GENERALIZED CHART ALGORITHM: AN EFFICIENT PROCEDURE FOR COST-BASED ABDUCTION

Yasuharu Den
ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, JAPAN
Tel: +81-7749-5-1328, Fax: +81-7749-5-1308, e-mail: den@itl.atr.co.jp

Abstract

We present an efficient procedure for cost-based abduction, which is based on the idea of using chart parsers as proof procedures. We discuss in detail three features of our algorithm — *goal-driven bottom-up derivation*, *tabulation of the partial results*, and *agenda control mechanism* — and report the results of the preliminary experiments, which show how these features improve the computational efficiency of cost-based abduction.

Introduction

Spoken language understanding is one of the most challenging research areas in natural language processing. Since spoken language is incomplete in various ways, i.e., containing speech errors, ellipsis, metonymy, etc., spoken language understanding systems should have the ability to process incomplete inputs by hypothesizing the underlying information. The abduction-based approach (Hobbs *et al.*, 1988) has provided a simple and elegant way to realize such a task.

Consider the following Japanese sentence:

(1)	*Sôseki*	*kat-*	*ta*
	(a famous writer)	buy	PAST

This sentence contains two typical phenomena arising in spoken language, i.e., *metonymy* and the *ellipsis* of a particle. When this sentence is uttered under the situation where the speaker reports his experience, its natural interpretation is *the speaker bought a Sôseki novel*. To derive this interpretation, we need to resolve the following problems:

- The metonymy implied by the noun phrase *Sôseki* is expanded to *a Sôseki novel*, based on the pragmatic knowledge that the name of a writer is sometimes used to refer to his novel.
- The particle-less thematic relation between the verb *katta* and the noun phrase *Sôseki* is determined to be the object case relation, based on the semantic knowledge that the object case relation between a trading action and a commodity can be linguistically expressed as a thematic relation.

This interpretation is made by abduction. For instance, the above semantic knowledge is stated, in terms of the predicate logic, as follows:

(2) $sem(e,x) \subset trade(e) \wedge commodity(x) \wedge obj(e,x)$

Then, the inference process derives the consequent $sem(e,x)$ by hypothesizing an antecedent $obj(e,x)$, which is never proved from the observed facts. This process is called *abduction*.

Of course, there may be several other possibilities that support the thematic relation $sem(e,x)$. For instance, the thematic relation being determined to be the agent case relation, sentence (1) can have another interpretation, i.e., *Sôseki bought something*, which, under some other situations, might be more feasible than the first interpretation. To cope with feasibility, the abduction-based model usually manages the mechanism for evaluating the goodness of the interpretation. This is known as *cost-based abduction* (Hobbs *et al.*, 1988).

In cost-based abduction, each assumption bears a certain cost. For instance, the assumption $obj(e,x)$, introduced by applying rule (2), is specified to have a cost of, say, \$2. The goodness of the interpretation is evaluated by accumulating the costs of all the assumptions involved. The whole process of interpreting an utterance is depicted in the following schema:

1. Find all possible interpretations, and
2. Select the one that has the lowest cost.

In our example, the interpretation that assumes the thematic relation to be the object case relation, with the metonymy being expanded to *a Sôseki novel*, is cheaper than the interpretation that assumes the thematic relation to be the agent case relation; hence, the former is selected.

An apparent problem here is the high computational cost; because abduction allows many possibilities, the schema involves very heavy computation. Particularly in the spoken language understanding task, we need to consider a great number of possibilities when hypothesizing various underlying information. This makes the abduction process

computationally demanding, and reduces the practicality of abduction-based systems. The existing models do not provide any basic solution to this problem. Charniak (Charniak and Husain, 1991; Charniak and Santos Jr., 1992) dealt with the problem, but those solutions are applicable only to the *propositional* case, where the search space is represented as a directed graph over *ground* formulas. In other words, they did not provide a way to build such graphs from rules, which, in general, contain variables and can be recursive.

This paper provides a basic and practical solution to the computation problem of cost-based abduction. The basic idea comes from the natural language parsing literature. As Pereira and Warren (1983) pointed out, there is a strong connection between parsing and deduction. They showed that parsing of DCG can be seen as a special case of deduction of Horn clauses; conversely, deduction can be seen as a generalization of parsing. Their idea of using chart parsers as deductive-proof procedures can easily be extended to the idea of using chart parsers as abductive-proof procedures. Because chart parsers have many advantages from the viewpoint of computational efficiency, chart-based abductive-proof procedures are expected to nicely solve the computation problem. Our algorithm, proposed in this paper, has the following features, which considerably enhance the computational efficiency of cost-based abduction:

1. *Goal-driven bottom-up derivation*, which reduces the search space.
2. *Tabulation of the partial results*, which avoids the recomputation of the same goal.
3. *Agenda control mechanism*, which realizes various search strategies to find the best solution efficiently.

The rest of the paper is organized as follows. First, we explain the basic idea of our algorithm, and then present the details of the algorithm along with simple examples. Next, we report the results of the preliminary experiments, which clearly show how the above features of our algorithm improve the computational efficiency. Then, we compare our algorithm with Pereira and Warren's algorithm, and finally conclude the paper.

