LiTH-IDA-R-88-11

ISSN-0281-4250

INSTITUTIONEN FÖR DATAVETENSKAP
UNIVERSITETET OCH TEKNISKA HÖGSKOLAN
LINKÖPING

An Incremental Chart Parser for PATR

by

Mats Wirén

RESEARCH REPORT
NLPLAB, May 1988

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

Computer and Information Science Department
Linköping University

PhD theses:

* Available at: University of Microfilms Intl., 300 N. Zeeb Road, Ann Arbor, MI 48106, USA.

(*Linköping Studies in Science and Technology. Dissertations.*)

*No 77	**Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982.
No 94	**Hans Lunell:** Code Generator Writing Systems, 1983.
No 97	**Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983.
*No 109	**Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation, 1984.
*No 111	**Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984.
*No 155	**Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons. 1987.
No 165	**James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning. 1987.
No 170	**Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems. 1987.

Licentiate of engineering Theses:

(*Linköping Studies in Science and Technology. Theses.*)

No 48	**Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching. 1985.
No 52	**Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems. 1985.
No 60	**Johan Fagerström:** Simulation and Evaluation of an Architecture based on Asynchronous Processes. 1985.
No 71	**Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System. 1987.
No 72	**Tony Larsson:** On the Specification and Verification of VLSI Systems. 1986.
No 73	**Ola Strömfors:** A Structure Editor for Documents and Programs. 1986.
No 74	**Christos Levcopoulos:** New Results about the Approximation Behaviour of the Greedy Triangulation. 1986.
No 104	**Shamsul I. Chowdhury:** Statistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology. 1987.
No 108	**Rober Bilos:** Incremental Scanning and Token-based Editing. 1987
No 111	**Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments. 1987.
No 113	**Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge. 1987.
No 118	**Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs. 1987
No 126	**Dan Strömberg:** Transfer and Distribution of Application Programs. 1987
No 127	**Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems. 1987
No 139	**Christer Bäckström:** Reasoning about Interdependent Actions. 1988.
No 140	**Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing. 1988.

LiTH-IDA-R-88-11 ISSN-0281-4250

An Incremental Chart Parser for PATR

by

Mats Wirén

This paper describes a technique for incremental chart parsing under a unification-based grammatical formalism, PATR, and outlines how this fits into continued work aimed at developing a parsing system which is both interactive and incremental. Incremental parsing means that input is analysed in a piecemeal fashion, in particular to allow arbitrary changes of previous input without having to reparse more than necessary. Interactive parsing means that the analysis is performed while the input is being entered (e.g. typed) to the system. The combination of these techniques could be used as a kernel for highly interactive and "reactive" natural-language processors, such as parsers for dialogue systems and language-sensitive text editors.

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

An Incremental Chart Parser for PATR

Mats Wirén
Department of Computer and Information Science
Linköping University
S-581 83 Linköping, Sweden

Abstract

This paper describes a technique for incremental chart parsing under a unification-based grammatical formalism, PATR, and outlines how this fits into continued work aimed at developing a parsing system which is both interactive and incremental. Incremental parsing means that input is analysed in a piecemeal fashion, in particular to allow arbitrary changes of previous input without having to reparse more than necessary. Interactive parsing means that the analysis is performed while the input is being entered (e.g. typed) to the system. The combination of these techniques could be used as a kernel for highly interactive and "reactive" natural-language processors, such as parsers for dialogue systems and language-sensitive text editors.

1 Introduction

1.1 The Problem

Ideally, a parser for an interactive natural-language processing system ought to operate in real time in such a way that the system accomplishes an analysis of an utterance while it is being entered to the system. Furthermore, should the user make a small change in the utterance, the system ought to be able to revise its analysis within a correspondingly small amount of time. In addition, it would be desirable if the system was able to interact dynamically with the user in order to resolve problems that may occur during the parsing process, for example asking for clarifications, promptly react to misspellings and other errors, etc.

In practice, a necessary (but not sufficient) condition for these desiderata is an *interactive, incremental parsing system.* Incremental parsing roughly means that input is analysed in a piecemeal fashion, in particular to be able to handle arbitrary modifications of previous input without reparsing more of it than necessary. Interactive parsing means that the analysis is performed while the input is being entered (e.g. typed) to the system, and possibly also that the system may interact dynamically with the user in order to resolve problems that occur during the parsing process.

This paper describes a first step towards realizing a system of this kind, viz. a technique for incremental parsing. For reasons to be discussed in section 2, the technique is grounded in chart parsing (Kay 1982). The technique will be developed from the perspective of a general context-free- and unification-based grammar formalism, PATR.[1] A preliminary parsing system embodying the ideas put forward in this paper has been built on top of M-PATR, a control-strategy-independent chart parser for PATR (Wirén 1988). Since the framework of chart parsing is independent of particular grammatical formalisms, it is possible to adapt the technique of incrementality to chart-based parsers for other grammar formalisms as well.

1.2 Definitions of Key Concepts

The word "incremental" is actually used in two differing senses in the literature. The first sense stresses that input should be processed in a *piecemeal* fashion. According to this view, an incremental problem solver successively generates and integrates partial solutions to form larger solutions based on current data and knowledge (Erman et al. 1980); an incremental semantic interpreter constructs a meaning representation of an utterance bit by bit, rather than in one go when it has come to an end (Bobrow and Webber 1980, Mellish 1985, Hirst 1987, Haddock 1987); and, similarly, an incremental parser constructs an analysis by successively generating new partial analyses and combining old ones into larger composites based on current input and grammar. Sometimes the order according to which new chunks of input are handled is constrained in some way, for example in such a way that a parser only may "increment" a sentence analysis from left to right. Specif-

This research has been supported by the National Swedish Board for Technical Development. An experimental system has been implemented on machines donated by the Xerox Corporation through their University Grants Program.

