LiTH-IDA-R-88-10 ISSN-0281-4250

INSTITUTIONEN FÖR DATAVETENSKAP
UNIVERSITETET OCH TEKNISKA HÖGSKOLAN
LINKÖPING

A Control-Strategy-Independent 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-88-10 ISSN-0281-4250

A Control-Strategy-Independent Parser for PATR

by

Mats Wirén

This paper describes M-PATR, a modular, control-strategy-independent parsing framework for the PATR formalism. M-PATR is control-strategy-independent in the sense of not being hardwired for any fixed control structure, and by letting the user or superior software system determine the processing regime by setting parameters that guide the rule-invocation strategy, search strategy, and parsing direction. It is argued that this kind of system is valuable as an experiment bed, as a pedagogical device, and as a kernel for natural-language processors that require various forms of opportunistic parsing strategies. In particular, an application of the system to interactive, incremental parsing is proposed.

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

A Control-Strategy-Independent Parser for PATR

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

Abstract

This paper describes M-PATR, a modular, control-strategy-independent parsing framework for the PATR formalism. M-PATR is control-strategy-independent in the sense of not being hardwired for any fixed control structure, and by letting the user or superior software system determine the processing regime by setting parameters that guide the rule-invocation strategy, search strategy, and parsing direction. It is argued that this kind of system is valuable as an experiment bed, as a pedagogical device, and as a kernel for natural-language processors that require various forms of opportunistic parsing strategies. In particular, an application of the system to interactive, incremental parsing is proposed.[1]

1 Introduction

This paper describes M-PATR, a modular, control-strategy-independent parsing framework for the PATR[2] formalism. M-PATR is control-strategy-independent in the sense of not being hardwired for any fixed control structure, and by letting the user or superior software system determine the processing regime by setting parameters that guide the rule-invocation strategy, search strategy, and parsing direction.

PATR is a unification-based linguistic formalism of the tool type, presently at the heart of much research in computational linguistics and natural-language processing (Shieber et al. 1983, Shieber 1984, Shieber 1986). The control-strategy independence is inherent in the PATR formalism itself which, by virtue of declarativeness, as well as the fact that the unification operation is associative and commutative, is neutral to different processing regimes (cf. section 2). M-PATR retains this control-strategy independence, essentially through the adoption of a maximally open-ended chart-parsing framework (Kay 1982).

Control-strategy independence is advantageous in natural-language processing because in this field (unlike, for example, in parser construction for programming languages) control issues tend to remain largely open. A good methodology is therefore to leave as many options as possible open in the parsing system, allowing the processing regime to be determined by parameters instead of altering the system itself. A system open to systematic variation of control strategies may be used to *statically* investigate processing performance under different strategies and sets of input. For example, one might be interested in finding out how useful a filtering technique is, or what mix of data-driven and hypothesis-driven processing should be practised. It could also be used for fine-tuning of the performance with respect to some particular application.

Furthermore, it is sometimes crucial for a system to utilize flexible run-time control, i.e., to switch *dynamically* (opportunistically) between different control strategies. The standard example of this is speech analysis. Another example is interactive, incremental parsing, which will be developed in a forthcoming paper (Wirén 1988). The purpose of such a parser is to analyse input in a piecemeal fashion, in particular so as to be able to handle arbitrary modifications of previous input without reparsing more of it than necessary (incrementality), and to perform this analysis as the input is being entered (interactivity). This kind of system may be used as a kernel for "reactive" natural-language processors, such as parsers for dialogue systems and language-sensitive text editors. Such a system requires direction-independence (as new input could be entered at arbitrary positions) and a flexible search machinery (to allow fine-tuning of the system behaviour in order to make it process the queue of tasks in a principled and efficient way).

Finally, a further use of the system is in training situations. In fact, the initial design characteristics of the system were largely determined by pedagogical needs.

[1]This research has been supported by the National Swedish Board for Technical Development. The system was implemented on machines donated by the Xerox Corporation through their University Grants Program.

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

2 The PATR Formalism

2.1 Introduction

PATR is a unification-based linguistic formalism of the tool type, developed at SRI International (Shieber et al. 1983, Shieber 1984, Shieber 1986). By virtue of its simplicity and generality it has during the last years become a kind of *lingua franca* in computational linguistics and natural-language processing.