Head-driven Derivation

Pereira and Warren showed that chart parsers can be used as proof procedures; they presented the *Earley deduction* proof procedure, that is a generalization of top-down chart parsers. However, they mentioned only top-down chart parsers, which is not always very efficient compared to bottom-up (left-corner) chart parsers. It seems that using left-corner parsers as proof procedures is not so easy, unless the rules given to the provers have a certain property. Here, we describe under what conditions left-corner parsers can be used as proof procedures.

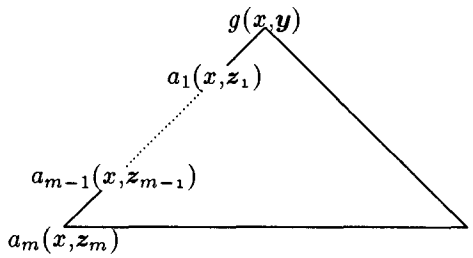

Figure 1: Concept of Head-driven Derivation

Let us begin with the general problems of Horn clause deduction with naive top-down and bottom-up derivations:

- Deduction with top-down derivation is affected by the frequent backtracking necessitated by the inadequate selection of rules to be applied.
- Deduction with bottom-up derivation is affected by the extensive vacuous computation, which never contributes to the proof of the initial goal.

These are similar to the problems that typically arise in natural language parsing with naive top-down and bottom-up parsers. In natural language parsing, these problems are resolved by introducing a more sophisticated derivation mechanism, i.e., *left-corner parsing*. We have attempted to apply such a sophisticated mechanism to deduction.

Suppose that the proof of a goal $g(x,y)$ can be represented in the manner in Figure 1; the first argument x of the goal $g(x,y)$ is shared by all the formulas along the path from the goal $g(x,y)$ to the left corner $a_m(x,z_m)$. In such a case, we can think of a derivation process that is similar to left-corner parsing. We call this derivation *head-driven derivation*, which is depicted as follows:

Step 1 Find a fact $a(w,z)$ whose first argument w unifies with the first argument x of the goal $g(x,y)$, and place it on the left corner.

Step 2 Find a rule $a_{m-1}(w,z_{m-1}) \subset a(w,z_m) \wedge B_1 \wedge \ldots \wedge B_n$ whose leftmost antecedent $a(w,z_m)$ unifies with the left-corner key $a(x,z)$, and introduce the new goals $B_1, \ldots,$ and B_n. If all these goals are recursively derived, then create the consequent $a_{m-1}(x,z_{m-1})$, which dominates $a(x,z_m)$, $B_1, \ldots,$ and B_n, and place it on the left corner instead of $a(x,z)$.

Step 3 If the consequent $a_{m-1}(x,z_{m-1})$ unifies with the goal $g(x,y)$, then finish the process. Otherwise, go back to **step 2** with $a_{m-1}(x,z_{m-1})$ being the new left-corner key.

Left-corner parsing of DCG is just a special case of head-driven derivation, in which the input string is shared along the left border, i.e., the path from a nonterminal to the leftmost word in the string that is dominated by that nonterminal. Also, semantic-head-driven generation (Shieber *et al.*, 1989; van Noord, 1990) and head-corner parsing (van Noord, 1991; Sikkel and op den Akker, 1993) can be seen as head-driven derivation, when the semantic-head/syntactic-head is moved to the leftmost position in the body of each rule and the argument representing the semantic-feature/head-feature is moved to the first position in the argument list of each formula.

To apply the above procedures, all rules must be in *chain form* $a_{m-1}(w,z_{m-1}) \subset a_m(w,z_m) \wedge B_1 \wedge \ldots \wedge B_n$; that is, in every rule, the first argument of the leftmost antecedent must be equal to the first argument of the consequent. This is the condition under which left-corner parsers can be used as proof procedures. Because this condition is overly restrictive, we extend the procedures so that they allow *non-chain rules*, i.e., rules not in chain form. **Step 1** is replaced by the following:

> **Step 1** Find a non-chain rule $a(w,z) \subset B_1 \wedge \ldots \wedge B_n$ such that the first argument w of the consequent $a(w,z)$ unifies with the first argument x of the goal $g(x,y)$, and introduce the new goals B_1, ..., and B_n. A fact is regarded as a non-chain rule with an empty antecedent. If all these goals are recursively derived, then create the consequent $a(x,z)$, which dominates B_1, ..., and B_n, and place it on the left corner.

Generalized Chart Algorithm

The idea given in the previous section realizes the goal-driven bottom-up derivation, which is the first feature of our algorithm. Then, we present a more refined algorithm based upon the idea, which realizes the other two features as well as the first one.

Chart Parsing and its Generalization

Like left-corner parsing, which has the drawback of repeatedly recomputing partial results, head-driven derivation will face the same problem when it is executed in a depth-first manner with backtracking. In the case of left-corner parsing, the problem is resolved by using the tabulation method, known as *chart parsing* (Kay, 1980). A recent study by Haruno *et al.* (1993) has shown that the same method is applicable to semantic-head-driven generation. The method is also applicable to head-driven derivation, which is more general than semantic-head-driven generation.

