically learn evaluation functions. To apply abductive inference to a wide range of tasks, this non-trivial issue needs to be addressed because the criterion of plausibility is highly task-dependent. A notable exception is a series of studies [Blythe et al.,

2011; Kate and Mooney, 2009; Singla and Domingos, 2011], which emulate abduction in the probabilistic deductive inference framework, Markov Logic Networks (MLNs) [Richardson and Domingos, 2006]. MLN-based approaches can exploit several choices of weight learning methods originally developed for MLNs [Huynh and Mooney, 2009; Lowd and Domingos, 2007, etc.]. However, MLN-based abduction has severe problems when they are applied to discource processing which we will discuss in Section 5.3.

In this chapter, we propose a novel supervised approach for learning the evaluation function of first-order logic-based abduction. This is a framework to learn the evaluation function from subsets of explanations (henceforth, we call it *partial abductive explanations*). More specifically, we assume that we apply abduction to a specific task, where a subset of the best explanation is associated with output labels, and the rest are regarded as hidden variables. We then formulate the learning problem as the task of discriminative structured learning with hidden variables. As the evaluation function, we use the parametrized non-linear evaluation function proposed by Hobbs et al. [1993].

5.1 Discriminative Learning for Weighted Abduction

In this section, we propose a method to learn the parameters of evaluation function in Weighted Abduction by recasting the parameter estimation problem as an online discriminative leaning problem with hidden variables.

The idea is four-fold:

1. We train the evaluation function with only partially specified gold abductive explanations which we represent as a partial set of the required literals (*gold partial explanations*).

2. We automatically infer complete correct abductive explanations from gold partial explanations by abductive inference.

3. We optimize the parameters of the evaluation function by minimizing the loss function where the loss is given by the difference of the costs of the

minimal-cost hypothesis and the complete correct abductive explanations.

4. We employ feed-forward neural networks to calculate the gradient of each parameter.

In the rest of this section, we first formalize explanation in Weighted Abduction with directed acyclic graphs (Section 5.1.1), and we then describe the outline of our learning method (Section 5.1.2) and elaborate on our learning framework in the simple case where complete abductive explanations are given (Section 5.1.3). We then describe a method for learning the parameters from partial abductive explanations (Section 5.1.4). Finally, we describe how to update the parameters through error back-propagation in FFNNs (Section 5.1.5).

5.1.1 Preliminaries

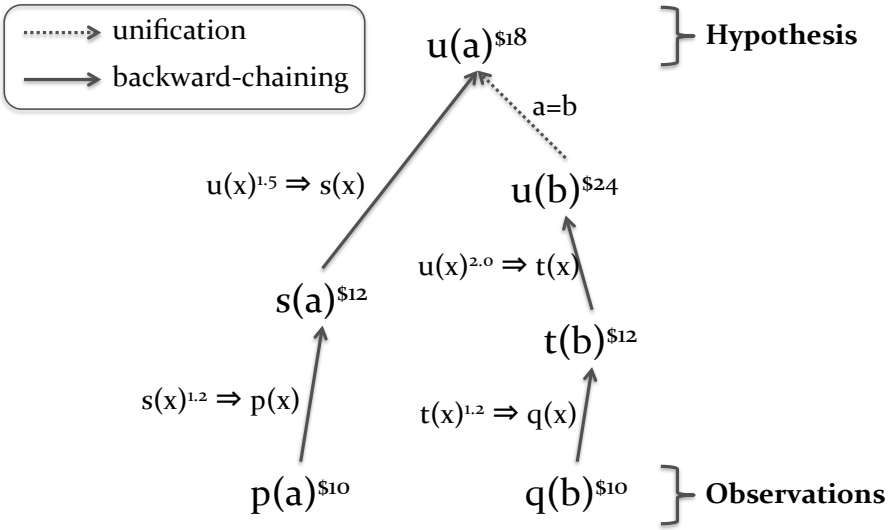

Figure 5.1: An example proof tree in DAG

In this chapter, we express the hypotheses of Weighted Abduction as directed acyclic graphs (DAG). Namely, we regard each literal in the hypothesis as a node of DAG and each of relation between literals as an edge of DAG. We call these graphs *proof graph* and use a notation $G_{O,B,H}$ to denote the proof graph made from the observation O, the background knowledge B and the hypothesis H.

We define following two types of edge in proof graphs:

- **Backward-chaining:** Given the tail node's literal $p(x)^{\$c_1}$ and the head node's literal $q(x)^{\$c_2}$, this relation indicates that $q(x) \cup B \models p(x)$. Namely, $q(x)$ is hypothesized with $p(x)$. Then, the cost of head node's literal is caluclated by multiplication of the cost of tail node's literal and the weight of background knowledge (e.g. $c_2 = c_1 w$, where $q(x)^w \Rightarrow p(x)$).
- **Unification:** Given the tail node's literal $p(x)$ and the head node's literal $p(y)$, this relation indicate that $p(x)$ and $p(y)$ are unified and $x = y$.

Between the tail node and the head node of each edge in a proof graph, the relation that the head node's literal explain the tail node's literal exists. Thus, the set of literals of leaf nodes in proof graphs corresponds to P_H and the set of literals of root nodes in proof graphs corresponds to O.

We show an example proof graph in Figure 5.1. This is the proof graph made from the following background knowledge, observation and hypothesis:

$$
\begin{aligned}
B = \{ \quad & \forall x \quad (s(x)^{1.2} \Rightarrow p(x)), \forall x (s(x)^{1.2} \Rightarrow q(x)), \\
& \forall x \quad (u(x)^{1.5} \Rightarrow s(x)), \forall x (u(x)^{2.0} \Rightarrow t(x))\}, && (5.1) \\
O = \quad & \exists x \quad (p(a)^{\$10} \wedge q(b)^{\$10}) && (5.2) \\
H = \quad & \exists x \quad (u(a)^{\$18} \wedge u(b)^{\$24} \wedge s(a)^{\$12} \wedge t(b)^{\$12} \wedge a = b) && (5.3)
\end{aligned}
$$

The cost of a hypothesis is calculated with Equation 2.2. Therefore, the cost of this hypothesis is calculated as $c(H) = \sum_{h \in P_H} c(h) = \$18$.

5.1.2 Outline of our method

In this section, we describe the outline of our learning method. The overall framework is illustrated in Figure 5.2.

First, we assume each training example to be a pair (O_i, τ_i), where O_i is an observation and τ_i is a gold partial explanation. A gold partial explanation is a set of literals that must be included in the correct abductive explanation T_i for the input observation O_i, i.e. $T_i \cup B \models O_i$ and $\tau_i \subseteq T_i$.

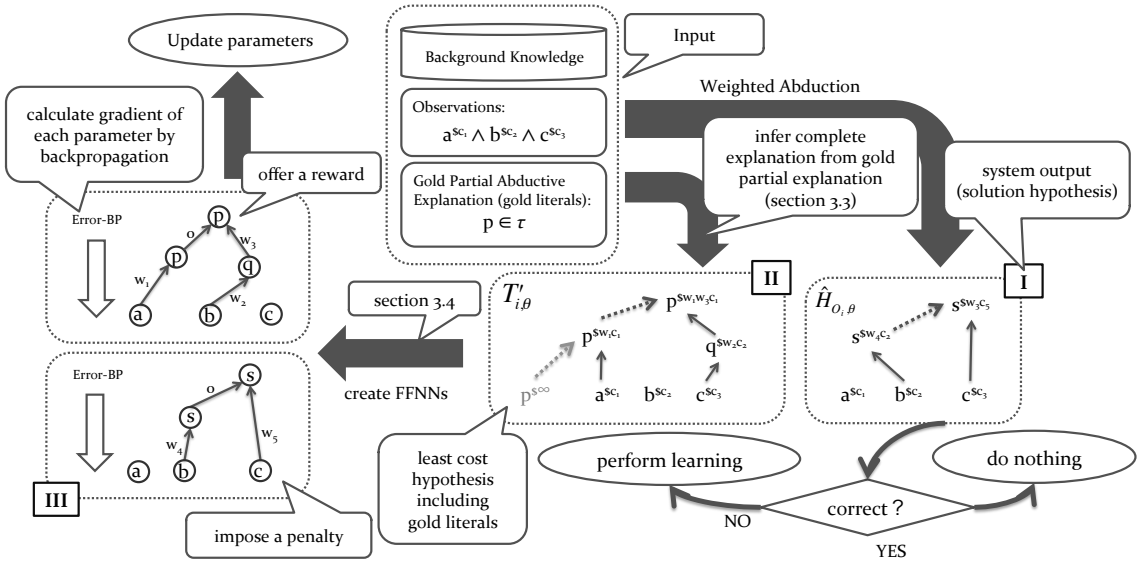

Figure 5.2: Outline of proposed parameter learning method

Next, we consider the online version of parameter learning [1]. For each cycle, given (O_i, τ_i), we perform Weighted Abduction for the observations O_i and background knowledge B with parameters $\mathbf{w}$, and get the solution hypothesis $\hat{H}_{O_i,\mathbf{w}}$ ([I] in Figure 5.2). If $\hat{H}_{O_i,\mathbf{w}}$ does not include τ_i (i.e. $\hat{H}_{O_i,\mathbf{w}}$ is an incorrect prediction), we update the parameters so that $\hat{H}_{O_i,\mathbf{w}}$ includes τ_i. In order to do so, we first infer a complete abductive explanation $\acute{T}_{i,\mathbf{w}}$ from the gold partial explanation τ_i ([II]). We then update parameters $\mathbf{w}$ by imposing a penalty to the wrong solution hypothesis $\hat{H}_{O_i,\mathbf{w}}$ and offering a reward to the inferred correct complete abductive explanation $\acute{T}_{i,\mathbf{w}}$. To compute these updates, we translate $\hat{H}_{O_i,\mathbf{w}}$ and $\acute{T}_{i,\mathbf{w}}$ to feed-forward neural networks and perform backpropagation on them ([III]).