[1] Throughout this paper, "PATR" will be referring to PATR-II (Shieber et al. 1983).

ically, chart parsing is a technique which does *not* impose any such constraints upon processing, while still embodying the above notion of incrementality.

The other sense of "incremental" stresses the necessity of also handling arbitrary *changes* within the current set of data. Thus, according to this view, an incremental parser should be able to handle not only piecemeal extensions to an input sentence, but also deletions and insertions inside of it.[2] In other words, partial analyses should not only be generated and integrated, but might also be retracted. This view of incremental parsing is prevalent within programming-environments research and can be exemplified by Ghezzi and Mandrioli (1979, 1980). In order to distinguish between these two notions of incrementality, I will refer to them as *weak* and *strong* incrementality, respectively. When leaving out the qualification, I will in general be referring to strong incrementality.

Clearly, these two concepts are related to monotonicity and nonmonotonicity. For example, chart parsing is monotonic in the sense of strictly increasing the number of chart edges as the analysis proceeds; it is also weakly incremental in handling arbitrary insertions to the left and right of previous input without reparsing. On the other hand, in order to handle arbitrary input changes (strong incrementality), edges sometimes have to be removed from the chart (something which in turn forces other edges to be removed, and so on). We then arrive at a notion of parsing resembling reason maintenance within the field of nonmonotonic reasoning; cf. for example ATMS (de Kleer 1986).

At this point, it might be useful to pin down concise working definitions of the key concepts of this paper.

Incremental parser. A parser capable of handling changes of previous input while expending an amount of effort which is proportional to the size of the changes.[3]

This seems to be the most concise definition from which all the crucial properties follow: arbitrary changes should be handled; the analysis should proceed in a piecemeal fashion; and the amount of reparsing should be minimal according to some measure.

In general, parsing could be seen as a mapping from a sentence to a set of parse trees, or, in this case, a chart representing (among other things) all complete parse trees.[4] Incremental parsing requires a more complex mapping

$$F(\eta, \kappa, \tau, c_0) \rightarrow c_1$$

from an edit operation η, a pair of cursor positions κ, a piece of text (i.e., a stream of characters) τ, and an initial chart c_0 to a new chart c_1 (cf. Fritzson 1985). Within the scope of this paper we are going to take a slightly simplified approach, assuming that we only deal with the primitive update operations *insert* and *delete*, each of which is applied to a sequence of words $w_l \cdots w_r$.

A related concept is interactive parsing (sometimes called on-line parsing).

Interactive parser. (Synonym: *on-line parser.*) A parser which monitors a text-input process, starting to parse immediately upon the onset of new input, thereby achieving enhanced efficiency as well as a potential for dynamic improvement of its performance, for example by promptly reporting errors, asking for clarifications, etc.

Strictly speaking, incrementality and interactivity are two independent properties — neither one presupposes the other. However, an incremental system that was not interactive would be pointless, and an interactive system that was not incremental would at least not be efficient (but such systems do exist; cf. Tomita's system in section 2). Thus, combining these two properties is indeed a natural thing to do.

A further related concept, mentioned at the outset of section 1, is real-time parsing.

Real-time parser. A parser which analyses input in real time, i.e., at the speed of production.

Real-time parsing is related to the aforementioned concepts in the following way: If a system is to perform language processing in real-time situations, it should be interactive; furthermore, in order to achieve the necessary efficiency, it should be incremental. I will, however, not have anything more to say about real-time parsing here.

2 Previous Work and a Novel Approach

The problem of incremental parsing was first stated within programming-environments research by Lindstrom (1970) and Earley and Caizergues (1972). Within this area, incremental parsing is used for applications such as incremental compilation, for example the Rational Ada-programming environment (Dart et al. 1987), and syntax-directed text editing, like the Synthesizer Generator (Reps and Teitelbaum 1987).

[2] An additional requirement is that this should be done in some reasonably efficient way. Cf. the definition below.

[3] This definition is formed partly in analogy with a definition of "incremental compilation" by Earley and Caizergues (1972:1040).

[4] In PATR, dags (directed acyclic graphs) as well as trees.

Detailed discussions about incremental parsing algorithms for deterministic context-free languages are reported in Ghezzi and Mandrioli (1979, 1980). Of course, these algorithms are not readily applicable to the processing of a nondeterministic, complex-feature-based formalism such as PATR.

Within natural-language processing, Tomita (1985, 1986, 1987) has reported a parser which parses interactively, but, incidentally, not incrementally according to the definition given above. The reason for this is that it does not handle arbitrary changes in an efficient way. The system is based directly on an LR-algorithm, and thus parses strictly from left to right as the user types something in. The user may delete input from right to left, in which case the system "unparses" this input. This means that, if the user wants to update only a small fragment in the beginning of a sentence, the system has to reparse everything from this update and on (of course, the user is forced to actually delete and retype everything from the change point).

This paper develops a novel approach to incremental natural-language parsing based on chart parsing.[5] As stated above, the aim of incremental parsing is to handle arbitrary changes ("editing operations") with minimal reparsing. In other words, only the portion of the previous analysis which is affected by a change should be recomputed. In order to determine what this minimal portion is, a data structure is maintained for suitably recording the information derived during a (partial) parse in order to shorten the analysis of the changed input.

The basic idea of this work is to make use of the chart as our record of partial analyses, and to maintain explicit dependencies between chart edges in order to propagate the effects of an update precisely to those chart edges that somehow depend on the directly affected edges. The advantages of grounding an incremental parser in a chart-parsing framework can be summarized as follows:

- chart parsing is an open-ended, well understood, and frequently used technique in natural-language processing;
- ordinary chart parsing is itself weakly incremental; thus it provides a good starting point for developing strong incrementality;
- it is easy to provide direct access to all partial analyses (edges) on the chart, and, specifically, to provide a mapping from the words of a sentence to (preterminal) chart edges;
- dependencies between edges can be maintained in a straightforward way.

[5]For references to chart parsing, see Kay (1982), Thompson (1981), and Thompson and Ritchie (1984).