To generalize charts to use in proof procedures,

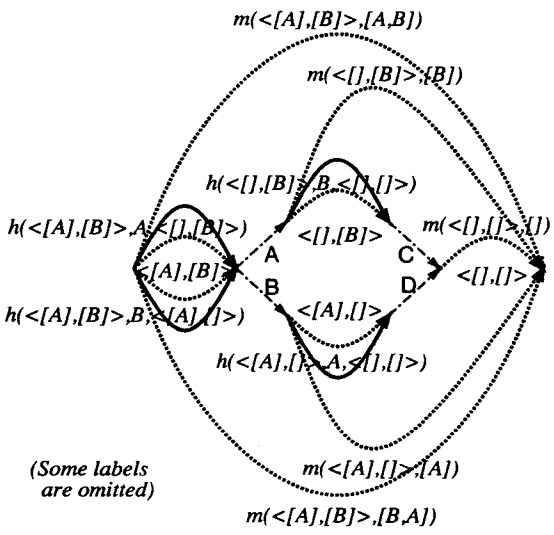

Figure 2: Example of Generalized Charts

we first define the *chart lexicons*. In chart parsing, lexicons are the words in the input string, each of which is used as the index for a subset of the edges in the chart; each edge incident from (the start-point of) lexicon w represents the substructure dominating the sub-string starting from w. In our case, from the similarity between left-corner parsing and head-driven derivation, lexicons are the terms that occur in the first argument position of any formula; each edge incident from (the start-point of) lexicon x represents the substructure dominating the successive sequence of the derived formulas starting from the fact in which x occupies the first argument position. For example, in the chart representing the proof in Figure 1, all the edges corresponding to the formulas on the left border, i.e. $a_m(x,z_m)$, $a_{m-1}(x,z_{m-1})$, ..., $a_1(x,z_1)$ and $g(x,y)$, are incident from (the start-point of) lexicon x, and, hence, x is the index for these edges.

Following this definition of the chart lexicons, there are two major differences between chart parsing and proof procedures, which Haruno also showed to be the differences between chart parsing and semantic-head-driven generation.

1. In contrast to chart parsing, where lexicons are determined immediately upon input, in proof procedures lexicons should be incrementally introduced.
2. In contrast to chart parsing, where lexicons are connected one by one in a linear sequence, in proof procedures lexicons should be connected in many-to-many fashion.

In proof procedures, the chart lexicons are not determined at the beginning of the proof (because

we don't know which formulas are actually used in the proof), rather they are dynamically extracted from the subgoals as the process goes. In addition, if the rules are nondeterministic, it sometimes happens that there are introduced, from one left-corner key, $a(x,z)$, two or more distinct successive subgoals, $b^1(w^1,y^1)$, $b^2(w^2,y^2)$, etc., that have different first arguments, w^1, w^2, etc. In such a case, one lexicon x should be connected to two or more distinct lexicons, w^1, w^2, etc. Furthermore, it can happen that two or more distinct left-corner keys, $a^1(x^1,z^1)$, $a^2(x^2,z^2)$, etc., incidentally introduce the successive subgoals, $b^1(w,y^1)$, $b^2(w,y^2)$, etc., with the same first argument w. In such a case, two or more distinct lexicons, x^1, x^2, etc., should be connected to one lexicon w. Therefore, the connections among lexicons should be many-to-many. Figure 2 shows an example of charts with many-to-many connections, where the connections are represented by pointers **A**, **B**, etc.

The Algorithm

We, so far, have considered deduction but not abduction. Here, we extend our idea to apply to abduction, and present the definition of the algorithm.

The extension for abduction is very simple. First, we add a new procedure, which introduces an assumption G for a given goal G. An assumption is treated as if it were a fact. This means that an assumption, as well as a fact, is represented as a passive edge in terms of the chart algorithm. Second, we associate a set $\mathcal{S}$ of assumptions with each edge e in the chart; $\mathcal{S}$ consists of all the assumptions that are contained in the completed part of the (partial) proof represented by the edge e. More formally, the assumption set $\mathcal{S}$ associated with an edge e is determined as follows:

1. If e is a passive edge representing an assumption A, then $\mathcal{S} = \{A\}$.
2. If e is a passive/active edge introduced from a non-chain rule, including fact, then $\mathcal{S}$ is empty.
3. If e is a passive/active edge predicted from a chain rule with a passive edge e' being the left-corner key, then $\mathcal{S}$ is equal to the assumption set $\mathcal{S}'$ of e'.
4. If e is a passive/active edge created by combining an active edge e_1 and a passive edge e_2, then $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ where $\mathcal{S}_1$ and $\mathcal{S}_2$ are the assumption sets of e_1 and e_2, respectively.

Taking these into account, the definition of our algorithm is as follows. f is a function that assigns a unique vertex to each chart lexicon. The notation $A{:}\mathcal{S}$ stands for the label of an edge e, where A is the label of e in an ordinary sense and $\mathcal{S}$ is the assumption set associated with e.

Initialization Add an active edge $[{}^?G]\top{:}\phi$ to the chart, looping at vertex 0, where G is the initial goal.

Apply the following procedures repeatedly until no procedures are applicable.