In this chapter, we assume that there is enough knowledge to infer the correct explanation in each problem (we call this *the knowledge completeness assumption*). If this assumption were not satisfied, which means that the correct explanation is not included in the candidate hypotheses, then we could not infer the

[1]The batch version can also be considered by accumulating the gradients for each cycle before updating the weights.

correct explanation irrespectively of parameters. In the following discussion, we do not consider a case of knowledge base shortage.

5.1.3 Learning from complete abductive explanations

Let us first assume that we have a set of training examples labeled with a complete abductive explanation. Namely, we consider a training dataset $D = \{(O_1, T_1), (O_2, T_2), ..., (O_n, T_n)\}$, where O_i is an observation and T_i is the gold (correct) complete abductive explanation for O_i, i.e. $T_i \cup B \models O_i$.

For each labeled example (O_i, T_i), the solution hypothesis $\hat{H}_i$ is obtained by:

$$\hat{H}_i = \arg\min_{H \in \mathcal{H}_i} c_{\mathbf{w}}(H) \tag{5.4}$$

We consider that a solution hypothesis $\hat{H}_i$ is correct if $\hat{H}_i = T_i$. Now we consider a loss function that calculates how far the current solution hypothesis is from the gold explanation, analogously to standard learning algorithms. If the current solution hypothesis is correct, the loss is zero. If $\hat{H}_i \neq T_i$, on the other hand, we consider the loss as given by the following loss function:

$$E(O_i, \mathbf{w}, T_i) = \begin{cases} \frac{1}{2}\left(\frac{c_{\mathbf{w}}(T_i) - c_{\mathbf{w}}(\hat{H}_i)}{c_{\mathbf{w}}(T_i) + c_{\mathbf{w}}(\hat{H}_i)} + m\right)^2 + \lambda \mathbf{w} \cdot \mathbf{w} & (\hat{H}_i \neq T_i) \\ 0 & (\hat{H}_i = T_i) \end{cases}, \tag{5.5}$$

where $\lambda \mathbf{w} \cdot \mathbf{w}$ is a regularization term and m is a margin. Our goal is to learn the evaluation function $c_{\mathbf{w}}$ that has minimal prediction errors. This goal is accomplished by learning parameters $\mathbf{w}^*$ which minimize the total loss as below:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{(O,T) \in D} E(O, \mathbf{w}, T) \tag{5.6}$$

We describe how to minimize the loss in Section 5.1.5.

Note that we use the ratio of evaluation functions $\frac{1}{2}\left(\frac{c_{\mathbf{w}}(T_i) - c_{\mathbf{w}}(\hat{H}_i)}{c_{\mathbf{w}}(T_i) + c_{\mathbf{w}}(\hat{H}_i)} + m\right)^2$ as the loss function, instead of $\frac{1}{2}\left(c_{\mathbf{w}}(T_i) - c_{\mathbf{w}}(\hat{H}_i)\right)^2$. In the following, we shortly justify the use of the ratio of evaluation function. Let us suppose that we employ $\frac{1}{2}\left(c_{\mathbf{w}}(T_i) - c_{\mathbf{w}}(\hat{H}_i)\right)^2$ as the loss function. Then, we can minimize the loss

function by minimizing the weight terms that appear in both $c_{\mathbf{w}}(T_i)$ and $c_{\mathbf{w}}(H_i)$, namely the weights assigned to axioms that are used in both T_i and H_i. For instance, given $O_i = \{p(a)^{\$c}\}$, $B = \{q(x)^{w_0} \Rightarrow p(x), s(x)^{w_1} \Rightarrow q(x), t(x)^{w_2} \Rightarrow q(x)\}$, $T_i = \{s(a)^{\$w_0 w_1 c}\}$, $H_i = \{t(a)^{\$w_0 w_2 c}\}$, we can minimize the value of loss function by minimizing the value of w_0. As a result, the learning procedure just decreases w_0 as much as possible to minimize the loss function. This prevents our framework from learning a meaningful evaluation function, because the minimization of weights does not imply that we can infer the gold hypothesis as the solution hypothesis. To avoid this problem, we employ the ratio of evaluation functions.

5.1.4 Learning from partial abductive explanations

In the above, we assumed that each training example has a complete abductive explanation. However, this assumption is not realistic in many cases because it is usually prohibitively costly for human annotators to give a complete abductive explanation for each given input. This leads us to consider representing a training example as a pair of observation O_i and gold partial explanation τ_i, which is a partial set of literals that must be included in the explanation of O_i. In the case of Figure 5.2, we assumed that the correct hypothesis for the given observation is partially specified by the literal $p \in \tau$.

This way of simplification is essential in real-life tasks. In plan recognition, for example, it is not an easy job for human annotators to give a complete explanation to an input sequence of observed events, but they can tell whether it is a shopping story or a robbing story much more easily, which can be indicated by a small set of gold literals.

Now, our goal is to learn the evaluation function from partial explanations $D = \{(O_1, \tau_1), (O_2, \tau_2), ..., (O_n, \tau_n)\}$. Regarding whether each gold literal is included in the solution hypothesis $\hat{H}_{O_i,\mathbf{w}}$ and the structure of the proof graph $G_{O_i,B,\hat{H}_{O_i,\mathbf{w}}}$ as hidden states, this task can be seen as discriminative structure learning with hidden states. The issue is how to infer the complete correct explanation $\acute{T}_{i,\mathbf{w}}$ from a given incomplete set τ_i of gold literals. Fortunately, this can

be done straightforwardly by adding the gold literals τ_i to the observation O_i:

$$O_i^+ = O_i \cup \{t^{\$\infty} \mid t \in \tau_i\}, \tag{5.7}$$

where each gold literal is assigned an infinitive cost. Then, the solution hypothesis $\hat{H}_{O_i^+,\mathbf{w}}$ is equivalent to the complete correct explanation $\acute{T}_{i,\mathbf{w}}$ if the following conditions are satisfied:

- A hypothesis including τ_i exists in the candidate hypotheses for O_i (the knowledge completeness assumption).
- $\hat{H}_{O_i^+,\mathbf{w}}$ has no backward chaining from $t \in \tau_i$.

Figure 5.2 ([II]) illustrates a simple case, where $\hat{H}_{O_i^+,\mathbf{w}}$ is inferred by adding the gold literal p to the observation. Since this added literal p is assigned an infinitive cost, it is strongly motivated to derive an explanation including that p, resulting in obtaining the correct explanation $\acute{T}_{i,\mathbf{w}}$.

When these conditions are satisfied, because each t has a huge cost, the system selects as the solution hypothesis $\hat{H}_{O_i^+,\mathbf{w}}$ the hypothesis in which most literals in τ_i unify with other literals. Then, assuming the existence of a hypothesis including τ_i in the candidate hypotheses for O_i, there is the hypothesis in which each of the literals in τ_i unifies to a literal in the candidate hypotheses for O_i^+, and it is selected as solution hypothesis $\hat{H}_{O_i^+,\mathbf{w}}$. Because the cost of t must be 0 when it is unified with an other literal included in O_i or hypothesized from O_i, the cost of $\hat{H}_{O_i^+,\mathbf{w}}$ is equal to cost of $\acute{T}_{i,\mathbf{w}}$. So $\hat{H}_{O_i^+,\mathbf{w}}$ must be equal to $\acute{T}_{i,\mathbf{w}}$.

It should be note that we can check whether candidate hypotheses satisfy the above-mentioned conditions by checking the cost of the solution hypothesis, because any non-unified $t^{\$}\infty$ will result in a huge cost.

5.1.5 Updating parameters with FFNNs

To update parameters, we want to compute the gradient of the loss function for each parameter. However, since the evaluation function and the loss function are both nonlinear to their parameters, their gradients cannot be computed straightforwardly.

To solve this problem, we propose employing feed-forward neural networks (FFNNs). An FFNN is a directed acyclic graph where the output of each node j is given by:

$$z_j = h(a_j), \tag{5.8}$$

$$a_j = \sum_{i \in \{i | e_{i \to j} \in \mathbb{E}\}} z_i \times \mathbf{w}_{i \to j}, \tag{5.9}$$

where z_i denotes the output of node i, a_i denotes the degree of activation of node i, $h(a)$ is an activation function, $e_{i \to j}$ denotes a directed edge from node i to node j, and $\mathbf{w}_{i \to j}$ denotes the weight of $e_{i \to j}$.

Then, we express the evaluation function of H with a FFNN. This is achieved by applying the following convertion to $G_{O,B,H}$:

1. The cost of each literal in $G_{O,B,H}$ is the output of the node in the corresponding FFNN.

2. Each backward-chaining edge in $G_{O,B,H}$ is an edge with weight w in the FFNN where w denote the weight of the background knowledge of the corresponding backward-chaining edge.

3. Each unification edge in $G_{O,B,H}$ is an edge with weight 0 in the FFNN.

4. The activation function of each layer in FFNNs is $h(a) = a$.

5. An *output node*) is added to the FFNN, making new edges with weight 1 between output node and each node that corresponds to each literal in P_H (i.e. leef nodes in the proof graph).

