

# j-Algo

The Algorithm  
Visualisation Tool

Benutzerhandbuch



# Inhaltsverzeichnis

<b>I. Das Hauptprogramm</b>	<b>7</b>
<b>1. Das Hauptprogramm</b>	<b>9</b>
1.1. Einleitung . . . . .	9
1.2. Technische Hinweise . . . . .	9
1.2.1. Systemvoraussetzungen . . . . .	9
1.2.2. Installation . . . . .	9
1.2.3. Deinstallation . . . . .	10
1.3. Grundfunktionen . . . . .	11
1.3.1. Neues Modul öffnen . . . . .	11
1.3.2. Gespeicherte Sitzungsdaten laden . . . . .	11
1.3.3. Sitzungsdaten speichern . . . . .	12
1.3.4. Modul schließen . . . . .	12
1.3.5. Einstellungen . . . . .	12
1.3.6. Hilfe . . . . .	13
1.3.7. Hinweis-Tipps . . . . .	13
1.4. Impressum . . . . .	14
<b>II. Die Module</b>	<b>15</b>
<b>2. Das Modul AVL-Bäume</b>	<b>17</b>
2.1. Einleitung . . . . .	17
2.2. Funktionsübersicht . . . . .	17
2.3. Programmstart - Der Willkommensbildschirm . . . . .	18
2.3.1. Baum laden . . . . .	18
2.3.2. Baum von Hand erstellen . . . . .	18
2.3.3. Zufallsbaum erstellen lassen . . . . .	19
2.3.4. Willkommensbildschirm anzeigen . . . . .	19
2.4. Die Arbeitsfläche . . . . .	20
2.5. Modulfunktionen . . . . .	21
2.5.1. Schlüsseingabe . . . . .	21
2.5.2. Algorithmusfunktionen . . . . .	22
2.5.3. AVL-Modus . . . . .	22
2.5.4. Baum auf AVL-Eigenschaft testen . . . . .	22
2.5.5. Baum löschen . . . . .	23
2.6. Algorithmussteuerung . . . . .	23

## Inhaltsverzeichnis

2.6.1.	Schritt-Pfeile . . . . .	23
2.6.2.	Abbruch und Beenden-Buttons . . . . .	24
2.6.3.	Animationsgeschwindigkeit . . . . .	24
2.7.	Dokumentation . . . . .	24
2.7.1.	Skript . . . . .	24
2.7.2.	Logbuch . . . . .	25
2.7.3.	Infobereich . . . . .	25
2.8.	Zusatzfunktionen . . . . .	25
2.8.1.	Navigators . . . . .	25
2.8.2.	Beamermodus . . . . .	26
2.9.	Impressum . . . . .	27
<b>3.</b>	<b>Das Modul Dijkstra</b> . . . . .	<b>29</b>
3.1.	Einleitung . . . . .	29
3.2.	Funktionsübersicht . . . . .	29
3.3.	Modul starten . . . . .	29
3.4.	Symbolleiste . . . . .	29
3.5.	Graph erstellen . . . . .	30
3.5.1.	Graphische Eingabe/Erstellen eines Graphen per Maus . . . . .	30
3.5.2.	Die Knotenliste . . . . .	31
3.5.3.	Die Kantenliste . . . . .	31
3.5.4.	Die Adjazenzmatrix . . . . .	31
3.6.	Ablauf des Algorithmus . . . . .	32
3.7.	Impressum . . . . .	33
<b>4.</b>	<b>Das Modul KMP</b> . . . . .	<b>35</b>
4.0.1.	Einleitung . . . . .	35
4.0.2.	Funktionsübersicht . . . . .	35
4.1.	Der Willkommensbildschirm . . . . .	35
4.1.1.	Modul KMP starten . . . . .	35
4.1.2.	Generierung der Verschiebetabelle . . . . .	36
4.1.3.	Suchen im Suchtext . . . . .	36
4.1.4.	Öffnen einer KMP-Sitzung . . . . .	36
4.1.5.	Präsentation von Lernbeispielen . . . . .	36
4.2.	Der Arbeitsbereich . . . . .	36
4.3.	Anzeigeoptionen . . . . .	37
4.3.1.	Skalierung . . . . .	37
4.3.2.	Aufteilung der Bereiche . . . . .	37
4.3.3.	Beamer - Modus . . . . .	38
4.3.4.	Zyklus - Anzeige . . . . .	38
4.4.	Legende . . . . .	39
4.4.1.	Elemente in der Phase 'Generierung der Verschiebetabelle' . . . . .	39
4.4.2.	Elemente in der Phase 'Suchen im Suchtext' . . . . .	39
4.4.3.	Im Dokumentationsbereich . . . . .	40
4.5.	Modulfunktionen . . . . .	41
4.5.1.	Eingabe von Pattern und Text . . . . .	41
4.5.2.	Generieren von Pattern und passenden Texten . . . . .	42