PATR is context-free-based in the sense that grammar rules state how phrase types combine to yield other phrase types. However, phrase types in PATR are not atomic as in context-free grammar, but constitute an informational domain of *feature structures*, complex bundles of features and values, or *dags* (rooted directed acyclic graphs with labeled arcs). A grammar rule in PATR can be thought of as consisting of two parts: Given that X_0 is the left-hand-side symbol, the first part specifies how strings of types $X_1, \ldots, X_n$ are concatenated to form a string of type X_0. The second part specifies a set of constraints over the feature structures of types $X_0, \ldots, X_n$. *Unification* is the operation that determines if two types are compatible by building the most general type compatible with both.

2.2 PATR Grammars

Formally, a PATR grammar is a tuple $\langle D_N, T, P, S\rangle$ where D_N is a possibly infinite set of dags (nonterminals), T is a finite set of terminals, P is a finite set of productions (rules), and S is the start symbol. Each production P is of the form $A \rightarrow \alpha$, where A is a nonterminal and α is either a string of nonterminals or a terminal. The latter case constitutes a lexical entry; we then call A a preterminal.

The dags of the PATR nonterminal domain are usually depicted using a bracketed notation, like in the example below of a dag D_0 representing an *AP* constituent:

$$\begin{bmatrix} \textit{cat}: & \textit{AP} & \\ \textit{agreement}: & \boxed{1} & \begin{bmatrix} \textit{number}: & \textit{sg} \\ \textit{gender}: & \textit{t} \\ \textit{species}: & \textit{def} \end{bmatrix} \\ \textit{subject}: & \boxed{1} & \end{bmatrix}$$

Dags are either *complex* (like for example D_0) or *atomic* (like the values *AP*, *sg*, etc.). Complex dags can be viewed as partial functions from labels to dag values, and the notation $D(l)$ will therefore be used to denote the value associated with the label l in the dag D. In the same vein, we can refer to the *domain* of a dag as $dom(D)$. For example, $dom(D_0) = \{cat, agreement, subject\}$. A dag with an empty domain is called the *empty* dag, the *null* dag, or a *variable*, and is written "[ ]". A *path* $\langle l_0\, l_1 \cdots\rangle$ in a dag is a sequence of label names which can be used to refer to a particular subpart of a dag by repeated application. For example, $D_0(\langle agreement\ number\rangle) = sg$.[3] Dags can be reentrant; in other words two labels can be required to share one common value (token identity; Lisp "EQ"). This is indicated with coindexing boxes as above.

In order to define unification over dags, following Shieber (1986), we first define the *subsumption* relation, $\sqsubseteq$, which provides a partial ordering (a lattice structure) for dags. Roughly speaking, a dag D subsumes a dag D' if D contains a subset of the information in D'. The null dag, containing no information, trivially subsumes any other dag. An atomic dag subsumes another atomic dag if and only if they are equal. A complex dag D subsumes a complex dag D' if and only if $D(l) \sqsubseteq D'(l)$ for all $l \in dom(D)$ and $D'(p) = D'(q)$ for all paths p and q such that $D(p) = D(q)$.[4] For example, the following relations hold:

$$[\,] \sqsubseteq [a\colon\ b] \sqsubseteq \begin{bmatrix} a\colon & b \\ c\colon & [d\colon\ e] \end{bmatrix} \sqsubseteq$$

$$\begin{bmatrix} a\colon & b \\ c\colon & [d\colon\ e] \\ f\colon & [d\colon\ e] \end{bmatrix} \sqsubseteq \begin{bmatrix} a\colon & b \\ c\colon & \boxed{1}\ [d\colon\ e] \\ f\colon & \boxed{1} \end{bmatrix}$$

The *unification* of two dags D' and D'' can now be defined as the most general dag D such that $D' \sqsubseteq D$ and $D'' \sqsubseteq D$. We write this as $D = D' \sqcup D''$. The unification of two dags is not always well-defined. When two dags contain conflicting information or when they are of incompatible types, the unification does not exist (algebraically), and we then say that the corresponding unification procedure *fails*. Looking at the different types of dags, we get the cases of figure 1.

	Null	*Atomic*	*Complex*
Null	yes	yes	yes
Atomic	yes	EQ?	fail
Complex	yes	fail	?