Introduction Let e be an active edge labeled $[\cdots[^?]B_j\cdots]A{:}\mathcal{S}$ incident from vertex s to t, where $B_j = b_j(x_j,y_j)$ is the first open box in e.

1. If the lexicon x_j is never introduced in the chart, then introduce it and run a pointer from t to $f(x_j)$. Then, do the following:
   (a) For every non-chain rule $a(w,z) \subset B_1 \wedge \ldots \wedge B_n$, including fact, such that w unifies with x_j, create an active edge labeled $[[^?]B_1\cdots[^?]B_n]a(x_j,z){:}\phi$ between vertex $f(x_j)$ and $f(x_j)+1$. (Create, instead, a passive edge labeled $a(x_j,z){:}\phi$ when the rule is a fact, i.e. $n = 0$.)
   (b) Create a passive edge labeled $B_j{:}\{B_j\}$ between vertex $f(x_j)$ and $f(x_j)+1$.
2. If the lexicon x_j was previously introduced in the chart, then run a pointer from t to $f(x_j)$. In addition, if the passive edge $B_j{:}\{B_j\}$ never exists in the chart, create it between vertex $f(x_j)$ and $f(x_j)+1$.

Prediction Let e be a passive edge labeled $C{:}\mathcal{S}$ incident from vertex s to t. For every chain rule $A' \subset A \wedge B_1 \wedge \ldots \wedge B_n$ such that A unifies with C, create an active edge labeled $[A[^?]B_1\cdots[^?]B_n]A'{:}\mathcal{S}$ between vertex s and t. (Create, instead, a passive edge labeled $A'{:}\mathcal{S}$ when A is the single antecedent, i.e., $n = 0$.)

Combination Let e_1 be an active edge labeled $[\cdots[^?]B_j[^?]B_{j+1}\cdots[^?]B_n]A{:}\mathcal{S}_1$ incident from vertex s to t, where B_j is the first open box in e_1, and let e_2 be a passive edge labeled $C{:}\mathcal{S}_2$ incident from vertex u to v. If B_j and C unify and there is a pointer from t to u, then create an active edge labeled $[\cdots B_j[^?]B_{j+1}\cdots[^?]B_n]A{:}\mathcal{S}_1\cup\mathcal{S}_2$ between vertex s and v. (Create, instead, a passive edge labeled $A{:}\mathcal{S}_1 \cup \mathcal{S}_2$ when B_j is the last element, i.e., $j = n$.)

Each passive edge $\top{:}\mathcal{S}$ represents an answer.

Examples

Here, we present a simple example of the application of our algorithm to spoken language understanding. Figure 3 provides the rules for spoken Japanese understanding, with which the sentence (1) is parsed and interpreted. They include the pragmatic, semantic and knowledge rules as well as the syntactic and lexical rules.

The syntactic rules allow the connection between a verb and a noun phrase with or with-

Syntactic Rules

$s(i,k,e) \subset vp(i,k,e)$
$vp(i,k,e) \subset np(i,j,c,x) \wedge vp(j,k,e) \wedge depend(\langle c,e,x \rangle_d)$
$vp(i,k,e) \subset np(i,j,x) \wedge vp(j,k,e) \wedge depend(\langle c,e,x \rangle_d)$
$np(i,k,c,x) \subset np(i,j,x) \wedge p(j,k,c)$
$depend(\langle c,e,x \rangle_d) \subset prag(\langle x \rangle_p, y) \wedge sem(\langle c,e,y \rangle_s)$

Lexical Rules

$np([Sôseki|k],k,x) \subset soseki(x)^{\$1}$
$vp([katta|k],k,e) \subset buy(e)^{\$1}$
$p([ga|k],k,c) \subset ga(c)^{\$1}$
$p([wo|k],k,c) \subset wo(c)^{\$1}$

Pragmatic Rules

$prag(\langle x \rangle_p, x)$
$prag(\langle x \rangle_p, y) \subset writer(x) \wedge write(\langle x,y \rangle_p)^{\$10} \wedge novel(y)^{\$1}$

Semantic Rules

$sem(s) \subset ga(s,c) \wedge ga(c)^{\$3}$
$sem(s) \subset wo(s,c) \wedge wo(c)^{\$3}$
$ga(\langle c,e,x \rangle_s, c) \subset intend(e) \wedge person(x) \wedge agt(\langle e,x \rangle_s)^{\$20}$
$wo(\langle c,e,x \rangle_s, c) \subset trade(e) \wedge commodity(x) \wedge obj(\langle e,x \rangle_s)^{\$2}$

Knowledge Rules

$person(x) \subset soseki(x)$
$writer(x) \subset soseki(x)$
$book(x) \subset novel(x)$
$commodity(x) \subset book(x)$
$trade(e) \subset buy(e)$
$intend(e) \subset trade(e)$

Figure 3: Example of Rules

out a particle, which permit structures like [VP[NP *Sôseki*][VP *katta*]]. Such a structure is evaluated by the pragmatic and semantic criteria. That is, the dependency between a verbal concept e and a nominal concept x is supported if there is an entity y such that x and y have a pragmatic relation, i.e., a metonymy relation, and e and y have a semantic relation, i.e., a thematic relation. The metonymy relation is defined by the pragmatic rules, based on certain knowledge, such as that the name of a writer is sometimes used to refer to his novel. Also, the thematic relation is defined by the semantic rules, based on certain knowledge, such as that the object case relation between a trading action and a commodity can be linguistically expressed as a thematic relation.