4.5.3.	Generierung der Verschiebetabelle . . . . .	42
4.5.4.	Suchen im Suchtext . . . . .	43
4.5.5.	Öffnen einer KMP-Sitzung . . . . .	44
4.5.6.	Präsentation von Lernbeispielen . . . . .	44
4.6.	Algorithmussteuerung . . . . .	44
4.7.	Dokumentation . . . . .	45
<b>5.</b>	<b>Das Modul Ebnf und Syntaxdiagramme</b>	<b>47</b>
5.1.	Einleitung und Funktionsübersicht . . . . .	47
5.2.	Der Willkommensbildschirm . . . . .	48
5.3.	EBNF-Definitionen . . . . .	49
5.3.1.	Der EBNF-Editor . . . . .	49
5.3.2.	Anzeige von EBNF-Defintionen . . . . .	53
5.3.3.	Speichern und Laden von EBNF-Defintionen . . . . .	55
5.4.	Der trans()-Algorithmus . . . . .	56
5.4.1.	Die Arbeitsfläche . . . . .	56
5.4.2.	Steuerung des Algorithmus . . . . .	57
5.5.	Syntaxdiagramme . . . . .	58
5.5.1.	Der Syntaxdiagramm-Editor . . . . .	58
5.5.2.	Die Syntaxdiagramm-Anzeige . . . . .	61
5.5.3.	Speichern und Laden von Syntaxdiagrammen . . . . .	62
5.6.	Der Rücksprung-Algorithmus . . . . .	63
5.6.1.	Die Arbeitsfläche . . . . .	63
5.6.2.	Steuerung des Algorithmus . . . . .	63
5.7.	Impressum . . . . .	69
<b>A.</b>	<b>Einleitung zu Datenstrukturen</b>	<b>71</b>
<b>B.</b>	<b>Suchbäume</b>	<b>73</b>
<b>C.</b>	<b>AVL-Bäume</b>	<b>77</b>

## *Inhaltsverzeichnis*

Teil I.

# Das Hauptprogramm





# 1. Das Hauptprogramm

## 1.1. Einleitung

Dieses Handbuch stellt eine Einführung und Hilfe für die Arbeit mit **j-Algo** dar. **j-Algo** ist eine Software, die sich mit der Visualisierung von Algorithmen beschäftigt. Sie soll dazu dienen, verschiedene Algorithmen zu veranschaulichen um sie so Studenten und anderen Interessierten verständlicher zu machen. Die Anwendung basiert auf einer Plugin-Struktur, die es ermöglicht, einzelne Module, die jeweils einen Algorithmus oder ein Themengebiet abdecken können, in das Programm zu integrieren und zu laden.

Sowohl **j-Algo** als auch die einzelnen Module entstanden im Rahmen des externen Softwarepraktikums im Studiengang Informatik der TU Dresden in Zusammenarbeit mit dem Lehrstuhl Programmierung. Die implementierten Module orientieren sich daher an den Lehrveranstaltungen „Algorithmen und Datenstrukturen“ sowie „Programmierung“ im Grundstudium Informatik an der TU Dresden. Das Einsatzgebiet soll vor allem die Vorlesung und das studentische Lernen zu Hause umfassen.