Figure 1: Cases of unification

The null dag unifies trivially with any other dag. An atomic dag unifies with another atomic dag if they are equal. Atomic dags never unify with complex dags. A complex dag unifies with another complex dag if all their subdags unify.

[3] This is sometimes written $\langle D_0\ agreement\ number\rangle = sg$.

[4] Equal signs here means token identity, i.e., that the paths share one common value.

As stated previously, PATR rules describe how strings are concatenated to form larger strings (the context-free portion) and how their feature structures relate (the feature specifications). A PATR rule can be written as follows:

$$
\begin{array}{l}
X_0 \rightarrow X_1 X_2 \\
\quad \langle X_0\ cat \rangle = S \\
\quad \langle X_1\ cat \rangle = NP \\
\quad \langle X_2\ cat \rangle = VP \\
\quad \langle X_1\ agreement \rangle = \langle X_2\ agreement \rangle
\end{array}
$$

A rule may be represented as a dag implicitly recording both the context-free production and the feature specifications. Below is the dag corresponding to the previous rule:

$$
\begin{bmatrix}
X_0: & \begin{bmatrix} cat: & S \end{bmatrix} \\
X_1: & \begin{bmatrix} cat: & NP \\ agreement: & \boxed{1}[\,] \end{bmatrix} \\
X_2: & \begin{bmatrix} cat: & VP \\ agreement: & \boxed{1} \end{bmatrix}
\end{bmatrix}
$$

This kind of rule format will be presupposed by the parsing algorithms in section 3 and 4. When referring to a rule, we will see it as an ordered pair $\langle P, D \rangle$, where P is a production $X_0 \rightarrow X_1 \cdots X_n$ and D is a dag representation of the rule with top-level labels $X_0, \ldots, X_n$ (like in the dag above).

When writing actual PATR grammars, it is however practical to eliminate the *cat* feature and record this information in the names of the constituents:

$$
\begin{array}{l}
S \rightarrow NP\ VP \\
\quad \langle NP\ agreement \rangle = \langle VP\ agreement \rangle
\end{array}
$$

2.3 Advantages of PATR

The principal advantages of the PATR formalism can be summarized as follows:

- declarativeness;
- monotonicity (nothing is ever lost or changed during derivation);
- order-independence (the grammar rules can be applied in any order);
- direction-independence (in the sense that it is neutral with respect to analysis and generation);
- mathematical well-foundedness (for example, it has been given a complete denotational semantics (Pereira and Shieber 1984)); and
- it provides, through the operation of unification, a natural and coherent way of checking, propagating, and merging partial information from different grammatical knowledge sources.

Put differently, PATR constitutes a linguistically felicitous, computationally effective, and mathematically well-founded programming language for linguistic description. (See Shieber (1986) for a more in-depth account of this.)

3 A Chart Parser for PATR

3.1 Technical Preliminaries

We adopt the following notation:[5] 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. An edge is a tuple

$$\langle i, j, X_0 \rightarrow \alpha.\beta, D \rangle$$

starting from vertex v_i and ending at vertex v_j with dotted rule $X_0 \rightarrow \alpha.\beta$[6] and dag D representing the rule.

3.2 A PATR Parsing Algorithm

We now give an algorithm for Earley-style (top-down, left-to-right) chart parsing under the PATR formalism. Alternative control strategies will be considered in section 4. We presuppose an agenda mechanism for scheduling of edge tasks (cf. section 4.3).

Input: A grammar $G = \langle D_N, T, P, S \rangle$ and an input string $w_1 \cdots w_n$.

Output: A chart.[7]

Method: The chart is initialized with an edge $\langle 1, 1, X_0 \rightarrow .\alpha, D \rangle$ 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.

For each vertex v_i do the following steps until no more edges can be added to the chart:

[5] The notation and terminology chosen here is rather unorthodox in reflecting both that of Earley's algorithm (Aho and Ullman 1972:321) and that of chart parsing (for example, Kay (1982), Thompson (1981), and Thompson and Ritchie (1984)).

[6] 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.

[7] The (whole) chart is usually not the kind of output that we are interested in. Rather, a reasonable output is obtained by extracting from the final chart the set of all root dags for the input string. In other words, the dags of all S edges spanning the entire chart should be extracted. Additionally, one might be interested in the corresponding context-free parse trees. These can be extracted in an analogous way provided that the context-free phrase structure is somehow recorded as the chart edges are built.