Then, the value of the output node is equal to the evaluation function $c_{\mathbf{w}}(H)$ in Weighted Abduction.

We show that the evaluation function of Weighted Abduction is converted into equivalent FFNNs as shown in Figure 5.3. This indicates the FFNN can express the evaluation function of Weighted Abduction. Therefore, we are able to apply various techniques in FFNNs to learning parameters of Weighted Abduction. Namely, gradients of the loss function can be caluclated easily by using the backpropagation technique of FFNNs.

Algorithm 4 parameter learning

1: **Input:** $B, \mathbf{w}, \mathbb{D}$
2: **repeat**
3: **for all** $(O, \tau) \in \mathbb{D}$ **do**
4: $\hat{H} \leftarrow Inference(O, \mathbf{w})$
5: **if** $\tau \not\subseteq \hat{H}$ **then**
6: $H^- \leftarrow \hat{H}$
7: $O^+ = O \cup \{t^{\$\infty} \mid t \in \tau\}$
8: $H^+ \leftarrow Inference(O^+, \mathbf{w})$
9: $E_{O,\mathbf{w}} \leftarrow LossFunction(H^+, H^-)$
10: $N \leftarrow MakeFFNN(H^+, H^-)$
11: **for all** $h \in P_{H^+} \cup P_{H^-}$ **do**
12: **if** $c(h) > 0$ **then**
13: assign gradient $\frac{\partial E_{O,\mathbf{w}}}{\partial c(h)}$
14: **end if**
15: **end for**
16: do backpropagation
17: $\mathbf{w} \leftarrow UpdateWeights(\mathbf{w}, \nabla E_{O,\mathbf{w}})$
18: **end if**
19: **end for**
20: **until** convergence
21: **Output: w**

Moreover, FFNNs are flexible framework and can express various functions by changing the activation functions or the network's structure. Thus, this idea can be apply to not only Weighted Abduction but other various frameworks of abduction.

5.1.6 Procedures of parameter learning

The overall learning procedure is given in Algorithm 1. First, the solution hypothesis is inferred from observation O, and if it does not include gold literals τ, it is treated as a negative example H^- (Line 3-6). Next, the positive example H^+ is inferred from observation O^+ (Line 7,8). The loss is then calculated from the costs of H^+ and H^- (Line 9) H^+ and H^- is converted into FFNNs (Line 10). The gradient of the loss function for each non-zero cost literal is assigned to the corresponding node in the FFNN (Line 11-15). The gradients of the

loss function for costs of the other literals are calculated by applying standard backpropagation to the converted FFNNs (Line 16). Updating the parameters is performed with these gradients (Line 17). The parameters are trained iteratively until the learning converges.

5.1.7 Featurizing Parameters

So far, we have assigned a parameter for each literal which corresponds to a particular background knowledge. However, in this setting, we can train parameters for the background knowledge only appear in the training data, therefore the trained system would not be able to deal with unseen data. In this section, we describe a method which featurizes parameters of weighted abduction and its learning algorithm.

We introduce a function which defines parameter values:

$$\theta_i = h(\mathbf{F}_i \cdot \phi) \tag{5.10}$$

where $\mathbf{F}_i \in \mathbb{R}^n$ is the feature vector for a parameter θ_i which corresponds to a particular background knowledge, $\phi \in \mathbb{R}^n$ is the weights for features and $h(\cdot)$ is the activation function. In training, instead of parameters, feature weights ϕ are trained based on a particular loss function. In Weighted Abduction, since parameter values must be more than 1.0, we employ the following activation function:

$$h(a) = \frac{1.0}{n} + exp(a) \tag{5.11}$$

where n is the number of literals in the left hand side of the background knowledge which the literal belongs to. Gradient values of feature weights can be calculated from gradient values of parameters as follows:

$$\frac{\partial E_{O,\mathbf{w}}}{\partial \phi_k} = \sum_i \left(\frac{\partial E_{O,\mathbf{w}}}{\partial \mathbf{w}_i} \mathbf{F}_{ik} \right) \tag{5.12}$$

where ϕ_k is k^{th} element of the feature weights, $\mathbf{F}_{ik}$ is k^{th} element of the feature vector for a parameter $\mathbf{w}_i$.

5.2 Evaluation

5.2.1 Evaluation for ablity to learn parameters

We evaluate the proposed learning procedure on the dataset of plan recognition. In this experiment, we address the following questions: (i) does our leaning procedure actually decrease prediction errors? (ii) are models trained by our learning procedure robust to unseen data? To answer these questions, we evaluate prediction performance on a plan recognition dataset in the two settings: a closed test (i.e., the same dataset is used for both training and testing) and an open test (i.e., two distinct datasets are used for training and testing). In order to obtain the lowest-cost hypotheses, we used the Integer Linear Programming-based abductive reasoner proposed by Inoue and Inui [2011].

5.2.1.1 Dataset

We used Ng and Mooney [1992]'s story understanding dataset, which is widely used for evaluation of abductive plan recognition systems [Kate and Mooney, 2009; Raghavan and Mooney, 2010; Singla and Domingos, 2011]. In this dataset, we need to abductively infer the top-level plans of characters from actions which are represented by the logical forms. For example, given "*Bill went to the liquor-store. He pointed a gun at the owner*," plan recognition systems need to infer *Bill*'s plan. The dataset consists of development set and a test set, each of which includes 25 plan recognition problems. The dataset contains on average 12.6 literals in observed logical forms. The background knowledge base contains of 107 Horn clauses. Figure 5.4 shows an example of this dataset.

In our evaluation, we introduced two types of axioms in addition to the original 107 axioms. First, to make the predicates representing top-level plans (e.g. shopping, robbing) disjoint, we generated 73 disjointness axioms (e.g. $robbing(x) \Rightarrow \neg shopping(x)$). Note that it is still possible to infer multiple top-level plans for one problem, because we are able to hypothesize $robbing(x) \wedge shopping(y)$. Second, we generated axioms of superplan-subplans relations (e.g. $going\_by\_plane(x) \Rightarrow going\_by\_vehicle(x)$). In total, we used 220 background axioms for our evaluation.

For evaluating the prediction performance of our system, we focused on how well the system infers top-level plans, and their subparts (i.e. subplans, role-fillers), following Singla and Domingos [2011]. More specifically, we use precision (ratio of inferred literals that are correct), recall (ratio of correct literals that are inferred by the system), and F-measure (harmonic mean of precision and recall), because the gold data often has multiple top-level plan predicates.

5.2.1.2 Experimental setting

We applied weight regularization in order to prevent overfitting to the training set. The hyperparameter for regularization λ was set to 0.1. For parameter updating, we employed the annealing approach; $\mathbf{w}_{new} = \mathbf{w} - \eta_0 k^i \nabla E_{\mathbf{w}}$ where η_0 (initial learning rate) was set to 0.0001, k (annealing parameter) was set to 0.95 and i is the number of iterations. The hyperparameters were selected based on performances on the development set. All weights were initialized to 0.0.

5.2.1.3 Results and discussion

At first, we report results of the closed test where the development set was used for both training and testing. Figure 5.5 shows the values of the loss function at each iteration on the development set. The curve indicates that our learning procedure successfully reduces values of the loss function at each iteration. The reason for the fluctuation in values is thought to be the existence of hidden variables.

In the open test, we trained our model on the development set and then tested on the test set. Figure 5.6 shows plots of values of the three measures (i.e. Precision, Recall and F-measure) on the test set at each iteration. Although the values are also fluctuate as with the closed test, performance rises in terms of all measures compared to the performances at iteration zero (i.e. initial values). The results suggest that the learning procedure is robust to unseen data.

Singla and Mooney [Singla and Domingos, 2011] report that the MLN-based approach achieve 72.10 F-measure on the same test set, which is slightly better than our results. However, our experimental setting and Singla's are different on various point such as framework of abduction (i.e. Weighted Abduction vs. MLN-based abduction), method of parameter learning (i.e. FFNNs vs. MLNs),

Feature	Explanation	Example feature value
LITERAL	each literal included in the left-hand side	$go\_step(s, g)$, $inst\_shopping(s)$
PRED	each predicate included in the left-hand side	go_step, $inst\_shopping$
CLAUSE	whole clause of axiom	$inst\_shopping(s) \wedge go\_step(s, g)$ $\rightarrow inst\_going(g)$
CLAUSE-OBS	combination of CLAUSE and each predicate in observations	$inst\_shopping(s) \wedge go\_step(s, g)$ $\rightarrow inst\_going(g)$ & $inst\_robbing$
PRED-OBS	combination of PRED and each predicate in observations	go_step & $inst\_robbing$

Table 5.1: Features used for the system. We show example feature values for the axioms $inst\_shopping(s) \wedge go\_step(s, g) \rightarrow inst\_going(g)$, and the observation $inst\_robbing(R)$.

Feature	Setting I	Setting II	Setting III
LITERAL	✓	✓	✓
PRED		✓	✓
CLAUSE		✓	✓
CLAUSE-OBS			✓
PRED-OBS			✓

Table 5.2: Settings of features used in the experiment.

method of parameter initialization (i.e. constant value vs. manually tuning). Therefore, it is unable to compare usefulness of these frameworks.

It has taken about half an hour to perform training for each iteration. Most of the time was spent in obtaining solution hypotheses using ILP-based abductive inference.