3 Incremental Chart Parsing

3.1 PATR and the M-PATR System

This paper develops an incremental chart-parsing machinery from the perspective of a general context-free- and unification-based grammatical formalism, PATR. A preliminary parsing system that realizes the ideas put forward here has been built on top of M-PATR, a control-strategy-independent chart parser for PATR. For more information about M-PATR as well as a concise introductory to PATR (including the notation used), see Wirén (1988). Since the method of chart parsing is independent of particular grammatical formalisms, it is possible to adapt this technique of incrementality to chart-based parsers for other grammar formalisms as well.

3.2 Maintaining Chart Updates through Edge Dependencies

The basic approach of this work is to maintain updates of the chart through explicit dependencies between chart edges. Loosely speaking, an edge can be said to depend upon another edge if it is formed somehow using this original edge. Thus, an edge formed through a predict operation could be said to depend on the (one) edge that triggered it.[6] Likewise, an edge formed through a combine operation depends on the active–inactive pair of edges that generated it under the fundamental rule.[7] A scanned edge clearly does not depend upon any other edge (scanning can be seen as a kind of initialization of the chart).

In order to derive edge dependencies one may associate with each edge its set of source edges, or "back pointers". This information can be used to derive the corresponding sets of "forward pointers" or dependants that we are interested in: Let $\mathcal{D}$ be a binary dependency relation such that $e\,\mathcal{D}\,e'$ if and only if e' is a dependant of e, i.e., e' has been formed (directly) using e. Furthermore, let $\mathcal{D}^*$ be the transitive closure of $\mathcal{D}$. Then $e\,\mathcal{D}^*\,f$ holds if and only if there is a chain of dependencies from e to f. Thus, given that an edge e is to be removed from the chart, all edges f for which $e\,\mathcal{D}^*\,f$ holds should also be removed.

[6]It is possible to call the predict function separately, for example when generating an initial top-down hypothesis. In this case, the source could be said either to be non-existent or, perhaps, to consist of the set of preterminal edges extending from the initial vertex.

[7]"The fundamental rule" of chart parsing refers to the basic conditions according to which a combined edge is formed (Thompson 1981:2).

3.3 Technical Preliminaries

3.3.1 The Chart

A *chart* is a directed graph. The nodes, or *vertices*, $v_1, \ldots, v_{n+1}$ correspond to the positions surrounding the words of an n-word sentence $w_1 \cdots w_n$. A pair of vertices v_i, v_j may be connected by arcs, or *edges*, bearing information about (partially) analysed constituents between v_i and v_j. We will take an edge to be a tuple

$$\langle s, t, X_0 \rightarrow \alpha.\beta, D, P \rangle$$

starting from vertex v_s and ending at vertex v_t with dotted rule $X_0 \rightarrow \alpha.\beta$[8], a dag D representing the rule according to PATR, and, finally, the set of immediately dependent edges (forward pointers), P.[9] We will take a vertex to be a tuple $\langle I, O \rangle$ where I is the set of incoming edges to the vertex and O is the set of outgoing edges from the vertex.

We will use dots to qualify elements of tuples. For example, $e.s$ will denote the starting vertex of edge e. Likewise, $v_i.I$ will denote the set of incoming edges to vertex number i.

Note that vertices and edges are coindexed in such a way that from a given vertex we may refer to its set of incoming and outgoing edges, and from a given edge we may refer to the pair of vertices it connects. Thus, when updating the chart, we must consider both the relevant vertices and edges.

3.3.2 A Simplified Mapping

Recall that, in general, incremental parsing can be seen as a mapping $F(\eta, \kappa, \tau, c_0) \rightarrow c_1$ from an edit operation η, a pair of cursor positions κ, a piece of text τ, and an initial chart c_0 to a new chart c_1. Within the scope of this paper, we are going to take a slightly simplified approach, making the following assumptions:

- only the two edit operations *insert* and *delete* are allowed (i.e., we can assume that a higher-level component has broken down more complex operations into suitable sequences of these primitive ones);
- edit operations only apply at the word level (i.e., at this stage we do not consider character editing);
- the (primitive) edit operations are "compact", i.e., each of them only affects a compact interval of the input (and the vertices);
- each *insert* or *delete* operation affects words $w_l \cdots w_r$ each of which maps to one or several preterminal edges extending from vertices $v_l, \ldots, v_r$, respectively.

This means that η may take only the values *insert* or *delete*, κ is a pair of positions l, r such that the sequence of positions $l, \ldots, r$ map directly to vertices $v_l, \ldots, v_r$, and, finally, τ is the corresponding sequence of words $w_l \cdots w_r$.

Below, we will make use of the constant $\delta = r - l + 1$ denoting the number of words that have been inserted or deleted as the result of an edit operation.

3.4 An Algorithm for Incremental Chart Parsing

3.4.1 Introduction

This section states an algorithm for incremental chart parsing[10] under the PATR formalism. More specifically, the algorithm updates the chart in accordance with *one* (primitive) editing operation; hence, the algorithm should be iterated for each such operation.[11]

The algorithm works according to four mutually exclusive cases which occur depending on the position in the input at which the update took place. The algorithm is grounded in a "standard" chart parser for PATR which is specified in section 3.4.3. In addition, subroutines for adding and removing chart edges are given in section 3.4.4.

3.4.2 Incremental Chart-Parsing Algorithm

Input: A grammar G,[12] an edit operation η, a pair of vertex numbers l, r, a sequence of words $w_l \cdots w_r$, and a chart c_0. We assume that chart c_0 consists of vertices $v_1, \ldots, v_{n+1}$.[13]

Output: A chart c_1.

Method: Select and execute one of the following four cases:

Case 1: Insertion at right end of c_0

for $i := l, \ldots, r$ **do Scan**(w_i);
$n := n + \delta$.