The subscript $\$c$ of a formula A represents the cost of assuming formula A. A is easy to assume when c is small, while A is difficult to assume when c is large. For instance, the cost of interpreting the thematic relation between a trading action and a commodity as the object case relation is low, say \$2, while the cost of interpreting the thematic relation between an intentional action and a third person as the agent case relation is high, say \$20. This assignment of costs is suitable for a situation in which the speaker reports his experience. In spite of the difficulty of assigning suitable costs in general, the cost-based interpretation is valuable, because it provides a uniform criteria for syntax, semantics and pragmatics. Hopefully, several techniques, independently developed in these areas, e.g., stochastic parsing, example-based/corpus-based techniques for word sense/structural disambiguation, etc., will be usable for better cost assignment. Probability will also be a key technique for the cost assignment (Charniak and Shimony, 1990).

Figure 4 and Table 1 show the chart that is created when a sentence (1) is parsed and interpreted using our algorithm. Although the diagram seems complicated, it is easy to understand if we break down the diagram. Included are the syntactic parsing of the sentence (indicated by edges 2, 6, 7, 14, 52 and 53), the pragmatic interpretation of the metonymy by *Sôseki S* (indicated by edges 17, 18, 20 and 24), the semantic interpretation of the thematic relation between a buying event B and a novel N written by *Sôseki* (indicated by edges 42, 44, 45, 47, 48 and 50), and so on. In the pragmatic interpretation, assumption $novel(N)$ (edge 21) is introduced, which is reused in the semantic interpretation. In other words, a single assumption is used more than once. Such a tricky job is naturally realized by the nature of the chart algorithm.

Agenda Control

Since the aim of cost-based abduction is to find out the *best* solution, not *all* solutions, it is reasonable to consider combining heuristic search strategies with our algorithm to find the best solution efficiently. Our algorithm facilitates such an extension by using the *agenda control mechanism*, which is broadly used in advanced chart parsing systems.

The agenda is a storage for the edges created by any of the three procedures of the chart algorithm, out of which edges to be added to the chart are selected, one by one, by a certain criterion. The simplest strategy is to select the edge which has the minimal cost at that time, i.e., *ordered search*.

Although ordered search guarantees that the first solution is the best one, it is not always very efficient. We can think of other search strategies, like *best first search*, *beam search*, etc., which are more practical than ordered search. To date, we have not investigated any of these practical search strategies. However, it is apparent that our chart algorithm, together with the agenda control mechanism, will provide a good way to realize these practical search strategies.