5.2.2 Evaluation for featurizing

We evaluate how effective is featurizing parameters in Weighted Abduction. In this experiment, we evaluate prediction performance on a plan recognition dataset in 10-fold cross validation on some feature settings. The abductive reasoner and the hyperparameters are same as in the evaluation in Section 5.2.1.

5.2.2.1 Features

We evaluated three settings on featurizing the paramters. Table 5.1 shows the details about the features used in the experiment and Table 5.2 shows three settings we used in the experiment. LITERAL and PRED features include information of individual literals and predicates included in axioms. CLAUSE features capture what axioms are used in a hypothesis. CLAUSE-OBS and PRED-OBS features combine predicate or clause with observation information. The reason for introducing CLAUSE-OBS and PRED-OBS is to capture dependence between parameter weights of Weighted Abduction and observations. Setting I corresponds original Weighted Abduction. Setting II is more generalized than Setting I. In addition, Setting III considers dependency between parameter weights of Weighted Abduction and observations.

5.2.2.2 Results and discussion

Figure 5.7 shows the results of the experiment. The result indicates that our extention about the parameters successfully improves robustness to unseen data and correct assignment of the weight of axioms depend on the observations.

5.3 Related Work

As mentioned in Section 1, abduction has been extensively studied in a wide range of contexts. However, less attention has been paid to how to automatically learn evaluation functions. In the field of Statistical Relational Learning, some researchers [Blythe et al., 2011; Kate and Mooney, 2009; Singla and Domingos, 2011, etc.] employ Markov Logic Networks [Richardson and Domingos, 2006] to emulate abductive inference. MLNs provide well-studied software packages of inference and learning.

However, MLN-based approaches require special procedures to convert abduction problems into deduction problems because of the deductive nature of MLNs. The pioneering work of MLN-based abduction [Kate and Mooney, 2009] converts background axioms into MLN logical formulae by (i) reversing implication and (ii) constructing axioms representing mutual exclusiveness of explanation (e.g. the

set of background knowledge axioms $\{p_1 \rightarrow q, p_2 \rightarrow q, p_3 \rightarrow q\}$ is converted into the following MLN formulae: $q \rightarrow p_1 \vee p_2 \vee p_3$, $q \rightarrow \neg p_1 \vee \neg p_2$, $q \rightarrow \neg p_1 \vee \neg p_3$ etc.). As the readers can imagine, MLN-based approach suffers from the inefficiency of inference due to the increase of converted axioms. Therefore, learning would not scale to larger problems due to the severe overhead [Inoue and Inui, 2012]. Singla and Domingos [2011] report that their MLN-based abduction models cannot be trained in larger dataset.

Moreover, when MLN-based approaches are applied to abduction-based discourse processing, a critical problem arises. MLN-based approaches represent a hypothesis as a truth assignment to ground atoms in the Herbrand base of background knowledge, while our framework represents a hypothesis as a set of first-order literals or equalities of logical variables. This means that a hypothesis generated by MLN-based approaches loses the first-order information in the input text. As shown in Section 1, each logical variable in the observation corresponds to a mention in the discourse; thus losing this information would be a serious drawback in discourse processing. For example, suppose that MLN-based approaches produce the hypothesis $\mathit{president}(A)$, $\mathit{male}(A)$, $\mathit{doctor}(B)$, $\mathit{male}(B)$ (A and B are constants) to the observation $\exists p, m_1, d, m_2\{\mathit{president}(p) \wedge \mathit{male}(m_1) \wedge \mathit{doctor}(d) \wedge \mathit{male}(m_2)\}$. Then, we can interpret this hypothesis as two types of first-order logical forms: $\mathit{president}(p) \wedge \mathit{male}(m_1) \wedge \mathit{doctor}(d) \wedge \mathit{male}(m_2) \wedge p = m_1 \wedge d = m_2$, or $\mathit{president}(p) \wedge \mathit{male}(m_1) \wedge \mathit{doctor}(d) \wedge \mathit{male}(m_2) \wedge p = m_2 \wedge d = m_1$. This means that we cannot decide which discourse mentions are identified as coreferential in the hypothesis generated by MLN-based approaches. Some previous work [Poon and Domingos, 2008; Song et al., 2012] represent coreference relations by introducing special predicates that describe two logical variables are equal, but they use MLNs to create a classifier (i.e. binary log-linear classification model that utilizes a number of features) rather than reasoner. Therefore, it is a non-trivial issue to use these coreference representations with logical inference aimed at complicated commonsense reasoning, which is our goal in abudction-based discourse processing.

5.4 Conclusion

We have proposed a supervised approach for learning the evaluation function of Weighted Abduction. We formulated the learning procedure in the framework of structured learning with hidden variables. Our approach enables us to learn the non-linear evaluation function from partial abductive explanations, which is the typical situation in real-life tasks because constructing complete abductive explanations is usually a cost-consuming task. To the best of our knowledge, this is the first work to address the issue of automatic parameter learning of the evaluation function of Weighted Abduction, which can evaluate both the correctness and informativeness of explanations. In our evaluation, we found that our learning procedure can reduce the value of loss function in each iteration, and learned weights are also robust to unseen dataset.

Our future work includes large-scale evaluation of our learning procedure. We plan to evaluate our procedure on the popular natural language processing tasks, coference resolution with a massive set of axioms extracted from several language resources (e.g. WordNet [Fellbaum, 1998a]). It is also a problem that it takes long time to training weights. This problem will be critical in training on a large data set. We will address this problem by improving of abductive reasoner and optimization methods. As discussed in Hobbs et al. [1993], coreference relation correponds to the unification of two logical variables. We therefore plan to incorporate a term that represents the cost of variable unification in the evaluation function of Weighted Abduction.

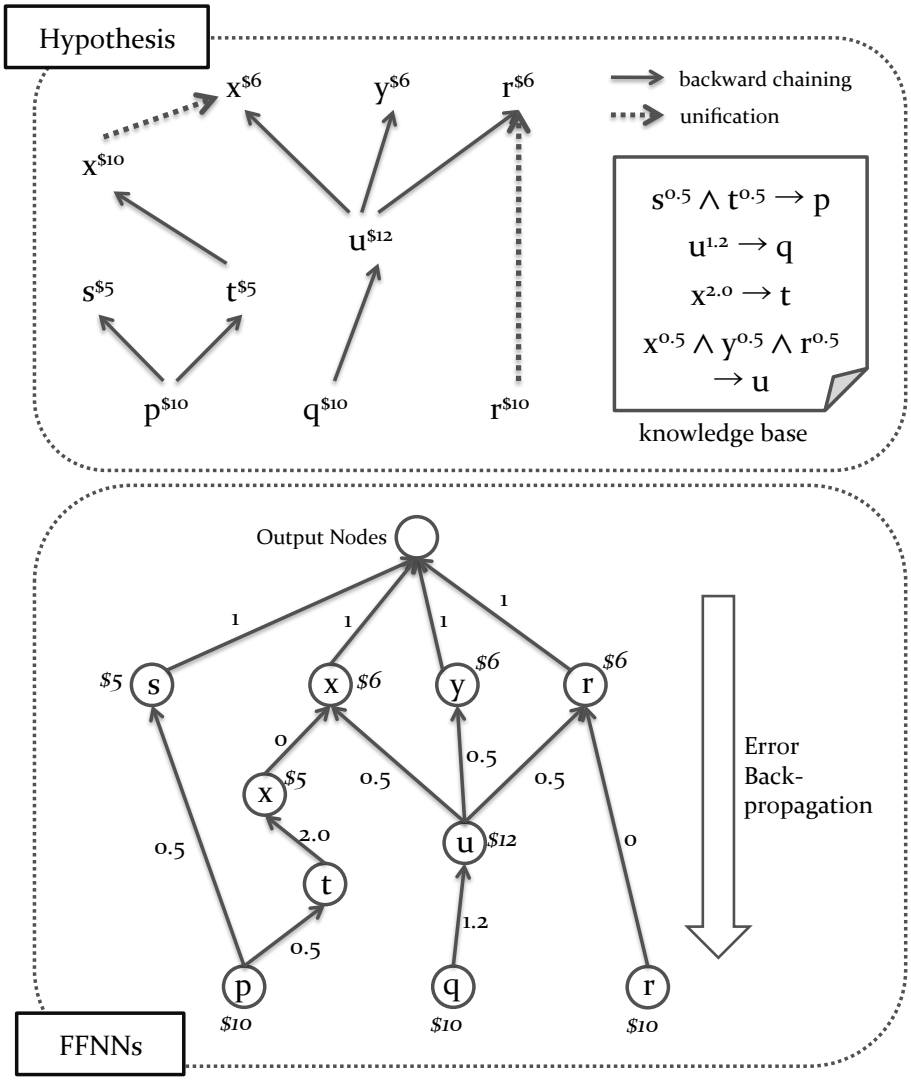

Figure 5.3: Example of transforming hypotheses into FFNNs

(a) Observations
"Bill went to the store. He paid for some milk"
$instance\_going(GO1)$
$goer(GO1, BILL)$
$destination\_go(GO1, STORE)$
$instance\_paying(PAY1)$
$payer(PAY1, BILL)$
$thing\_paid(PAY1, MILK)$

(b) Correct abductive explanations
$instance\_shopping(s)$
$shopper(s, BILL)$
$go\_step(s, GO1)$
$pay\_step(s, PAY1)$
$thing\_shopped\_for(s, MILK)$

(c) Background knowledge
$instance\_shopping(s) \wedge go\_step(s, g) \rightarrow instance\_going(g)$

$instance\_shopping(s) \wedge pay\_step(s, pay) \rightarrow instance\_paying(pay)$

$instance\_shopping(s) \wedge go\_step(s, g) \wedge shopper(s, p) \rightarrow goer(g, p)$

$instance\_shopping(s) \wedge pay\_step(s, pay) \rightarrow payer(pay, p)$

$instance\_shopping(s) \wedge go\_step(s, g) \wedge store(s, str) \rightarrow destination\_go(g, str)$

$instance\_shopping(s) \wedge pay\_step(s, pay) \wedge thing\_shopped\_for(s, t) \rightarrow thing\_paid(pay, t)$

Figure 5.4: Example dataset.