[8] A dotted rule $X_0 \rightarrow \alpha.\beta$ corresponds to an (active) X_0 edge containing an analysis of constituent(s) α, looking for constituent(s) β in order to yield a complete (inactive) edge.

[9] In other words, the set P of an edge e consists of all edges e' for which $e\,D\,e'$ holds.

[10] The notation and terminology chosen here is freely borrowed from the chart-parsing literature as well as from statements of Earley's algorithm (Aho and Ullman 1972:321).

[11] In practice, this might not be the most efficient strategy; cf. section 4.1.

[12] For specification of a PATR grammar, see Wirén (1988).

[13] For simplicity, we fix the rule-invocation strategy to top-down (cf. section 3.4.3). We assume that the chart has been accordingly initialized with an edge $\langle 1, 1, X_0 \rightarrow .\alpha, D, \emptyset \rangle$ for each rule $\langle X_0 \rightarrow \alpha, D \rangle$ such that $D(\langle X_0\ cat \rangle) = S$, where S is the start symbol of the grammar.

This case occurs when δ words $w_l \cdots w_r$ have been inserted to the right of the previous input (i.e., $l = n + 1$). This kind of insertion only requires weak incrementality, and is handled simply by scanning the new input, extending the original chart c_0 δ steps to the right.[14]

Case 2: Deletion at right end of c_0

for $i := l, \ldots, r$ **do**
  $\forall e$: $e \in v_i.O \wedge preterminal(e)$
    RemoveEdgeChain(e);
$n := n - \delta$.

This case occurs when δ words $w_l \cdots w_r$ have been deleted at the right end of the previous input (i.e., $r = n$). It is handled by removing the preterminal edges corresponding to each deleted word along with their entire chains of dependent edges.

Case 3: Deletion within c_0

(* Remove the deleted edges *)
for $i := l, \ldots, r$ **do**
  $\forall e$: $e \in v_i.O \wedge preterminal(e)$
    RemoveEdgeChain(e);
(* Adjust the chart to the left *)
(* First the edges *)
for $i := r + 1$ **to** n **do**
  $\forall e$: $e \in v_i.O$
    $e.s := e.s - \delta$;
    $e.t := e.t - \delta$;
(* Then the vertices *)
$v_l.O := v_{r+1}.O$;
for $i := l + 1$ **to** $n + 1 - \delta$ **do** $v_i := v_{i+\delta}$;
for $i := n + 2 - \delta$ **to** $n + 1$ **do** $v_i := \emptyset$;
$n := n - \delta$.

This case occurs when δ words $w_l \cdots w_r$ have been deleted from an interval *within* previous input (i.e., $l > 1$ and $r < n$). It is handled by deleting each preterminal edge along with its chain of dependent edges, and then collapsing the chart, moving all edges from vertex v_{r+1} and on δ steps to the left.

Case 4: Insertion within c_0

(* Remove all edges that "cross" the change vertex *)
$\forall e \forall f \forall g$:
  $f \in v_l.I \wedge preterminal(f)$
  $g \in v_l.O \wedge preterminal(g)$
  $e \in f.P \cap g.P$
    RemoveEdgeChain(e);
(* Adjust the chart to the right *)
(* First the vertices *)
for $i := n + 1$ **downto** $l + 1$ **do** $v_{i+\delta} := v_i$;
$v_{r+1}.O := v_l.O$;
for $i := l + 1$ **to** r **do** $v_i := \emptyset$;
$v_l.O := \emptyset$;
$v_{r+1}.I := \emptyset$;
(* Then the edges *)
$n := n + \delta$;
for $i := r + 1$ **to** n **do**
  $\forall e$: $e \in v_i.O$
    $e.s := e.s + \delta$;
    $e.t := e.t + \delta$;
(* Scan the new input *)
for $i := l, \ldots, r$ **do Scan**(w_i).

This case occurs when δ words $w_l \cdots w_r$ have been inserted at a position *within* previous input (i.e., $1 < l < n + 1$). It is handled by first removing all edges that "cross" the change vertex v_l (the vertex at which the new insertion is about to start). This is done by removing every edge (along with dependants) which is in the dependency chain of *both* some preterminal edge incident to the change vertex and some preterminal edge extending from it. (Note that in this case the preterminal edges themselves should not be removed.) Secondly, the chart is split at vertex v_l and all edges extending from this vertex or some vertex to the right of it are moved δ steps to the right. Finally, the new input is scanned and the corresponding edges inserted into the chart.

3.4.3 Standard Chart-Parsing Operations

This section specifies a chart parser (for PATR) upon which the algorithm of the preceding section is grounded. Only the operation **Scan** is explicitly "called" from the algorithm of the preceding section. It is assumed that **Predict** and **Combine** are appropriately triggered by the addition of new edges to the chart (and that these actions can be scheduled through an agenda mechanism). For simplicity, we only give the top-down (Earley-style) predictor. Alternative PATR predictors can be found in Wirén (1988).

Scan(w_i):

If $w_i = a$, then for all lexical entries of the form $\langle X_0 \rightarrow a, D\rangle$, **AddEdge**$(\langle i, i+1, X_0 \rightarrow a., D, \emptyset\rangle)$.[15]

Informally, this means adding an inactive, preterminal edge for each word sense of the word.

Predict:

For each edge e ending at v_j of the form $\langle i, j, X_0 \rightarrow \alpha.X_m\beta, D, P\rangle$ and each rule of the

[14] The words could be scanned in any order, thus not necessarily in the order indicated above.

[15] For simplicity, we presuppose that preterminal edges only extend between adjacent vertices.

form $\langle Y_0 \rightarrow \gamma, E\rangle$ such that $E(\langle Y_0\ cat\rangle) = D(\langle X_m\ cat\rangle)$, **AddEdge**$(\langle j, j, Y_0 \rightarrow .\gamma, E, \{e\}\rangle)$ if it is not subsumed[16] by another edge.

Informally, this means predicting an edge according to each rule whose left-hand-side category matches the category being looked for by the active edge under consideration.