Scan: If $w_i = a$, then for all lexical entries of the form $\langle X_0 \rightarrow a, D\rangle$, add the edge $\langle i, i{+}1, X_0 \rightarrow a., D\rangle$.[8]

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

Predict/TopDown: For each edge ending at v_j of the form $\langle i, j, X_0 \rightarrow \alpha.X_m\beta, D\rangle$ and each rule of the form $\langle Y_0 \rightarrow \gamma, E\rangle$ such that $E(\langle Y_0\ cat\rangle) = D(\langle X_m\ cat\rangle)$, add the edge $\langle j, j, Y_0 \rightarrow .\gamma, E\rangle$ if it is not subsumed[9] 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 of the form $\langle i, j, X_0 \rightarrow \alpha.X_m\beta, D\rangle$ and each edge of the form $\langle j, k, Y_0 \rightarrow \gamma., E\rangle$, add the edge $\langle i, k, X_0 \rightarrow \alpha X_m.\beta, D \sqcup [X_m\colon E(Y_0)]\rangle$ 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,[10] and the dag of the inactive edge can be unified in with the dag of the needed constituent.[11]

An important difference between context-free and unification-based chart parsing concerns the redundancy test needed (at the prediction step) to prohibit more than one edge of the same kind from being added to the chart (resulting, for example, from left-recursion). In context-free chart parsing, we can ensure that no two identical edges are added simply by testing for equality between atomic categories. In unification-based parsing we instead have to make sure that no previously added edge *subsumes* a proposed edge. This is because we are only interested in new edges that are less specific than the old ones, the reason being that everything we could do with a more specific edge we could also do with a more general one.

There is another difference concerning the prediction step. As Shieber (1985) points out, when moving to an infinite nonterminal domain, parsing is no longer guaranteed to terminate if we predict from the whole feature structures. To get around this, he introduces a *restriction* operation which limits the nonterminal domain to a finite set of equivalence classes. Since it is desirable to be able to use the full feature-structure information, I plan to include this operation in the M-PATR system.

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

[9] 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.

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

[11] The basic conditions according to which the new edge is formed is sometimes referred to as "the fundamental rule" of chart parsing (Thompson 1981:2).

4 The Control-Strategy Space

4.1 Introduction

The control-strategy space in parsing is here viewed as extending along three dimensions: rule invocation, search, and parsing direction.

Since PATR is a declarative formalism, it is independent of particular processing strategies and control structures. M-PATR retains this control-strategy independence, essentially through the adoption of a maximally open-ended chart-parsing framework. More specifically, "independence" means that within each control-strategy dimension M-PATR is supplied with a number of parameterized "standard" strategies, and that, if needed, the user should be able to functionally specify new strategies on top of the original system. Note that it is outside the scope of M-PATR proper to decide when to select a particular strategy; rather, it is up to the user or to a superior application system to determine this.

A system similar in spirit is Henry Thompson's MCHART (Thompson 1981), an open-ended chart-parsing framework which is independent of control with respect to rule invocation and search. The main difference between MCHART and M-PATR is that the latter is hardwired for a particular grammatical format, PATR, whereas the former has a functional interface for determining the grammar formalism. In addition, MCHART is aimed at left-to-right parsing only.

4.2 Rule-Invocation Strategy

4.2.1 Introduction

The rule-invocation strategy in chart parsing is determined by the conditions for when and how new edges are predicted (the prediction step). The following standard rule-invocation strategies are currently entertained by the system: top-down, bottom-up (i.e., left-corner), top-down with bottom-up filtering, bottom-up with top-down filtering, and a strategy based on the alternative bottom-up algorithm according to Kilbury (1985). Predictions and filtering are currently guided only by information pertaining to the context-free categories of rules. As mentioned in section 3, using information from the full feature sets requires some kind of restriction in order to guarantee termination.

Calls to predictors other than those supplied by the system currently have to be added manually into the code. A cleaner mechanism for determining the rule-invocation strategy, such as the kind of signal table utilized by MCHART, may be added later on.

4.2.2 Bottom-Up

We now give the predictor for bottom-up (or left-corner) chart parsing under the PATR formalism.