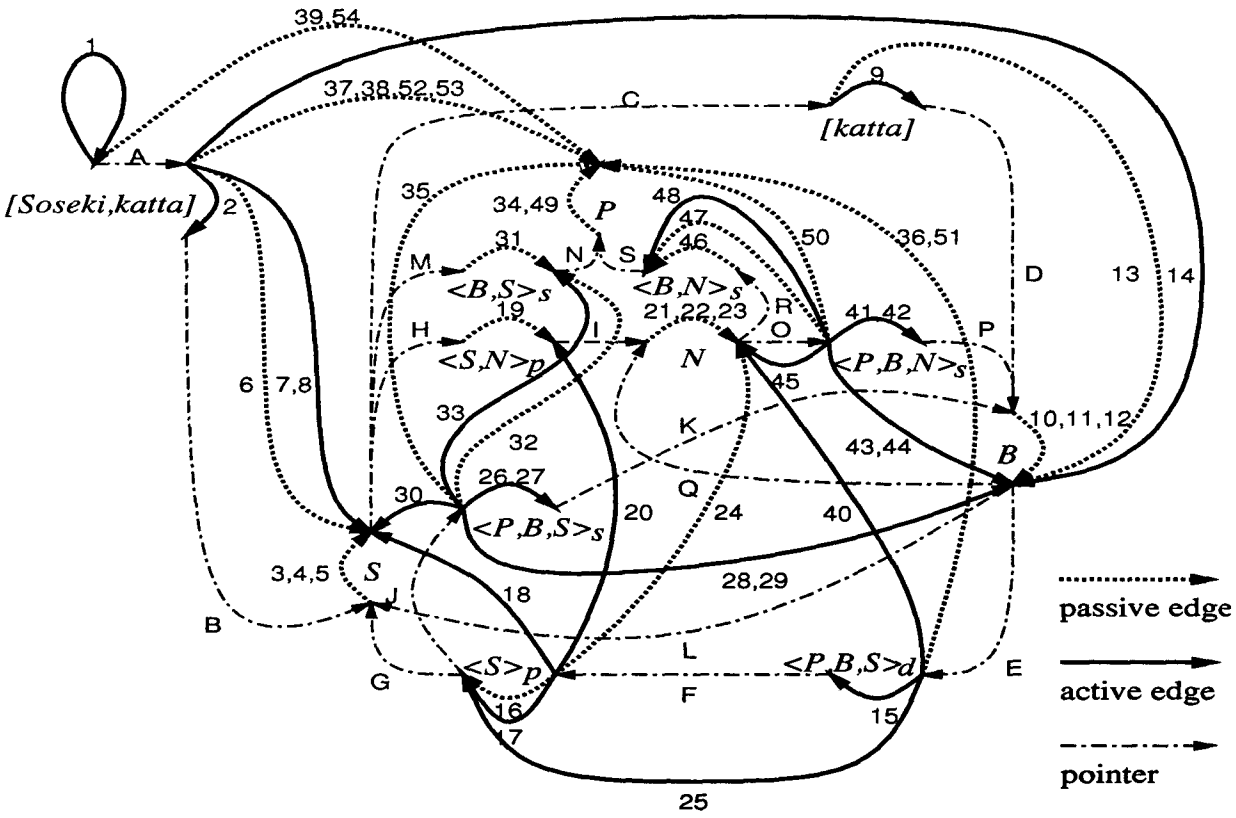

Figure 4: Chart Diagram for *Sôseki katta*

Preliminary Experiments

We conducted preliminary experiments to compare four methods of cost-based abduction: top-down algorithm (TD), head-driven algorithm (HD), generalized chart algorithm with full-search (GCF), and generalized chart algorithm with ordered search (GCO). The rules used for the experiments are in the spoken language understanding task, and they are rather small (51 chain rules + 35 non-chain rules). The test sentences include one verb and 1–4 noun phrases, e.g., sentence (1).

Table 2 shows the results. The performance of each method is measured by the number of computation steps, i.e., the number of derivation steps in TD and HD, and the number of passive and active edges in GCF and GCO. The decimals in parentheses show the ratio of the performance of each method to the performance of TD. The table clearly shows how the three features of our algorithm improve the computational efficiency. The improvement from TD to HD is due to the first feature, i.e., goal-driven bottom-up derivation, which eliminates about 50% of the computation steps; the improvement from HD to GCF is due to the second feature, i.e., tabulation of the partial results,

Table 2: Comp. among TD, HD, GCF, and GCO

Ns	TD	HD	GCF	GCO
1	215	112 (0.52)	83 (0.39)	75 (0.35)
2	432	218 (0.50)	148 (0.34)	113 (0.26)
3	654	330 (0.50)	193 (0.30)	160 (0.24)
4	876	442 (0.50)	238 (0.27)	203 (0.23)

which decreases the number of steps another 13%–23%; the improvement from GCF to GCO is due to the last feature, i.e., the agenda control mechanism, which decreases the number of steps another 4%–8%. In short, the efficiency is improved, maximally, about four times.

Comparison with Earley Deduction

We describe, here, some differences between our algorithm and Earley deduction presented by Pereira and Warren. First, as we mentioned before, our algorithm is mainly based on bottom-up (left-corner) derivation rather than top-down derivation, that Earley deduction is based on. Our experiments showed the superiority of this approach in our par-

ticular, though not farfetched, example.

Second, our algorithm does not use *subsumption-checking* of edges, which causes a serious computation problem in Earley deduction. Our algorithm needs subsumption-checking only when a new edge is introduced by the combination procedure. In the parsing of augmented grammars, even when two edges have the same nonterminal symbol, they are different in the annotated structures associated with those edges, e.g., feature structures; in such a case, we cannot use one edge in place of another. Likewise, in our algorithm, edges are always annotated by the assumption sets, which, in most cases, prevent those edges from being reused. Therefore, in this case, subsumption-checking is not effective. In our algorithm, reuse of edges only becomes possible when a new edge is introduced by the introduction procedure. However, this is done only by adding a pointer to the edge to be reused, and, to invoke this operation, *equality-checking* of lexicons, not edges, is sufficient.

Finally, our algorithm has a stronger connection with chart parsing than Earley deduction does. Pereira and Warren noted that the indexing of formulas is just an implementation technique to increase efficiency. However, indexing plays a considerable role in chart parsing, and how to index formulas in the case of proof procedures is not so obvious. In our algorithm, from the consideration of head-driven derivation, the index of a formula is determined to be the first argument of that formula. All formulas with the same index are derived the first time that index is introduced in the chart. Pointers among lexicons are also helpful in avoiding nonproductive attempts at applying the combination procedure. All the devices that were originally used in chart parsers in a restricted way are included in the formalism, not in the implementation, of our algorithm.

Concluding Remarks

In this paper, we provided a basic and practical solution to the computation problem of cost-based abduction. We explained the basic concept of our algorithm and presented the details of the algorithm along with simple examples. We also showed how our algorithm improves computational efficiency on the basis of the results of the preliminary experiments.

We are now developing an abduction-based spoken language understanding system using our algorithm. The main problem is how to find a good search strategy that can be implemented with the agenda control mechanism. We are investigating this issue using both theoretical and empirical approaches. We hope to report good results along these lines in the future.

Acknowledgments

The author would like to thank Prof. Yuji Matsumoto of Nara Institute of Science and Technology and Masahiko Haruno of NTT Communication Science Laboratories for their helpful discussions.

References

[Charniak and Husain, 1991] Eugene Charniak and Saadia Husain. A new admissible heuristic for minimal-cost proofs. *Proceedings of the 12th IJCAI*, pages 446–451, 1991.