Combine:

For each edge e_a of the form $\langle i, j, X_0 \rightarrow \alpha.X_m\beta, D, P_a\rangle$ and each edge e_i of the form $\langle j, k, Y_0 \rightarrow \gamma., E, P_i\rangle$, **AddEdge**$(\langle i, k, X_0 \rightarrow \alpha X_m.\beta, D \sqcup [X_m\colon E(Y_0)], \{e_a, e_i\}\rangle)$ if the unification succeeds and this edge is not subsumed by another edge.

Informally, this means forming a new edge whenever the category of the first needed constituent of an active edge matches the category of an inactive edge,[17] and the dag of the inactive edge can be unified in with the dag of the needed constituent.

3.4.4 Add and Remove Operations

This section specifies the operations needed for adding and removing chart edges. We assume that E is the set of all edges currently on the chart.

AddEdge(e):

$v_{e.s}.O := v_{e.s}.O \cup \{e\}$;
$v_{e.t}.I := v_{e.t}.I \cup \{e\}$;
$E := E \cup \{e\}$.

RemoveEdgeChain(e):

$\forall e'\colon e' \in e.P \wedge exists(e')$
  RemoveEdgeChain(e');
RemoveEdge(e).

For a given edge e, this function first incrementally removes all edges for which $e\ \mathcal{D}^*\ e'$ holds and then removes e itself. The "exists" predicate tests whether the edge still exists on the chart — a deletion along some other dependency chain might already have removed it.

RemoveEdge(e):

$v_{e.s}.O := v_{e.s}.O \setminus \{e\}$;
$v_{e.t}.I := v_{e.t}.I \setminus \{e\}$;
$E := E \setminus \{e\}$.

[16] One edge subsumes another edge if and only if the first three elements of the edges are identical and the fourth element of the first edge subsumes that of the second edge.

[17] Note that this condition is tested by the unification which specifically ensures that $D(\langle X_m\ cat\rangle) = E(\langle Y_0\ cat\rangle)$.

4 Discussion

4.1 Efficiency Considerations

The purpose of the algorithm given above is to provide a first step towards the goal of an interactive, incremental parsing system by giving an algorithm for how to maintain a chart which is consistent with respect to primitive updates of the input. In giving the algorithm, we have not specifically considered efficiency aspects. This section discusses some issues related to this.

4.1.1 Fine-Tuning Parsing Operations

When putting together a fully operational (interactive) system, it will hardly be practical merely to iterate the present algorithm in order to account for repeated editing operations or to simulate more complex editing operations. For example, a deletion of some words in the middle of a sentence may well be the first part of a replacement, in which case it is not sensible to collapse the chart, combining two pieces which may shortly have to be split again. In this situation we might need an alternative incremental parsing operation which, instead of outputting *one* chart, outputs two disconnected pieces, trying to combine them only if the user remains silent a certain amount of time. Another possibility would be to evaluate both possibilities (connect–disconnect) in parallel while waiting for some definite information from the user. However, efficiency judgements like these should be empirically motivated, and therefore have to wait until we have a more definitive (fully interactive) system.

4.1.2 Recording Back Pointers Only

Henry Thompson (personal communication 1988) has pointed out that, instead of both recording source edges (back pointers) and computing sets of dependants (forward pointers) from these, it might actually suffice to record only the back pointers, provided that the frequency of updates is small and the total number of edges is not too large. The idea is to sweep the whole edge space each time there is an update, repeatedly deleting anything with a back pointer to a non-existent edge, and iterating until one gets through a whole pass with no new deletions.

4.1.3 Keeping Top-Down Predictions

Previously, we have assumed that, when removing an edge, all of its dependants should be removed along with it. In the case of edges generated through top-down predictions, this appears not always to be necessary, or desirable. The reason for this is that,

since a top-down prediction is made at the ending vertex of the triggering edge, its secondary (dependent) edges will extend outside of this edge. Bottom-up predictions, on the other hand, are made at the starting vertex of the triggering edge, which means that their secondary edges will eventually cover this edge. Suppose that we are parsing top-down and that an update has occurred within an *NP* constituent. Clearly, it is then not necessary to remove the top-down *VP* predictions that are to be found at the end of the *NP* edge (and which may in fact already have generated legal analyses). Thus, a good heuristic is to refrain from removing top-down predictions.[18]

4.1.4 LR Parsing

Tomita (1985, 1986) has developed a generalized LR parser that handles full context-free languages. In subsequent work (Tomita 1987) he has also generalized this parser to handle a complex-feature-based formalism.[19] Given that input is typically entered from left to right, a possible avenue for future work would be to try to take advantage of the efficiency exibited by this kind of generalized LR parsing while retaining incrementality; thus, in a sense, combining the work of Ghezzi and Mandrioli (1979, 1980) with the work of Tomita.

4.2 Lexical Component

Thus far we have regarded the lexical component of the system as a black box. This section discusses some issues related to this.

An interactive, incremental parsing system requires an interactive lexicon. While the preliminary system uses a primitive full-form lexicon, the planned approach is to make use of a lexical component which stores words in a "letter tree", analysing words from left to right. This technique takes advantage of the fact that the normal way of inputting words is by entering characters from left to right. By dynamically maintaining a pointer (for each word under input) into the current position in the dictionary tree, one may perform an interactive (and, to some extent, incremental) lookup. A lexical–morphological theory which accords with these principles, and which is tentatively planned to be used in the full-scale implementation, is Koskenniemi's two-level model (Koskenniemi 1983 a, 1983 b). Also the UCP parser (Sågvall Hein 1987) accords with these principles. In addition, UCP employs chart parsing both on the character level and the word level (using one and the same chart), thereby in part doing away with the clear distinction between syntactic and lexical information which may pose problems for languages like Swedish.