Predict/BottomUp: For each edge starting at v_i of the form $\langle i, j, X_0 \rightarrow \alpha., D\rangle$ and each rule of the form $\langle Y_0 \rightarrow Y_1\beta, E\rangle$ such that $E(\langle Y_1\, cat\rangle) = D(\langle X_0\, cat\rangle)$, add an edge of the form $\langle i, i, Y_0 \rightarrow .Y_1\beta, E\rangle$ if this edge is not subsumed by another edge.

Informally, this means predicting an edge according to each rule whose first right-hand-side category matches the category of the inactive edge under consideration.

Note that the chart does not have to be initialized in bottom-up parsing.

4.2.3 Bottom-Up à la Kilbury

We now give Kilbury's bottom-up predictor (Kilbury 1985) as conceived of under the PATR formalism.

Predict/Kilbury: For each edge starting at v_i of the form $\langle i, j, X_0 \rightarrow \alpha., D\rangle$ and each rule of the form $\langle Y_0 \rightarrow Y_1\beta, E\rangle$, add an edge of the form $\langle i, j, Y_0 \rightarrow Y_1.\beta, E \sqcup [Y_1\colon D(X_0)]\rangle$ if the unification succeeds.

Informally, this means predicting an edge according to each rule whose first right-hand-side category matches the category of the inactive edge under consideration,[12] provided that the dag of the inactive edge can be unified in with the first needed constituent of the edge to be predicted. The predicted edge should extend between the same vertices as the triggering edge.

Again, note that the chart does not have to be initialized in bottom-up parsing. Note also that the resulting predicted edge may be active or inactive depending on the contents of the inactive edge and on what is required by the active edge.

This strategy has two advantages (cf. Wirén 1987): First, it apparently saves edges since it avoids predicting empty active edges. Secondly, the usual redundancy (subsumption) test is not needed. The reason for this is that a predicted edge always covers the triggering (inactive) edge. Since the triggering edge is guaranteed to be unique, the predicted edge will also be unique.[13] This seems to be a significant advantage because the subsumption test may be very costly, especially when running large grammars. The price to be paid for this is that ϵ-rules (empty productions) cannot be handled. It must also be kept in mind that right-recursive rules may result in gross inefficiency for unfiltered bottom-up strategies.

4.2.4 Top-Down and Bottom-Up Filtering

Top-down filtering with respect to the context-free categories can be formed from the standard bottom-up strategy by adding a further condition which rejects a predicted edge if its category is not needed by any previous active edge. Analogously, bottom-up filtering can be formed from the top-down strategy by adding a further condition which rejects a predicted edge if it cannot combine with any subsequent preterminal (inactive) edge (cf. Wirén 1987).

4.3 Search Strategy

The search strategy in chart parsing is determined by the policy according to which the agenda is managed. More specifically, it depends on the order in which edges are added to the chart and the order in which tasks, i.e., pairs of active and inactive edges, are considered under the fundamental rule.

M-PATR makes use of an ordered agenda mechanism which controls the adding of edges to the chart and the running of tasks. The user may specify what priority level each such task is to be queued at, and may also specify what ordering regime is to be applied to the queue. LIFO (depth-first) and FIFO (breadth-first) are provided as standard options by the system. Alternative ordering regimes must be functionally specified by the user.

It might be argued that the choice of search strategy in general is irrelevant in chart parsing since it only affects the *order* in which the successive partial analyses are developed, and hence does not affect processing efficiency. But if the system is to handle updates of the input in the course of parsing, as in incremental parsing, the system has to work "non-monotonically" in which case the amount of work to be done cannot be determined in advance. Different search strategies may then exhibit different performance characteristics. Consequently, it becomes important to fine-tune the behaviour of the system to make it process the queue of tasks in a principled

[12] Note that, again, this condition is tested by the unification which specifically ensures that $E(\langle Y_1 cat\rangle) = D(\langle X_0 cat\rangle)$.

[13] This presupposes that no two rules of the grammar have the same context-free skeleton.

and efficient way, for example in order to report input errors as quickly as possible.

4.4 Parsing Direction

Systems that process typed input typically proceed in a strict left-to-right fashion. Speech applications, on the other hand, have frequently utilized alternative parsing strategies. For example, HEARSAY (Erman et al. 1980) and GLP (Goerz 1981) employed *island driving*: find whatever words are immediately identified, then expand out to either side of all of them. HWIM (Wolf and Woods 1980) adopted a hybrid between island driving and a left-to-right strategy.