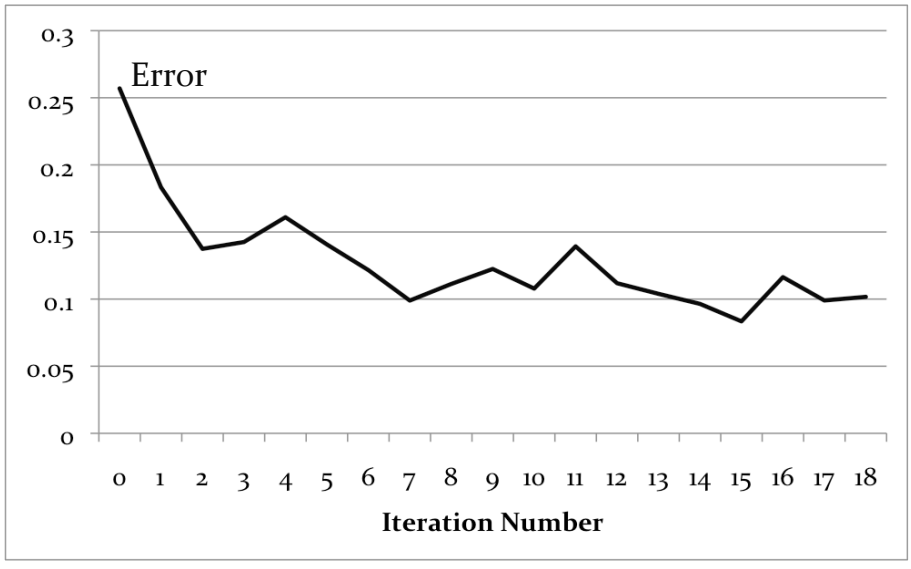

Figure 5.5: Loss function values (closed test)

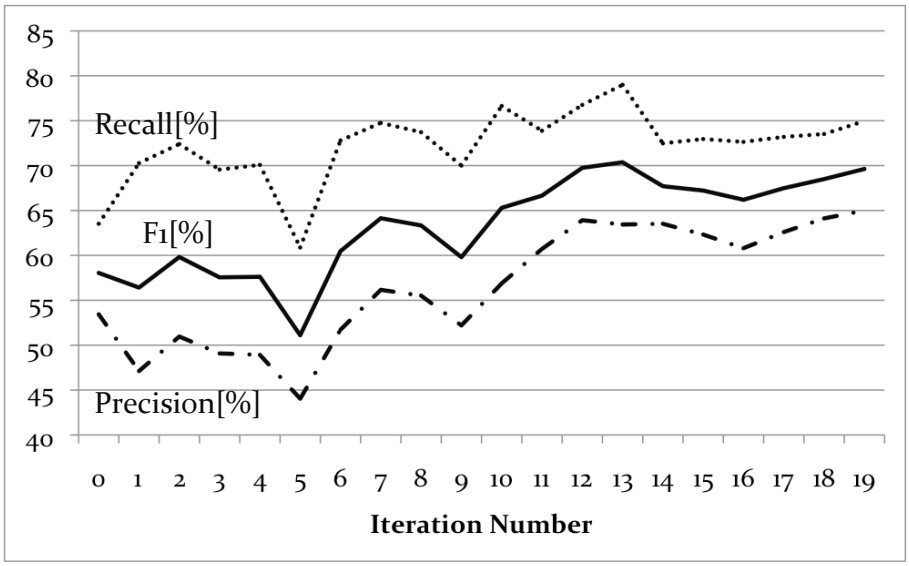

Figure 5.6: Open test results.

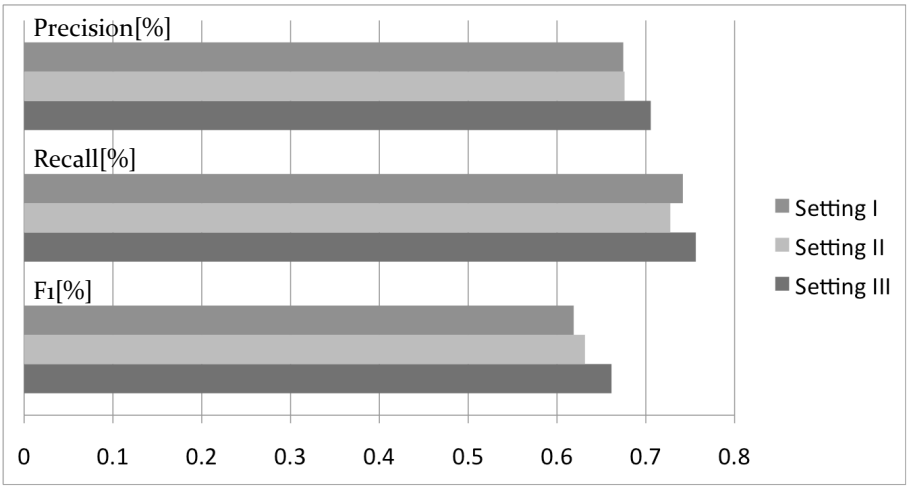

Figure 5.7: Results on each feature setting.

Chapter 6

Scalable and Trainable Open Source Abductive Reasoner

We have implemented the proposed methods in one software package, which is called *Phillip*. The software is publicly available at the Github[1]. In this chapter, we outline Phillip.

6.1 Introduction

On studies of inference-based approaches, the existence of user-friendly inference engines is important. However, existing implementations of abduction, such as Mini-TACITUS [Mulkar et al., 2007] and Henry [Inoue and Inui, 2011, 2012], have problems as follows:

Lack of efficiency: Existing softwares are not enough efficient to perform abductive inference with large knowledge base.

Lack of flexibility: An existing software is nothing more than an implementation of a specific model of abduction. For instance, Mini-TACITUS and Henry are implementations of Weighted Abduction. Ones to develop some new abductive inference model are forced to make a new software from

[1] http://github.com/kazeto/phillip/

scratch, or to be familiar with the implementation of an existing software in order to alter it.

In view of the above, we implemented a new software for abduction as open-source software, *Phillip*. We show special features of Phillip as follows:

Scalable: Phillip is written in C++ and based on the methods which we have proposed in Chapter 3 and Chapter 4. Therefore abduction with Phillip may be much more efficient than with other implementations.

Trainable: The learning method proposed in Chapter 5 is implemented in Phillip. One can easily tune the parameter of the evaluation function.

Flexible: Phillip is able to deal with user-defined evaluation function. One can easily implement his/her new inference model.

Closs-platform: Phillip is available on DOS, OS X and LINUX.

We expect these features to make the entry barriers of the domain of abduction-based discourse processing researches low.

6.2 Basic Usage

In this section, we outline the usage of Phillip. For futher detail, refer to Github wiki[1].

In Phillip, logical formulae are expressed in S-expressions. For instance, the observation and the knowledge base in Figure 2.1 is written as follows:

```
(O (^ (animal x :10) (bark e y :10)))
(B (=> (dog x :1.5) (animal x)))
(B (=> (cat x :1.5) (animal x)))
(B (=> (dog x :1.2) (bark e x)))
(B (=> (poodle x :2.0) (dog x)))
```

The procedure of abduction on Phillip consists of following two steps:

[1]https://github.com/kazeto/phillip/wiki

Compiling step pre-estimates the heuristic distance between predicates in given knowledge base. This step is enough to be executed only once for each knowledge base.

Inference step takes observation (written in S-expressions) as input and outputs the solution hypothesis.

On the inference step, one can configure following three components via command options: (i) how to generate the potential elemental hypotheses, (ii) what ILP problems to convert the potential elemental hypotheses into, and (iii) what ILP solver to use for optimization of the ILP problem converted. Hence Phillip can be adopted for various uses.

6.2.1 User-Defined Evaluation Function

As noted above, each existing abductive reasoner can only specific evaluation function. Hence one who wants to develop some new abductive inference model is forced to make a new software from scratch, or to be familiar with the implementation of an existing software in order to alter it. This circumstance has been making abduction-based discourse processing researches stagnant.

For this problem, Phillip provides the way for users to implement their new abduction model easily. Specifically, one can adopt user-defined components in the inference step noted above. If one wants to develop the model which adopts a new evaluation function, what he or she have to do is only to implement the ILP-conversion component corresponding the evaluation function (i.e., he or she can use built-in components for generation and optimization).

6.3 Conclusion

In this chapter, we outlined the implementation of our methods, Phillip. Phillip is an opensource software for cross-platform and much more efficient than with other implementations. Futhermore, Phillip provides the way for users to implement their new abduction model easily. We expect that Phillip make the entry barriers of the domain of abduction-based discourse processing researches low.