An interesting question is whether the kind of incrementality that we have developed at the *inter-word* level (syntactic, and possibly semantic, level) could also be applied at the *intra-word* level (the morphological level). Differently put, could every (possible) morpheme within a word be separately analysed so that, if an update occurs within the word, only the analyses of the updated morpheme(s) and their dependent analyses are recomputed? If we take morphologically moderately complex languages like Swedish, English, or German, considering inflectional morphology and derivational morphology, this should be possible to a relatively high degree. On the other hand, with respect to compounding, languages like Swedish and German in which words are "smashed" pose severe problems for an approach like this because the number of possible morphemes within a word in general becomes exceedingly great. In addition, one would ultimately fail at the semantic level since compound word meanings in general cannot be constructed compositionally.

4.3 Extra-Linguistic Constraints

The present parsing machinery is limited to developing analyses only considering constraints that are derived based on the grammar and the input sentence. An interesting path for future research would be to consider also constraints imposed from outside of the linguistic input, such as contextual constraints. For example, a common phenomenon in (spoken) Swedish is that an utterance formed as an assertion may be grammatically ambiguous between assertion and question. In order to resolve this, one might rely on a contextual constraint forcing for example the utterance to be interpreted as a question. This should then result in retracting (but not necessarily deleting) the analysis corresponding to assertion, and preferably also those sources of it that do not support any other (valid) analyses. Since full source information is available, such a requirement does not pose a problem. The additional requirements that extra-linguistic constraints would put on the kind of system developed in this paper could be summarized as follows:

- in general, extra-linguistic constraints should not result in deletion of edges, but rather label them *out* (i.e., not consider them during the continued processing) since another external

[18] A top-down prediction which is a dependant of a removed edge still is less "certain" than other edges. A possible way of reflecting this, while still keeping it, is to put it to sleep (cf. Stock 1986:214).

[19] The formalism handled by his system is however in one sense less expressive than PATR since the informational domain is limited to tree-structured graphs instead of directed acyclic graphs.

constraint might later invalidate the labelling;

- extra-linguistic constraints might invalidate *any* edge, whereas previously only preterminal edges have been directly amenable for retraction;
- invalidation of an edge through an extra-linguistic constraint should result in invalidation of those (direct and indirect) *sources* of it that are not supporters of any other (valid) analyses;
- in order to properly handle these kinds of constraints, possibly also between (constituent-level) elements of multiple sentences, we may ultimately need a reason-maintenance system.

5 Conclusion

This paper has described a technique for incremental chart parsing; more specifically, a technique for efficiently keeping a chart consistent with respect to arbitrary insertions and deletions of words within an input sentence. Chart parsing is particularly useful in this context because it appears to be highly suitable to the needs of incremental parsing while at the same time constituting an open-ended, well understood, and widely used technique in natural-language processing. Continued work along these lines aims at developing an interactive, incremental parsing system. Such a system could in turn be used as a kernel for highly interactive and reactive natural-language processors, such as parsers for dialogue systems, parsers for interactive computer-aided translation systems, language-sensitive text editors, etc.

Acknowledgements

I would like to thank Lars Ahrenberg for fruitful discussions and helpful comments on several drafts of this paper. I also want to thank Nils Dahlbäck, Arne Jönsson, Magnus Merkel, and Bernt Nilsson for rewarding discussions on the topics of this paper. I am indebted to Henry Thompson for some valuable guidance exchanged over the net, and to Peter Fritzson and Bengt Lennartsson for giving me pointers to related work within programming-environments research.

References

Aho, Alfred V. and Jeffrey D. Ullman (1972). *The Theory of Parsing, Translation, and Compiling. Volume I: Parsing.* Prentice-Hall, Englewood Cliffs, New Jersey.

Bobrow, Robert J. and Bonnie Lynn Webber (1980). Knowledge Representation for Syntactic/Semantic Processing. *Proc. First Annual National Conference on Artificial Intelligence*, Stanford, California: 316–323.

Dart, Susan A., Robert J. Ellison, Peter H. Feiler and A. Nico Habermann (1987). Software Development Environments. *Computer* 20(11):18–28.

de Kleer, Johan (1986). An Assumption-based TMS. *Artificial Intelligence* 28(2):127–162.

Earley, Jay and Paul Caizergues (1972). A Method for Incrementally Compiling Languages with Nested Statement Structure. *Communications of the ACM* 15(12):1040–1044.

Erman, Lee D., Frederick Hayes-Roth, Victor R. Lesser and D. Raj Reddy (1980). The HEARSAY-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys* 12(2):213–253.

Fritzson, Peter (1985). The Architecture of an Incremental Programming Environment and Some Notions of Consistency. *Proc. Workshop on Software Engineering Environments for Programming-in-the-Large*, June 9–12, 1985, Harwichport, Massachusetts: 64–79.

Ghezzi, Carlo and Dino Mandrioli (1979). Incremental Parsing. *ACM Transactions on Programming Languages and Systems* 1(1):58–70.

Ghezzi, Carlo and Dino Mandrioli (1980). Augmenting Parsers to Support Incrementality. *Journal of the Association for Computing Machinery* 27(3):564–579.

Haddock, Nicholas J. (1987). Incremental Interpretation and Combinatory Categorial Grammar. *Proc. Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy: 661–663.

Hirst, Graeme (1987). *Semantic Interpretation and the Resolution of Ambiguity.* Cambridge University Press, Cambridge, England.

Kay, Martin (1982). Algorithm Schemata and Data Structures in Syntactic Processing. In: Sture Allén, ed., *Text Processing. Proceedings of Nobel Symposium 51.* Almqvist & Wiksell International, Stockholm, Sweden: 327–358. Also: CSL-80-12, Xerox PARC, Palo Alto, California.

Koskenniemi, Kimmo (1983 a). *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production.* Publication No. 11, Department of General Linguistics, University of Helsinki, Helsinki, Finland.

Koskenniemi, Kimmo (1983 b). Two-Level Model for Morphological Analysis. *Proc. Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, Federal Republic of Germany: 683–685.