M-PATR takes advantage of the order-independence of chart parsing (and of PATR) by being able to parse a sentence either left to right or right to left. More specifically, this means that the system may *scan* input in either direction; what happens after every scanning step depends on the mode of rule invocation and search. The standard chart-parsing technique with edges extending from left to right on the chart is adopted both ways.

Thompson (1981:8 ff.) has suggested a more refined technique for bidirectional chart parsing according to which active edges can be extended in either direction. This idea has been further developed by Steel and De Roeck (1987) and by Stock et al. (1988). There are two advantages of this technique: First, in certain situations, it may save a lot of edges. For example, consider a rule like $A \rightarrow A$ *conjunction* A. If prediction of the rule could be triggered bottom-up from the conjunction (generating an active, bidirectional A edge) instead of invoking the rule every time an A constituent is found, then a lot of work could be saved.

Secondly, this technique seems particularly suitable for the processing of noisy input (such as speech) because it increases the predictive power through the ability of extending active edges in both directions. Incidentally, this also provides an advantage in incremental parsing. In ordinary chart parsing, the last vertex of the chart will at the end of a parse typically contain predictions corresponding to various possible continuations of the sentence. An incremental chart parser could then, upon addition of input to the right, immediately resume analysis. However, there is an asymmetry in that no such predictions will exist for possible additions to the left; those predictions, if needed (for example in order to form a hypothesis about an unrecognizable item), would have to be generated separately. A chart parser employing bidirectional edges, on the other hand, would typically have access to predictions going both ways.

In practice, the use of bidirectional chart edges presupposes some kind of control indicating whether a rule is to be used for top-down or bottom-up prediction, and, if bottom-up, which categories should trigger it. (In case of heavily noisy input, this may not be necessary.) For this purpose, Steel and De Roeck (1987) introduce rule annotations (which only affect the parsing process and not the grammar). A problem with this approach is the magnification of the task of grammar writing that it leads to, and possibly also that it leads to a conceptual mingling of linguistic and technological problems. In addition, as Steel and De Roeck point out, it is not clear how to guarantee a correct set of annotations; conflicting or incomplete annotations may result in loss of parses. Therefore, adding this kind of machinery to the system has not (as of yet) seemed worthwhile.

5 The M-PATR System

This section briefly describes some additional aspects in connection with the M-PATR system.

A grammatical description in the system consists not only of rules and lexical entries, but also of a number of *templates* (cf. Karttunen 1986:76). Templates are used for encoding common subpatterns among lexical items, thereby facilitating concise language definitions. A template definition consists of a template name followed by a number of dags and/or names of other templates.

So far only a couple of experimental and very limited grammars have been used together with the system. A version of the Swedish PATR grammar by Merkel (1986) is planned to be utilized. Also, as the system is being put to use within related research projects at Linköping, such as the LINLIN natural-language-interface project (Ahrenberg 1987), other grammars will be used as well. Furthermore, it is planned to use a modified version of M-PATR in the CRIME (critiquing in medicine) text-generation project (Rankin 1988).

M-PATR currently does not include any facilities for representing morphological generalizations — all word forms must be entered individually. An important task is to augment the system with a morphological analyser, possibly of the two-level type (Koskenniemi 1983 a, 1983 b). Incidentally, the kind of tree-structured lexicon used in two-level morphology seems to fit particularly well to interactive parsing.

The graph-unification technique adopted is Karttunen's "reversible" unification which, according to Karttunen (1986:79), has shortened parsing times by a factor of three compared to the kind of unification with extensive copying that was used in the earlier PATR implementations. The recently proposed "nondestructive" unification (Wroblewski 1987) in

effect seems to be very much like Karttunen's technique.

The system has been implemented in INTERLISP-D for Xerox Lisp machines. The system relies heavily on the graphical facilities of these machines. In particular, the chart is drawn incrementally in a window as the parsing proceeds, and the user may select control-strategy parameters by clicking in a menu.

6 Conclusion

The kind of control-strategy-independent parsing framework described in this paper is useful for several purposes: First, it may be used to statically investigate processing performance under different control strategies and sets of input. It could also be used for fine-tuning system behaviour with respect to some particular application. Furthermore, it may be used as a kernel for parsing systems that need flexible run-time control by switching dynamically (opportunistically) between different control strategies. Several of these properties will be explored within continued work aimed at developing an interactive, incremental parsing system (Wirén 1988).