Chapter 7

Conclusions

7.1 Summary

While abduction has long been studied in a wide range of contexts and has been considered a promising framework for natural language processing, its application to real world tasks has been hindered. In this thesis, we have addressed two of the issues which hinder application of abduction to NLP practical problems; *scalability* and *trainability*.

The key contribution of this thesis can be summarized as follows:

1. We proposed an efficient inference method of abductive reasoning on first-order logic. Based on ILP formulated Abduction [Inoue and Inui, 2011; Inoue et al., 2012], the method eliminates redundant inference from the search space.

2. We proposed a method to discriminatively tune the parameters of the evaluation function in first-order abduction. This method is not task-specific nor model-specific and hence it is widely applicable.

3. We have implemented the proposed methods in one software package, which is called *Phillip*. The software is an opensource software and publicly available at the Github[1]. One can easily develop a new abduction framework on Phillip.

[1] http://github.com/kazeto/phillip/

In Chapter 2, we gave outlines of first-order logic and abduction and introduced some previous works. Abduction system takes observation and background knowledge as input and produces the solution hypothesis as output. There have been two big obstacles to apply abduction-based discourse processing to practical problems: (i) how to search the solution hypothesis efficiently and (ii) how to train the evaluation function in a supervised manner.

In Chapter 3 and Chapter 4, we proposed the methods for boosting the computational efficiency of abduction. Here the basic common idea is two folds:

1. They are based on ILP formulated Abduction [Inoue and Inui, 2011; Inoue et al., 2012], whose computational efficiency is state-of-the-art.

2. They achieved improvement in efficiency by eliminating redundant inference from the search space on the potential elemental hypotheses generation step in ILP-formulated Abduction.

In Chapter 3, based on the idea that a literal not contributing any unification is redundant, we proposed a method that eliminates literals not contributing any unification from the search space by using an A* algorithm. In Chapter 4, focusing on the problem of computational cost caused by literals representing thematic roles, we proposed a method that eliminates the redundant operations related with such literals from the search space. Our evaluation revealed that our system is far more efficient than the other existing abductive reasoners.

In Chapter 5, we proposed a method to discriminatively learn the evaluation function of first-order logic-based abduction. This method is not task-specific nor model-specific and is therefore widely applicable. This method can be applied to an evaluation function if only the evaluation function is differentiable with respect to its parameters to tune. In our evaluation, we showed that our learning procedure can reduce the value of loss function in each iteration, and learned parameters are also robust to unseen dataset.

In Chapter 6, we introduced our software, Phillip, and outlined the usage of Phillip. Phillip is an opensourse software for cross platform written in C++, and is the most efficient abductive reasoner. Futhermore, Phillip has good flexibility in its implementation, hence one can easily implement a new abduction-based

discourse processing framework — All he or she have to do is implement his or her evaluation function on Phillip.

7.2 Future Direction

The proposed methods in this thesis could overcome the two obstacles of abduction-based real-life discourse processing noted above. This enables us to evaluate accuracy of abduction-based system for real world task and to compare that system with other existing frameworks empirically.

However, the several problems to solve still remain yet. In the next subsections, we elaborate the issues to address for developing a system of abduction-based real-life discourse processing.

7.2.1 Extending Knowledge Base

In order to achieve abduction-based real-life discourse processing system, we need various kinds of world knowledge, such as causality relation, presupossition relation, lexical category, properties of proper nouns, paraphrase and so on. Since task performance of an abduction-based system strongly depends on size and accuracy of background knowledge, it is important issue how to hervest world knowledge for discourse processing in large quantities and with high accuracy. As a solution to this issue, there are three options as follows:

The first option is, as we extract background knowledge from WordNet and FrameNet in our experiments, to extract knowledge from other existing thesauruses, such as ConceptNet[1], freebase[2] and YAGO[3]. This solution is expected to be able to hervest knowledge about proper nouns in large quantities.

The second option is to acquire knowledge from large corpus by statistical methods. This solution is expected to be able to hervest knowledge about general words, such as causality relation and paraphrase. We will begin with improving a method of knowledge acquisition from ClueWeb12 stated in Section 4.4.1.

[1] http://conceptnet5.media.mit.edu/
[2] http://www.freebase.com
[3] http://www.mpi-inf.mpg.de/yago-naga/yago/

The third option is to create knowledge by crowdsourcing. This solution has the advantage that it can obtain knowledge with high accuracy.

7.2.2 Integrating with Deduction

Our formulation of abduction, which is defined in Section 2.2.1, has the problem that it cannot deal with explanations which include deductive inference.

We show an example in Figure 7.1. Our formulation cannot generate this explanation because the formulation can use background knowledge only backward and the literal $dog(x)$ is infered only from $poodle(x)$ or $bark(e_2, x)$ with *deductive* inference. Our ultimate goal is to develop a system to make implicit information in natural language texts explicit. So it is considered to be a critical problem that a system can deal with only implicit information which explain the observation and can not deal with one which is induced from the observation.

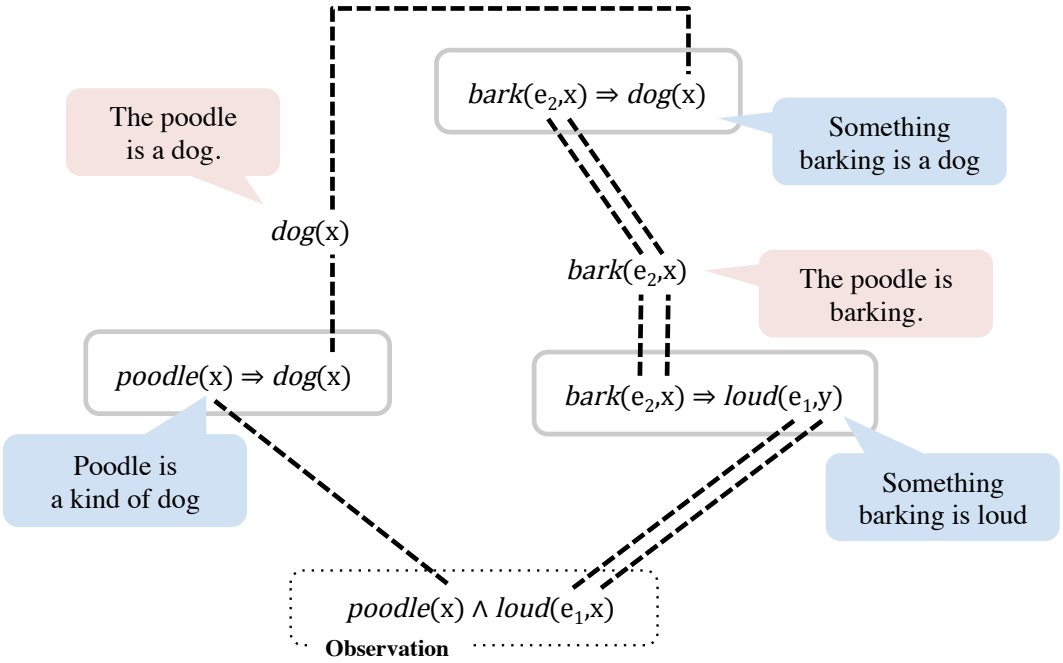

Figure 7.1: An example of the explanation that includes deductive inference.

For this problem, we will consider to extend our formulation so that abduction

integrates with deduction. Specifically, we allow not only backward chaining but also forward chaining on the potential elemental hypotheses generation.

The issue here is that the combination of backward chaining and forward chaining can cause an explosion increase of elemental hypotheses. Hence, we will consider how to integrate abduction with deduction with keeping the computational cost of inference.

7.2.3 Investigating the Better Logical Meaning Representation

As stated in Section 4.1.1, what logical representation to express the information extracted from natural language expressions is an important issue. The biggest one of the problems about the logical meaning representation is how to express the interpretation of a sentence which has a contradictory conjunction, such as "*but*" and "*however*".

For example, let us consider the interpretation of a sentence "*John was shot, but he did not die.*". Supposing knowledge that someone shot tends to die, it is expected that John died. Since this expectation contradicts the observation, the contradictory conjunction "*but*" can interpreted as what express the contradiction between the expectation and the observation. However, existing framework of abduction can not express interpretations like this because a candidate hypothesis is defined to be consistent with the observation.

The most straight-forward solution is to describe discourse relations as literals. For example, the above interpretation can be represented by writing observation and knowledge base as following O and B respectively:

$$\begin{aligned} O &= john(j) \wedge shoot(e_1) \wedge dobj(e_1, x) \wedge die(e_2) \wedge nsubj(e_2, j) \\ &\quad \wedge not(e_3, e_2) \wedge CONTRADICT(e_1, e_2) \qquad (7.1) \\ B &= \{shoot(e_1) \wedge dobj(e_1, x) \wedge CAUSE(e_1, e_2) \Rightarrow die(e_2) \wedge nsubj(e_2, x), \\ &\quad CONTRADICT(e_1, e_2) \Rightarrow CAUSE(e_1, e_2) \wedge not(e_3, e_2)\} \qquad (7.2) \end{aligned}$$

where $not(x, y)$ represents that *not* event x negates y, $CAUSE(x, y)$ represents causality relation between x and y and $CONTRADICT(x, y)$ represents con-

tradiction between x and y. Given O and B, the above interpretation can be represented as shown in Figure 7.2.