[Charniak and Santos Jr., 1992] Eugene Charniak and Eugene Santos Jr. Dynamic MAP calculations for abduction. *Proceedings of the 10th AAAI*, pages 552–557, 1992.

[Charniak and Shimony, 1990] Eugene Charniak and Solomon E. Shimony. Probabilistic semantics for cost based abduction. *Proceedings of the 8th AAAI*, pages 106–111, 1990.

[Haruno *et al.*, 1993] Masahiko Haruno, Yasuharu Den, Yuji Matsumoto, and Makoto Nagao. Bidirectional chart generation of natural language texts. *Proceedings of the 11th AAAI*, pages 350–356, 1993.

[Hobbs *et al.*, 1988] Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. *Proceedings of the 26th Annual Meeting of ACL*, pages 95–103, 1988.

[Kay, 1980] Martin Kay. Algorithm schemata and data structures in syntactic processing. Technical Report CSL-80-12, XEROX Palo Alto Research Center, 1980.

[Pereira and Warren, 1983] Fernando C.N. Pereira and David H.D. Warren. Parsing as deduction. *Proceedings of the 21st Annual Meeting of ACL*, pages 137–144, 1983.

[Shieber *et al.*, 1989] Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C.N. Pereira. A semantic-head-driven generation algorithm for unification-based formalisms. *Proceedings of the 27th Annual Meeting of ACL*, pages 7–17, 1989.

[Sikkel and op den Akker, 1993] Klaas Sikkel and Rieks op den Akker. Predictive head-corner chart parsing. *The 3rd International Workshop on Parsing Technologies*, pages 267–276, 1993.

[van Noord, 1990] Gertjan van Noord. An overview of head-driven bottom-up generation. *Current Research in Natural Language Generation*, chapter 6, pages 141–165. Academic Press, 1990.

[van Noord, 1991] Gertjan van Noord. Head corner parsing for discontinuous constituency. *Proceedings of the 29th Annual Meeting of ACL*, pages 114–121, 1991.