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, and Magnus Merkel for rewarding discussions. The implementation owes much to what I have learnt from Lauri Karttunen on unification-based parsing. Bernt Nilsson has cunningly sorted out various implementation problems. Sven Moen implemented the chart-display routine.

References

Ahrenberg, Lars (1987). Parsing into Discourse Object Descriptions. *Proc. Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark: 140–147.

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.

Goerz, G. (1981). GLP: A General Linguistic Processor. *Proc. Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., Canada: 429–431.

Karttunen, Lauri (1986). D-PATR: A Development Environment for Unification-Based Grammars. *Proc. 11th International Conference on Computational Linguistics*, Bonn, Federal Republic of Germany: 74–80.

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: Report CSL-80-12, Xerox PARC, Palo Alto, California.

Kilbury, James (1985). Chart Parsing and the Earley Algorithm. KIT-Report 24, Projektgruppe Künstliche Intelligenz und Textverstehen, Technische Universität Berlin, West Berlin. Also in: U. Klenk, ed. (1985), *Kontextfreie Syntaxen und verwandte Systeme. Vorträge eines Kolloquiums in Grand Ventron im Oktober, 1984.* Niemeyer, Tübingen, Federal Republic of Germany.

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.

Merkel, Magnus (1986). A Swedish Grammar in D-PATR. Experiences of Working with D-PATR. Research report LiTH-IDA-R-86-31, Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Pereira, Fernando C. N. and Stuart M. Shieber (1984). The Semantics of Grammar Formalisms Seen as Computer Languages. *Proc. 10th International Conference on Computational Linguistics/ 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford, California: 123–129.

Rankin, Ivan (1988). Towards Effective Text Generation in Critiquing Expert Systems. Ms., Department of Computer and Information Science, Linköping University, Linköping, Sweden.

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.

Steel, Sam and Anne De Roeck (1987). Bidirectional Chart Parsing. *Advances in Artificial Intelligence. Proceedings of the 1987 AISB Conference, University of Edinburgh, 6–10 April 1987.* John Wiley & Sons, Chichester, England: 223–235.

Stock, Oliviero, Rino Falcone and Patrizia Insinnamo (1988). Island Parsing and Bidirectional Charts. To appear in *Proc. 12th International Conference on Computational Linguistics*, Budapest, Hungary.

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.

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). An Incremental Chart Parser for PATR. Research report LiTH-IDA-R-88-11, Department of Computer and Information Science, Linköping University, Linköping, Sweden.

Wolf, J. and William A. Woods (1980). The HWIM Speech Understanding System. In: W. Lea, ed., *Trends in Speech Recognition.* Prentice-Hall, Englewood Cliffs, New Jersey: 316–339.

Wroblewski, David A. (1987). Nondestructive Graph Unification. *Proc. Sixth National Conference on Artificial Intelligence*, Seattle, Washington: 582–587.

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-88-10

Upplagans storlek:
Number of copies:

Datum:
Date: May 1988

Projekt:
Project:

Titel:
Title: A Control-Strategy-Independent 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 M-PATR, a modular, control-strategy-independent parsing framework for the PATR formalism. M-PATR is control-strategy-independent in the sense of not being hardwired for any fixed control structure, and by letting the user or superior software system determine the processing regime by setting parameters that guide the rule-invocation strategy, search strategy, and parsing direction. It is argued that this kind of system is valuable as an experiment bed, as a pedagogical device, and as a kernel for natural-language processors that require various forms of opportunistic parsing strategies. In particular, an application of the system to interactive, incremental parsing is proposed.

Nyckelord (högst 8):
Keywords (8):

Bibliotekets anteckningar

A Selection of Previous Research Reports.

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.

LiTH-IDA-R-87-05 **Christer Bäckström:** Logical Modelling of Simplified Geometrical Objects and Mechanical Assembly Processes.

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-88-10	**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	**Wlodek Drabent, Simin Nadim-Tehrani, Jan Maluszynski:** Algor[illegible]
LiTH-IDA-R-88-03	**Staff**[illegible] Clean Amal[illegible]ocedures. Accepted to th[illegible]*d Fifth Logic Prog*[illegible]5-19, 1988.