Lindstrom, G. (1970). The Design of Parsers for Incremental Language Processors. *Proc. 2nd ACM Symposium on Theory of Computing*, Northampton, Massachusetts: 81–91.

Mellish, Christopher S. (1985). *Computer Interpretation of Natural Language Descriptions*. Ellis Horwood, Chichester, England.

Reps, Thomas and Tim Teitelbaum (1987). Language Processing in Program Editors. *Computer* 20(11):29–40.

Sågvall Hein, Anna (1987). Parsing by Means of Uppsala Chart Processor (UCP). In: Leonard Bolc, ed., *Natural Language Parsing Systems*. Springer-Verlag, Berlin: 203–266.

Shieber, Stuart M. (1984). The Design of a Computer Language for Linguistic Information. *Proc. 10th International Conference on Computational Linguistics/22nd Annual Meeting of the Association for Computational Linguistics*, Stanford, California: 362–366.

Shieber, Stuart M. (1985). Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms. *Proc. 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, Illinois: 145–152.

Shieber, Stuart M. (1986). *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes No. 4. University of Chicago Press, Chicago, Illinois.

Shieber, Stuart M., Hans Uszkoreit, Fernando C. N. Pereira, Jane J. Robinson and Mabry Tyson (1983). The Formalism and Implementation of PATR-II. In: Barbara Grosz and Mark Stickel, eds., *Research on Interactive Acquisition and Use of Knowledge*. SRI Final Report 1894, SRI International, Menlo Park, California.

Stock, Oliviero (1986). Dynamic Unification in Lexically Based Parsing. *Proc. 7th European Conference on Artificial Intelligence*, Brighton, England: 212–221.

Tennant, Harry R., Kenneth M. Ross, Richard M. Saenz, Craig W. Thompson and James R. Miller (1983). Menu-Based Natural Language Understanding. *Proc. 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, Massachusetts: 151–158.

Thompson, Henry (1981). Chart Parsing and Rule Schemata in GPSG. Research Paper No. 165, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland. Also in: *Proc. 19th Annual Meeting of the Association for Computational Linguistics*, Stanford, California: 167–172.

Thompson, Henry and Graeme Ritchie (1984). Implementing Natural Language Parsers. In: Tim O'Shea and Marc Eisenstadt, *Artificial Intelligence: Tools, Techniques, and Applications*. Harper & Row, New York, New York: 245–300.

Tomita, Masaru (1985). An Efficient Context-Free Parsing Algorithm for Natural Languages. *Proc. Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California: 756–764.

Tomita, Masaru (1986). *Efficient Parsing for Natural Language. A Fast Algorithm for Practical Systems*. Kluwer Academic Publishers, Norwell, Massachusetts.

Tomita, Masaru (1987). An Efficient Augmented-Context-Free Parsing Algorithm. *Computational Linguistics* 13(1–2):31–46.

Wirén, Mats (1987). A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing. *Proc. Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark: 226-233. Also: Research report LiTH-IDA-R-87-13, Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Wirén, Mats (1988). A Control-Strategy-Independent Parser for PATR. *Proc. First Scandinavian Conference on Artificial Intelligence*, Tromsø, Norway: 155–160. Revised as: Research report LiTH-IDA-R-88-10, Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Avdelning, institution, fakultet
Division, department, faculty

Department of Computer and
Information Science
Linköping University

ISBN:

ISSN: ISSN-0281-4250

Rapportnr:
Report no: LiTH-IDA-R-88-11

Upplagans storlek:
Number of copies:

Datum:
Date: May 1988

Projekt:
Project:

Titel:
Title: An Incremental Chart Parser for PATR

Författare:
Author: Mats Wirén

Uppdragsgivare:
Commissioned by:

Dnr.:
Call no:

Rapporttyp:
Kind of report:

- ☐ Examensarbete/Final project
- ☐ Delrapport/Progress report
- ☐ Reserapport/Travel report
- ☐ Slutrapport/Final report
- ☒ Övrig rapport/Other kind of report

Rapportspråk:
Language:

- ☐ Svenska/Swedish
- ☒ Engelska/English
- ☐ _____

Sammanfattning (högst 150 ord):
Abstract (150 words):

This paper describes a technique for incremental chart parsing under a unification-based grammatical formalism, PATR, and outlines how this fits into continued work aimed at developing a parsing system which is both interactive and incremental. Incremental parsing means that input is analysed in a piecemeal fashion, in particular to allow arbitrary changes of previous input without having to reparse more than necessary. Interactive parsing means that the analysis is performed while the input is being entered (e.g. typed) to the system. The combination of these techniques could be used as a kernel for highly interactive and "reactive" natural-language processors, such as parsers for dialogue systems and language-sensitive text editors.

Nyckelord (högst 8):
Keywords (8):

Bibliotekets anteckningar

A Selection of Previous Research Reports.

LiTH-IDA-R-88-03 **Staffan Bonnier, Jan Maluszynski:** Towards a Clean Amalgamation of Logic Programs with External Procedures. Accepted to the *Joint Fifth Int. Logic Programming Conf. and Fifth Logic Programming Symp.*, Seattle Washington, August 15-19, 1988.

LiTH-IDA-R-88-02 **Ulf Nilsson:** Inferring Restricted AND-Parallelism in Logic Programs using Abstract Interpretation.

LiTH-IDA-R-88-01 **Jacek Leszczylowski, Staffan Bonnier, Jan Maluszynski:** Logic Programming with External Procedures: Introducing S-Unification. A shorter version is to appear in *Information Processing Letters*, North Holland.

LiTH-IDA-R-87-26 **Jonas Löwgren:** Applying a Rapid Prototyping System to Control Panel Dialgues.

LiTH-IDA-R-87-24 **Sven Moen:** Drawing Dynamic Trees.

LiTH-IDA-R-87-22 **Lars Ahrenberg, Arne Jönsson:** An Interactive System for Tagging Dialogues. Also in *Proc. of 14th Conference of the Association for Literary and Linguistic Computing*, Göteborg 1-5 June, 1987.