Table 1: Table Representation of the Chart

#	Arc	A-Set	From
1	$[[?]s(\Phi,[],e)]\top$	ϕ	–
2	$[[?]soseki(S)^{\$1}]$ $np(\Phi,\Psi,S)$	ϕ	1
3	$soseki(S)^{\$1}$	$\{\alpha\}$	2
4	$person(S)$	$\{\alpha\}$	3
5	$writer(S)$	$\{\alpha\}$	3
6	$np(\Phi,\Psi,S)$	$\{\alpha\}$	2+B+3
7	$[np(\Phi,\Psi,S)$ $[?]vp(\Psi,k,e)$ $[?]depend(\langle c,e,S\rangle_d)]$ $vp(\Phi,k,e)$	$\{\alpha\}$	6
8	$[np(\Phi,\Psi,S)$ $[?]p(\Psi,k,c)]$ $np(\Phi,k,c,S)$	$\{\alpha\}$	6
9	$[[?]buy(B)^{\$1}]$ $vp(\Psi,[],B)$	ϕ	7
10	$buy(B)^{\$1}$	$\{\beta\}$	9
11	$trade(B)$	$\{\beta\}$	10
12	$intend(B)$	$\{\beta\}$	11
13	$vp(\Psi,[],B)$	$\{\beta\}$	9+D+10
14	$[np(\Phi,\Psi,S)$ $vp(\Psi,[],B)$ $[?]depend(\langle P,B,S\rangle_d)]$ $vp(\Phi,[],B)$	$\{\alpha,\beta\}$	7+C+13
15	$[[?]prag(\langle S\rangle_p,x)$ $[?]sem(\langle P,B,x\rangle_s)]$ $depend(\langle P,B,S\rangle_d)$	ϕ	14
16	$prag(\langle S\rangle_p,S)$	ϕ	15
17	$[[?]writer(S)$ $[?]write(\langle S,x\rangle_p)^{\$10}$ $[?]novel(x)^{\$1}]$ $prag(\langle S\rangle_p,x)$	ϕ	15
18	$[writer(S)$ $[?]write(\langle S,N\rangle_p)^{\$10}$ $[?]novel(N)^{\$1}]$ $prag(\langle S\rangle_p,N)$	$\{\alpha\}$	17+G+5
19	$write(\langle S,N\rangle_p)^{\$10}$	$\{\gamma\}$	18
20	$[writer(S)$ $write(\langle S,N\rangle_p)^{\$10}$ $[?]novel(N)^{\$1}]$ $prag(\langle S\rangle_p,N)$	$\{\alpha,\gamma\}$	18+H+19
21	$novel(N)^{\$1}$	$\{\delta\}$	20
22	$book(N)$	$\{\delta\}$	21
23	$commodity(N)$	$\{\delta\}$	22
24	$prag(\langle S\rangle_p,N)$	$\{\alpha,\gamma,\delta\}$	20+I+21
25	$[prag(\langle S\rangle_p,S)$ $[?]sem(\langle P,B,S\rangle_s)]$ $depend(\langle P,B,S\rangle_d)$	ϕ	15+F+16
26	$[[?]intend(B)$ $[?]person(S)$ $[?]agt(\langle B,S\rangle_s)^{\$20}]$ $ga(\langle P,B,S\rangle_s,P)$	ϕ	25
27	$[[?]trade(B)$ $[?]commodity(S)$ $[?]obj(\langle B,S\rangle_s)^{\$2}]$ $wo(\langle P,B,S\rangle_s,P)$	ϕ	25
28	$[intend(B)$ $[?]person(S)$ $[?]agt(\langle B,S\rangle_s)^{\$20}]$ $ga(\langle P,B,S\rangle_s,P)$	$\{\beta\}$	26+K+12
29	$[trade(B)$ $[?]commodity(S)$ $[?]obj(\langle B,S\rangle_s)^{\$2}]$ $wo(\langle P,B,S\rangle_s,P)$	$\{\beta\}$	27+K+11
30	$[intend(B)$ $person(S)$ $[?]agt(\langle B,S\rangle_s)^{\$20}]$ $ga(\langle P,B,S\rangle_s,P)$	$\{\alpha,\beta\}$	28+L+4
31	$agt(\langle B,S\rangle_s)^{\$20}$	$\{\epsilon\}$	30
32	$ga(\langle P,B,S\rangle_s,P)$	$\{\alpha,\beta,\epsilon\}$	30+M+31
33	$[ga(\langle P,B,S\rangle_s,P)$ $[?]ga(P)^{\$3}]$ $sem(\langle P,B,S\rangle_s)$	$\{\alpha,\beta,\epsilon\}$	32
34	$ga(P)^{\$3}$	$\{\zeta\}$	33
35	$sem(\langle P,B,S\rangle_s)$	$\{\alpha,\beta,\epsilon,\zeta\}$	33+N+34
36	$depend(\langle P,B,S\rangle_d)$	$\{\alpha,\beta,\epsilon,\zeta\}$	25+J+35
37	$vp(\Phi,[],B)$	$\{\alpha,\beta,\epsilon,\zeta\}$	14+E+36
38	$s(\Phi,[],B)$	$\{\alpha,\beta,\epsilon,\zeta\}$	37
39	$\top$	$\{\alpha,\beta,\epsilon,\zeta\}$	1+A+38
40	$[prag(\langle S\rangle_p,N)$ $[?]sem(\langle P,B,N\rangle_s)]$ $depend(\langle P,B,S\rangle_d)$	$\{\alpha,\gamma,\delta\}$	15+F+24
41	$[[?]intend(B)$ $[?]person(N)$ $[?]agt(\langle B,N\rangle_s)^{\$20}]$ $ga(\langle P,B,N\rangle_s,P))$	ϕ	40
42	$[[?]trade(B)$ $[?]commodity(N)$ $[?]obj(\langle B,N\rangle_s)^{\$2}]$ $wo(\langle P,B,N\rangle_s,P)$	ϕ	40
43	$[intend(B)$ $[?]person(N)$ $[?]agt(\langle B,N\rangle_s)^{\$20}]$ $ga(\langle P,B,N\rangle_s,P)$	$\{\beta\}$	41+P+12
44	$[trade(B)$ $[?]commodity(N)$ $[?]obj(\langle B,N\rangle_s)^{\$2}]$ $wo(\langle P,B,N\rangle_s,P)$	$\{\beta\}$	42+P+11
45	$[trade(B)$ $commodity(N)$ $[?]obj(\langle B,N\rangle_s)^{\$2}]$ $wo(\langle P,B,N\rangle_s,P)$	$\{\beta,\delta\}$	44+Q+23
46	$obj(\langle B,N\rangle_s)^{\$2}$	$\{\eta\}$	45
47	$wo(\langle P,B,N\rangle_s,P)$	$\{\beta,\delta,\eta\}$	45+R+46
48	$[wo(\langle P,B,N\rangle_s,P)$ $[?]wo(P)^{\$3}]$ $sem(\langle P,B,N\rangle_s)$	$\{\beta,\delta,\eta\}$	47
49	$wo(P)^{\$3}$	$\{\theta\}$	48
50	$sem(\langle P,B,N\rangle_s)$	$\{\beta,\delta,\eta,\theta\}$	48+S+49
51	$depend(\langle P,B,S\rangle_d)$	$\{\alpha,\beta,\gamma,\delta,\eta,\theta\}$	40+O+50
52	$vp(\Phi,[],B)$	$\{\alpha,\beta,\gamma,\delta,\eta,\theta\}$	14+E+51
53	$s(\Phi,[],B)$	$\{\alpha,\beta,\gamma,\delta,\eta,\theta\}$	52
54	$\top$	$\{\alpha,\beta,\gamma,\delta,\eta,\theta\}$	1+A+53

$\Phi = [S\hat{o}seki,katta]$, $\Psi = [katta]$

$\alpha = soseki(S)^{\$1}$, $\beta = buy(B)^{\$1}$, $\gamma = write(\langle S,N\rangle_p)^{\$10}$, $\delta = novel(N)^{\$1}$,
$\epsilon = agt(\langle B,S\rangle_s)^{\$20}$, $\zeta = ga(P)^{\$3}$, $\eta = obj(\langle B,N\rangle_s)^{\$2}$, $\theta = wo(P)^{\$3}$