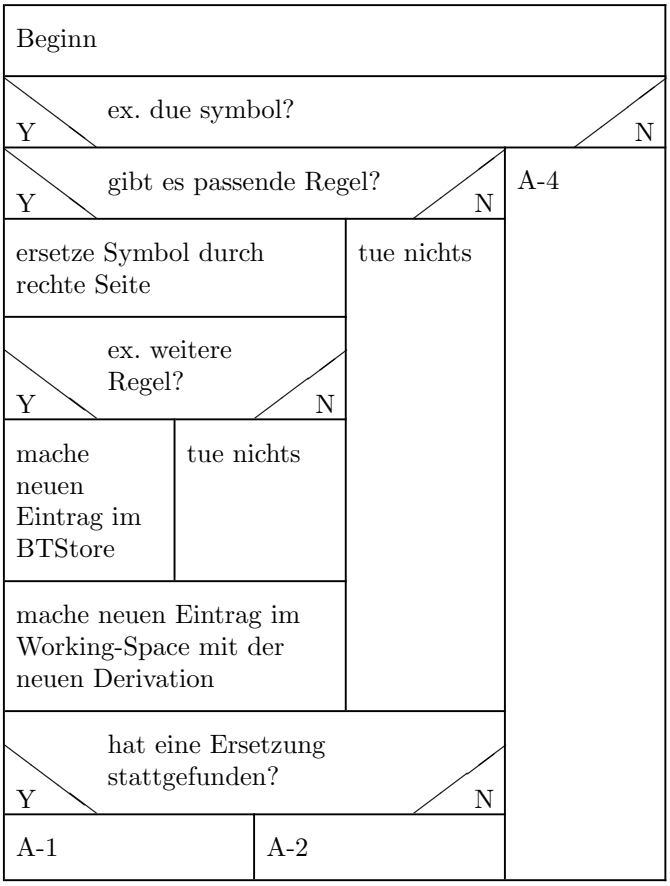
PS Maschinelle Syntaxanalyse (Parsing)
WS 01/02
PT-1 Projektbericht

Manu Raster

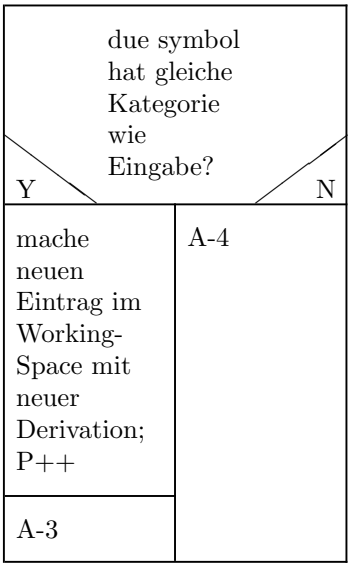
24. April 2002

1 PT-1 in Nassi-Shneiderman Diagrammen

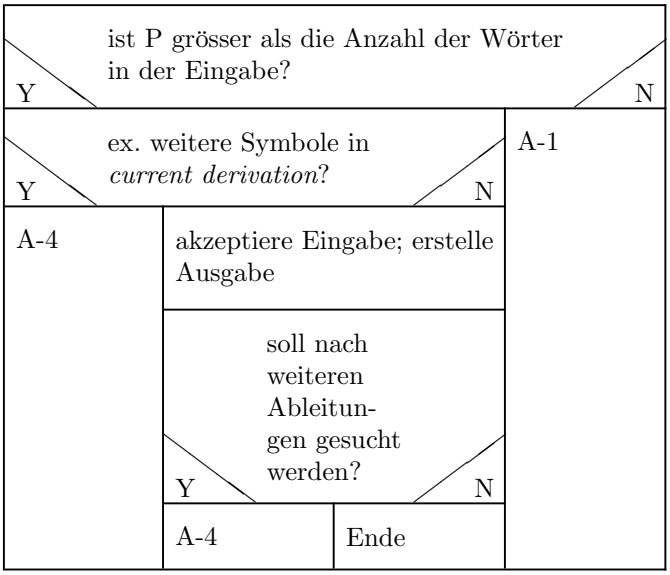
A-1 — Expansion of non-terminal categories