LiTH-IDA-R-87-21 **Harold W. Lawson, Jr.:** Challenges and Directions in Computers and Education.

LiTH-IDA-R-87-20 **Krzysztof Kuchcinski, Zebo Peng:** Parallelism Extraction from Sequential Programs for VLSI Applications. This paper is to appear in *Microprocessing and Microprogramming, the Euromicro Journal*, 1988.

LiTH-IDA-R-87-19 **Zebo Peng:** Construction of Asynchronous Concurrent Systems from their Behavioral Specifications. Also in *Proc. of the 10th World Computer Congress*, Dublin, Ireland, Sept. 1-5, 1986.

LiTH-IDA-R-87-18 **Henrik Nordin:** Reuse and Maintenance Techniques in Knowledge-Based Systems.

LiTH-IDA-R-87-17 **Tony Larsson:** Specification and Verification of VLSI Systems Actional Behaviour This is a close version of a paper presented at *the 8th international conference on Computer Hardware Description Languages, CHDL,* 87.

LiTH-IDA-R-87-16 **Shamsul I. Chowdhury:** State of the Art in Statistical Expert Systems. Also in *Proc. of the Conference on Expert Systems and their Applications*, Avignon, May 1987.

LiTH-IDA-R-87-15 **Nils Dahlbäck:** Kommunikation med datorer i naturligt språk - vad är det och vem behöver det?

LiTH-IDA-R-87-14 **Peter Fritzson:** Window System Architectures - an Overview. Also presented at *A Tutorial on Software Development Environments,* Linköping University, Nov 24-25 1986.

LiTH-IDA-R-87-13 **Mats Wirén:** A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing. Also in *Proc. of the Third Conference of the European Chapter of the Association for Computational Linguistics,* Copenhagen, April 1—3, 1987.

LiTH-IDA-R-87-11 **Sture Hägglund, Christer Hansson, Tomas Sokolnicki:** Knowledge-Based Training of Case Management Routines and Emergency Procedures. Also in *Proc. of the 3rd Int. Conf. on Expert Systems*, London, 1987.

LiTH-IDA-R-87-10 **Andrzej Lingas:** On Parallel Complexity of the Subgraph Isomorphism Problem.

LiTH-IDA-R-87-09 **Andrzej Lingas, Marek Karpinski:** Subtree Isomorphism and Bipartite Perfect Matching are Mutually NC Reducible.

LiTH-IDA-R-87-08 **Christos Levcopoulos, Andrzej Lingas, Jörg-R. Sack:** Nearly Optimal Heuristics for Binary Search Trees with Geometric Generalizations. Also in *Proc of ICALP'87*, Karlsruhe, July 1987.

LiTH-IDA-R-87-07 **Peter Haneclou:** A Formal Approach to Reason-maintenance Based on Abstract Domains.

LiTH-IDA-R-87-06 **Johan Hultman:** COPPS - A Software System for Defining and Controlling Actions in a Mechanical System.

The Department of Computer and Information Science
Linköping University

organizes undergraduate and graduate studies in *Computer Science*, *Telecommunication and Computer Systems*, and *Administrative Data Processing*. Research activities have an emphasis on advanced software technology and computer systems design and are organized in a number of research laboratories:

- *ACTLAB - Laboratory for Complexity of Algorithms*, which is concerned with the design and analysis of efficient sequential and parallel algorithms, and complexity theory, especially in the areas of computational geometry, data structures on bounded domains and graph algorithms.
- *AIELAB - Artificial Intelligence Environments Laboratory,* which conducts research on representation and manipulation of knowledge, problem solving techniques and expert systems with mechanical engineering applications.
- *ASLAB - Application Systems Laboratory,* which studies design of advanced support systems for interactive use of computers, including tools for automated construction of applications software.
- *CADLAB - Laboratory for Computer-Aided Design of Electronics*, which concentrates its research activities around tools for integrated development of hardware and software, graphics-based modelling and simulation techniques.
- *LIBLAB - Laboratory for Library and Information Science,* which studies methods for access to documents and the information contained in the documents, concentrating on catalogs and bibliographic representations, and on the human factors of library use.
- *LOGPRO - Laboratory for Logic Programming*, which concentrates its research activities around foundations of logic programming, relations to other programming paradigms and methodology.
- *NLPLAB - Natural Language Processing Laboratory*, which conducts research related to the development and use of natural language interfaces to computer software.
- *PELAB - Programming Environments Laboratory*, which works with design of tools for software development, specific functions in such tools and theoretical aspects of programs under construction.
- *RKLLAB - Laboratory for Representation of Knowledge in Logic*, which covers issues and techniques such as non-monotonic logic, probabilistic logic, modal logic and truth maintenance algorithms and systems.

Research Reports 1988

LiTH-IDA-R-88-11	**Mats Wirén:** An Incremental Chart Parser for PATR.
LiTH-IDA-R-88-10	**Mats Wirén:** A Control-Strategy-Independent Parser for PATR. Revised version of paper appearing in *Proc. of First Scandinavian Conf. on Artificial Intelligence*, Tromsö, Norway, March 9 - 11, 1988.
LiTH-IDA-R-88-09	**Peter Fritzson:** Incremental Symbol Processing.
LiTH-IDA-R-88-08	**Erik Sandewall:** Formal Semantics for Reasoning about Change with Ramified Causal Minimization.
LiTH-IDA-R-88-07	**Erik Sandewall et al:** IDA Annual Report 1987.
LiTH-IDA-R-88-06	**Christer Bäckström:** A Representation of Coordinated Actions Characterized by Interval Valued Conditions.
LiTH-IDA-R-88-05	**Christer Bäckström:** Keeping and Forcing: How to Represent Cooperating Actions.
LiTH-IDA-R-88-04	**W[illegible] an Maluszynski:** Al[illegible]