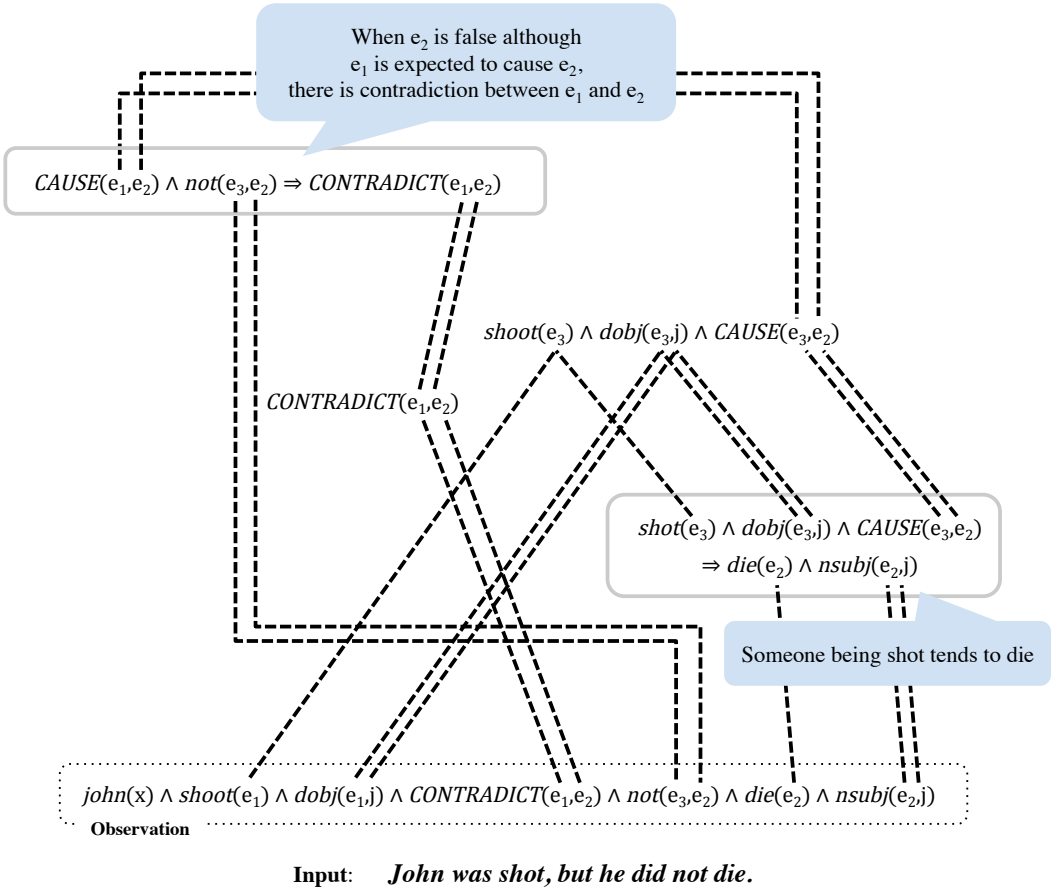

Figure 7.2: An example of the explanation to a sentence which has a contradictory conjunction.

However, this solution may also cause the increase of the number of elemental hypothese and then cause computational inefficiency. Hence, we will explore how to deal with contradictory conjunctions with keeping the computational efficiency of abduction.

7.2.4 Developing a Evaluation Function for Discourse Processing

What solution hypothesis to be outputted from a system is depends on the two factor; its background knowledge and its evaluation function. In order to develop an accurate discourse processing system, an accurate evaluation function — that regards a candidate hypothesis agreeing with the human interpretation highly — is necessary. We consider the existing evaluations to have following problems:

First, the most of existing evaluation functions cannot take plausibility of equality assumptions into account. For example, since Weighted Abduction imposes no cost upon each equality assumption, a pair of literals can be unified if only they share the same predicate. This can be a problem in case when a input sentence includes words not being coreferential but whose surfaces are same.

Second, the most of existing evaluation functions presupposes that each implication rule in background knowledge is valid (i.e., the truth of its premises entails the truth of its conclusion). Hence, they can not take the possibility that a hypothesis explains the observation into account. However, some of implication rules to be used in real-life discourse processing do not satisfy this presupposition, such as $shoot(e_1, x, y) \Rightarrow die(e_2, y)$ (y might be alive although he or she has been shot). This contradiction is considered to work badly on the weight learning.

We will explore the evaluation function which can deal with these issues.

Proof of Theorem

.1 Proof of Safety of Pruning Heuristic Estimated Distances

In this section, we proof that the method of pruning HEDs proposed in Section 4.3 preserve the optimality of the solution hypothesis iff the Validity Condition and Simplicity Condition are satisfied. We denote HED between a predicate pair $\{p, q\}$ before and after applying the method as $h_1(p, q)$ and $h_2(p, q)$ respectively.

Now, we focus on a set of operation path (we denote $\mathbb{R}$) which is eliminated by this method. Formally, supposing that an operation path in $\mathbb{R}$ connects observable literals o_1 and o_2, $h_1(o_1, o_2) < \infty$ and $h_2(o_1, o_2) = \infty$ are satisfied. Our goal here is to prove that a candidate hypothesis generated by an operation path $R \in \mathbb{R}$ cannot be the solution hypothesis iff the Validity Condition and Simplicity Condition are satisfied.

The proving strategy is as follows. First, using the definitions of the Validity Condition and the Simplicity Condition, we prove that a candidate hypothesis generated by an operation path in a subset of $\mathbb{R}$ cannot be the solution hypothesis. Next, denoting the others as $\mathbb{R}'$, we prove that an operation path in $\mathbb{R}'$ always includes an unification operation between functional literals and the unification operation introduces invalid equality assumptions.

From the Validity Condition and the Simplicity Condition, a operation path R is not included in the solution hypothesis obviously iff it holds at least one out of the following cases:

(i) R includes unification between functional literals and at least one out of those functional literals has no parent.

(ii) R includes a backward chaining operation which introduces invalid equality assumptions.

(iii) the anchors of R are functional literals, they share the same parent, and the evidence of R includes no content literal excluding the parent.

(iv) one of the anchors of R is a functional literal, another is its parent, and the evidence of R includes no content literal excluding the parent.

Obviously, a operation path which satisfies (i) or (ii) is not included in the solution hypothesis from the Validity Condition, and so is a operation path which satisfies (iii) or (iv) from the Simplicity Condition. Therefore, we consider about the operation paths $\mathbb{R}' \subseteq \mathbb{R}$ which do not satisfy the above conditions, hereafter. Since each operation path in $\mathbb{R}'$ does not include a backward chaining operation which introduces invalid equality assumptions, it is enough to prove that an operation path in $\mathbb{R}'$ always includes an unification operation which introduces invalid equality assumptions.

To begin with, we prove that an operation path in $\mathbb{R}'$ always includes an unification operation for functional literals.

Proof We use the reductio ad absurdum. Namely we show that, the claim that an operation path in $\mathbb{R}'$ never includes unification between functional literals implies a contradiction. From the definition of operation path, this claim is equal to the claim that an operation path in $\mathbb{R}'$ can include unification between content literals. More formally, the claim is that, there exists a pair of observable literals $\{o_1, o_2\}$ which satisfies the following: (1) $\{o_1, o_2\}$ satisfies $h_1(o_1, o_2) < \infty$ and $h_2(o_1, o_2) = \infty$, (2) o_1 and o_2 are explained by content literals c_1 and c_2 respectively and (3) c_1 and c_2 are unifiable.

Since c_1 and c_2 are unifiable, they share the same predicate and hence they satisfy $h_1(c_1, c_2) = h_2(c_1, c_2) = 0 < \infty$. Since an operation path in $\mathbb{R}'$ never includes backward chaining which introduces invalid equality assumptions, a literal d_1, which c_1 directly explains, satisfies $h_2(c_1, d_1) < \infty$. Similarly, a literal d_w, which c_2 directly explains, satisfies $h_2(c_2, d_2) < \infty$. Consequently, $h_2(c_1, c_2) < \infty$ is satisfied and then $h_2(d_1, d_2) < \infty$ is satisfied. The arguments like this can be applied to literals which d_1 or ${}_d2$ explains, hence $h_2(o_1, o_2) < \infty$ can be induced.

This is inconsistent with $h_2(o_1, o_2) = \infty$ in the definition of $\mathbb{R}$. Therefore, an operation path in $\mathbb{R}'$ always includes unification between functional literals.

Next, denoting the anchors of an operation path in $\mathbb{R}$ as f_1 and f_2, we prove that unification between f_1 and f_2 always introduces invalid equality assumptions.

Proof Denoting the parents of f_1 and f_2 as p_1 and p_2 respectively, we prove that p_1 and p_2 satisfy $h_2(p_1, p_2) = \infty$ using the reductio ad absurdum.

Here the claim for the reductio ad absurdum is that p_1 and p_2 satisfy $h_2(p_1, p_2) < \infty$. From the definition of HEDs for functional literals (defined in Section 4.3.3), the claim leads $h_2(f_1, f_2) < \infty$ and therefore leads $h_2(o_1, o_2) < \infty$.

This is inconsistent with $h_2(o_1, o_2) = \infty$ in the definition of $\mathbb{R}$. Therefore, p_1 and p_2 cannot be coreferent and then the unification between f_1 and f_2 always introduce invalid equality assumptions.

From these proofs, an operation path in $\mathbb{R}$ can be classified into following two types: (1) it holds at least one out of the above four cases and (2) it include unification between functional literals whose parents cannot be coreferent. Therefore, it is proved that all of operation path in $\mathbb{R}$ cannot be included in the solution hypothesis and the method of pruning HEDs preserve the optimality of the solution hypothesis.