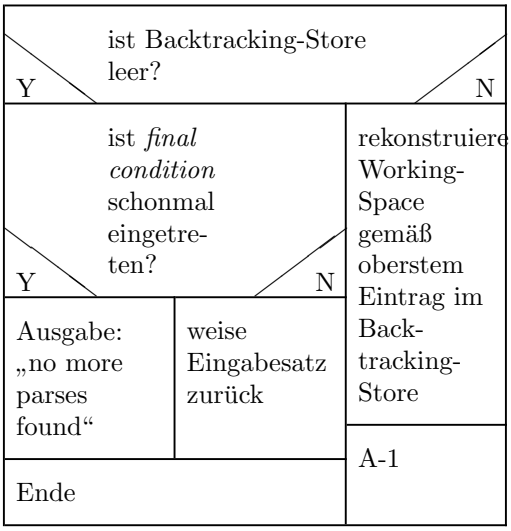
A-2 — Recognition of terminal categories



A-3 — Final condition



A-4 — Backtracking



2 die Umsetzung in Java

Die Wahl der Programmiesprache fiel auf Java aus folgenden Gründen:

- In den Java-Klassenbibliotheken stehen alle benötigten Datenstrukturen wie *Stack*, Arrays mit dynamischer Grösse (*Vector*), *Hashtables* zu Verfügung.
- Weitere nützliche Klassen stehen im *online-parser*-Projekt von Markus Dreyer [1] zur Verfügung. Der *online-parser* ist gut dokumentiert und stellt Klassen zur Handhabung von Produktionsregeln, Lexikon und Ein- und Ausgabe in Java bereit.
- Die Bestandteile von PT-1 (Working-Space, Backtracking-Store, Algorithmus) lassen sich gut als Klassen im OO-Paradigma modellieren.

Die Darstellung des Algorithmus in Nassi-Shneiderman Diagrammen erwies sich als äußerst hilfreich für die Umsetzung in Java. Schwierigkeiten traten allerdings bei der Umsetzung von A-4 auf. Das Vorbereiten der kommenden Regelausführung nach dem Rücksprung und die Kontrolle des Zählers für mehrere Regeln zu einem NT erforderte ausgiebiges Testen. Die Fehlersuche wurde dadurch erschwert, dass sich keiner der gängigen Debugger benutzen ließ, weil die Klassen des *online-parser*-Projekts keine Debug-Informationen enthalten und deren Quellcode auch nicht offenliegt. So mussten an vielen Stellen `println`-Anweisungen eingebaut werden, um die Variablen während der Laufzeit beobachten zu können.

2.1 Merkmale

Der PT-1 Parser besteht aus drei Klassen: `PT1`, `WorkingSpace`, `BTStore`. Instanzen von den beiden Klassen `WorkingSpace` und `BTStore` werden in der Hauptklasse `PT1` als Elemente von Stacks verwendet. Die Stack-Methode `push` erwartet

eine Kopie des Objekts, das auf den Stack gelegt werden soll. `WorkingSpace` und `BTStore` implementieren daher die `cloneable`-Klasse.

Die vorliegende Version des PT-1 kommt gut zurecht mit sinnvollen Eingabesätzen wie z.B. „they visit friends in Egypt“, „we sleep“, „we visit pyramids with friends in Egypt“. Diese Software ist aber nicht perfekt oder etwa fehlerfrei. Beim Ziel eine möglichst gute Version vom PT-1 zu bauen, sind die folgenden Punkte als erstes zu berücksichtigen:

- Exception Handling. Ungrammatische Sätze oder Sätze mit Wörtern, die nicht im Lexikon stehen, werden manchmal noch nicht richtig behandelt. Der Parser landet dann in einer Endlosschleife.
- Ausgabe des Parse-trees

Literatur

[1] Markus Dreyer. *Parser programmieren in Java*. 2000. `http://janus.cl.uni-heidelberg.de/kurs/ss00/javapars/`.

[2] Peter Hellwig. *Parsing natürlicher Sprachen: Grundlagen, Parsing natürlicher Sprachen: Realisierungen*, pages 348–431. In I. S. Bátori [5], 1989.

[3] Peter Hellwig. *Natural Language Parsers – A Course in Cooking*. 1999. `http://janus.cl.uni-heidelberg.de/kurs/ws01/pars/pars99.ps`.

[4] Ian Holyer. *Compiler Design Formal Forum*. 1999. `http://www.cs.bris.ac.uk/~ian/formal/`.

[5] W. Putschke I. S. Bátori, W. Lenders, editor. *Computational Linguistics. Ein internationales Handbuch zur computerunterstützten Sprachforschung und ihrer Anwendung*. DeGruyter, Berlin, 1989.