References

J. Blythe, J. R. Hobbs, P. Domingos, R. J. Kate, and R. J. Mooney. Implementing Weighted Abduction in Markov Logic. In *IWCS*, pages 55–64, 2011. 12, 15, 55, 69

N. Chambers and D. Jurafsky. Unsupervised Learning of Narrative Schemas and their Participants. In *ACL*, pages 602–610, 2009. 55

E. Charniak and R. P. Goldman. A Probabilistic Model of Plan Recognition. In *AAAI*, pages 160–165, 1991. 7

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2-3): 281–332, 2005. 33

K. Crammer, O. Dekel, J. Keshet, and Y. Singer S. Shalev-Shwartz. Online Passive-Aggressive Algorithms. pages 551–585, 2006. 12

I. Dagan, B. Dolan, B. Magnini, and D. Roth. Recognizing textual entailment: Rational, evaluation and approaches - Erratum. *Natural Language Engineering*, 16(1):105, 2010. 55

Donald Davidson. *Essays on Actions and Events*. Oxford University Press, 1980. 33

C. Fellbaum. WordNet: an electronic lexical database. 1998a. 24, 55, 71

Christian Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998b. 47

Joseph E Gonzalez, Yucheng Low, Carlos E Guestrin, and David O'Hallaron. Distributed Parallel Inference on Large Factor Graphs. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009. 23

J. R. Hobbs. Ontological promiscuity. In *ACL*, pages 61–69, Chicago, Illinois, 1985. 33

J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993. 7, 10, 24, 25, 36, 48, 55, 56, 71

D. Hovy, C. Zhang, E. Hovy, and A. Penas. Unsupervised discovery of domain-specific knowledge from text. In *ACL*, pages 1466–1475, 2011. 55

T. N. Huynh and R. J. Mooney. Max-Margin Weight Learning for Markov Logic Networks. In *SRL*, 2009. 56

N. Inoue and K. Inui. ILP-Based Reasoning for Weighted Abduction. In *Proceedings of AAAI Workshop on Plan, Activity and Intent Recognition*, 2011. 8, 9, 15, 22, 32, 38, 43, 55, 66, 75, 78, 79

N. Inoue and K. Inui. Large-scale Cost-based Abduction in Full-fledged First-order Predicate Logic with Cutting Plane Inference. In *Proceedings of the 13th European Conference on Logics in Artificial Intelligence*, pages 281–293, 2012. 9, 15, 16, 19, 24, 25, 28, 38, 43, 70, 75

Naoya Inoue, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. Online large-margin weight learning for first-order logic-based abduction. In *Proceedings of the 15th Information-Based Induction Sciences Workshop*, pages 143–150, 2012. 12, 32, 78, 79

Vladimir Jojic, Stephen Gould, and Daphne Koller. Accelerated dual decomposition for MAP inference. In *Proceedings of the 25th International Conference on Machine Learning*, pages 503–510, 2010. 23

R. J. Kate and R. J. Mooney. Probabilistic abduction using markov logic networks. In *Proceedings of the IJCAI-09 Workshop on Plan, Activity, and Intent Recognition (PAIR-09)*, Pasadena, CA, July 2009. 56, 66, 69

H. J. Levesque. The Winograd Schema Challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011. 46, 47

D. Lowd and P. Domingos. Efficient Weight Learning for Markov Logic Networks. In *PKDD*, pages 200–211, 2007. 56

Michael C McCord. *Slot grammar*. Springer, 1990. 33

R. Mulkar, J. Hobbs, and E. Hovy. Learning from Reading Syntactically Complex Biology Texts. In *Proceedings of the 8th International Symposium on Logical Formalizations of Commonsense Reasoning*, Palo Alto, 2007. 75

H. T. Ng and R. J. Mooney. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *KR*, pages 499–508, 1992. 66

Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. Scaling Inference for Markov Logic via Dual Decomposition. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1032–1037, 2012. 23

E. Ovchinnikova, N. Montazeri, T. Alexandrov, J. Hobbs, M. McCord, and R. Mulkar-Mehta. Abductive Reasoning with a Large Knowledge Base for Discourse Processing. In *Proceedings of the International Conference on Computational Semantics*, pages 225–234, Oxford, UK, 2011. 55

Terence. Parsons. *Events in the semantics of English : a study in subatomic semantics / Terence Parsons*. MIT Press Cambridge, Mass, 1990. ISBN 0262161206. 33

Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 650–659, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D08-1068`.

S. Raghavan and R. J. Mooney. Bayesian Abductive Logic Programs. In *Star-AI 10*, pages 82–87, 2010. 7, 15, 66

A. Rahman and V. Ng. Resolving Complex Cases of Definite Pronouns: The Winograd Schema Challenge. In *Proceedings of EMNLP-CoNLL*, pages 777–789, 2012. 46

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, pages 107–136, 2006. 12, 15, 56, 69

J. Ruppenhofer, M. Ellsworth, M.R. Petruck, C.R. Johnson, and J. Scheffczyk. FrameNet II: Extended Theory and Practice. Technical report, 2010. 24, 55

S. Schoenmackers, J. Davis, O. Etzioni, and D. Weld. Learning First-order Horn Clauses from Web Text. In *EMNLP*, pages 1088–1098, 2010. 55

P. Singla and P. Domingos. Abductive Markov Logic for Plan Recognition. In *AAAI-11*, pages 1069–1075, 2011. 7, 15, 56, 66, 67, 69, 70

Yang Song, Jing Jiang, Wayne Xin Zhao, Sujian Li, and Houfeng Wang. Joint learning for coreference resolution with markov logic. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1245–1254, Jeju Island, Korea, July 2012. ACL. 70

P. R. Thagard. The best explanation: Criteria for theory choice. *The Journal of Philosophy*, 1:76–92, 1978. 11

Jacopo Urbani, Spyros Kotoulas, Eyal Oren, and Frank Van Harmelen. Scalable distributed reasoning using mapreduce. In *Proceedings of the 8th International Semantic Web Conference*, pages 634–649. Springer, 2009. 23

List of Publications

Journal Papers (Refereed)

1. Kazeto Yamamoto, Naoya Inoue, Kentaro Inui. Boosting Abductive Reasoning with Functional Literals. (Submitted)

2. Kazeto Yamamoto, Naoya Inoue, Kentaro Inui, Yuki Arase and Jun'ichi Tsujii. Boosting the Efficiency of First-order Abductive Reasoning Using Pre-estimated Relatedness between Predicates. International Journal of Machine Learning and Computing, Vol.5, No.2, pp.114-120. April 2015.

International Conference/Workshop Papers (Refereed)

1. Kazeto Yamamoto, Naoya Inoue, Yotaro Watanabe, Naoaki Okazaki and Kentaro Inui. Discriminative Learning of First-order Weighted Abduction from Partial Discource Explanations. Proceedings of the 14th International Conference on Intelligent Text Processing and Computational Linguistics, pp.1291-1380, March 2013.

2. Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki and Kentaro Inui. Finding the Best Model among Representative Compositional Models. Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing, pp.65-74, December 2014.

3. Atsushi Keyaki, Yuki Arase, Kazeto Yamamoto and Jun'ichi Tsujii. Word Sence Disambiguation with Integer Linear Programming using Word Sence Graph (in Japanese). Proceedings of WebDB Forum 2014, Nobember 2014.

Awards

1. 2014 IACSIT Barcelona Conferences Excellent Oral Presentation (2014)
2. Annual Meeting Excellent Paper, Association for Natural Language Processing (2014)
3. Annual Meeting Excellent Paper, Association for Natural Language Processing (2013)
4. Honorable Mention, The 7th NLP Symposium for Young Researchers (2012)
5. Honorable Mention, IPSJ-SIGNL (2012)

Other Publications (Not refereed)

1. Kazeto Yamamoto, Naoya Inoue and Kentaro Inui. Phillip: Abductive Reasoner for Discourse Processing (in Japanese). Proceedings of the 21st Annual Meeting of the Association for Natural Language Processing, pp.377-380, March 2015.
2. Kazeto Yamamoto, Naoya Inoue, Kentaro Inui, Yuki Arase and Jun'ichi Tsujii. Boosting Efficiency of Abductive Reasoning based on A* Algorithm (in Japanese). In IPSJ SIG Technical Reports, Vol.2014-NL-217, July 2014.
3. Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki and Kentaro Inui. Generating Phrase Embedding using Dependencies (in Japanese). Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing, pp.1055-1058, March 2014.
4. Sonse Shimaoka, Masayasu Muraoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki and Kentaro Inui. Distributional Meaning Representation of Words and Phrases with Gaussian Distribution (in Japanese). Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing, pp.1051-1054, March 2014.
5. Sonse Shimaoka, Kazeto Yamamoto and Kentaro Inui. Learning of Word Embedding by Auto Encoder (in Japanese). Proceedings of the 19th Annual

Meeting of the Association for Natural Language Processing, pp.612-619, March 2013.

6. Naoya Inoue, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki and Kentaro Inui. Online Large-margin Weight Learning for First-order Logic-based Abduction. In Proceedings of the 15th Information-Based Induction Sciences Workshop, pp.143-150, November 2012.

7. Kazeto Yamamoto, Naoya Inoue, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. Backpropagation Learning for Weighted Abduction (in Japanese). In IPSJ SIG Technical Reports, Vol.2012-NL-206, May 2012.