**j-Algo** ist eine freie Software, die beliebig oft kopiert werden darf.

## 1.2. Technische Hinweise

### 1.2.1. Systemvoraussetzungen

Folgende minimale Systemanforderungen werden für den reibungslosen Einsatz von **j-Algo** benötigt:

- IBM-kompatibler PC
- Mindestens 64 MB RAM
- WINDOWS 98(SE)/ME/2000/XP , LINUX SuSE/Red Hat
- Java 2 Platform Standard Edition 5.0 (siehe: <http://java.sun.com/>)
- Maus und Tastatur
- Monitor mit einer Auflösung von mindestens 800x600

### 1.2.2. Installation

#### Windows

Entpacken Sie nach dem Herunterladen das ZIP-komprimierte Archiv in einen Ordner Ihrer Wahl. In diesem Ordner finden Sie eine Datei namens „j-algo.bat“. Öffnen Sie diese Datei mit einem Doppelklick, und das Programm wird gestartet.

## *1. Das Hauptprogramm*

### **Unix**

Entpacken Sie nach dem Herunterladen das TGZ-komprimierte Archiv in einen Ordner Ihrer Wahl. In diesem Ordner finden Sie eine Datei namens „j-algo.sh“. Öffnen Sie die Konsole und starten sie mittles `sh j-algo.sh` das Programm.

### **1.2.3. Deinstallation**

Der komplette Programmordner kann jederzeit gefahrlos von der Festplatte gelöscht werden.

## 1.3. Grundfunktionen

**j-Algo** bietet eine Reihe von Grundfunktionen, die unabhängig von den Modulen zur Verfügung stehen, bzw. für jedes Modul die gleiche Bedeutung haben. Im Einzelnen sind das das Öffnen von Modulen sowie das Laden und Speichern von Sitzungsdaten. Die Grundfunktionen sind über die Werkzeugleiste oder den Menüpunkt <DATEI> erreichbar.

### 1.3.1. Neues Modul öffnen

Ein Klick auf den Button <NEU> in der Werkzeugleiste gibt Ihnen die Möglichkeit, ein beliebiges neues Modul zu öffnen. Dabei wird ein Auswahldialog geöffnet, in welchem die installierten Module aufgelistet sind. Hier werden Ihnen außerdem kurze Informationen zu diesen Modulen angezeigt. Sie können wählen, ob dieser Auswahldialog bei jedem Start des Programmes angezeigt werden soll oder nicht.

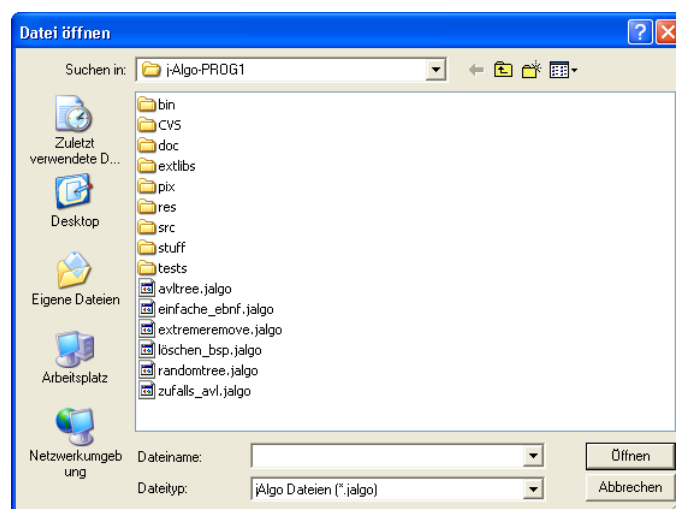
Alternativ dazu kann über das Menü <DATEI>→<NEU> das gewünschte Modul geladen werden. Dies ist der schnellere Weg und zu empfehlen, wenn man bereits einen Überblick über die installierten Module hat.

### 1.3.2. Gespeicherte Sitzungsdaten laden

Mit einem Klick auf den Button <ÖFFNEN> erscheint ein Dialog zur Dateiauswahl. Hier haben Sie die Möglichkeit, eine Datei auszuwählen, in welcher modulspezifische Sitzungsdaten gespeichert wurden. Die Dateien, die von **j-Algo** gespeichert werden, tragen die Dateiendung „.jalgo“.

Achtung: Da jedes Modul von **j-Algo** seine Daten in einer solchen Datei ablegt, kann man beim Blick auf die ungeöffnete Datei nicht erkennen, mit welchem Modul diese assoziiert wurde. Es wird jeweils das assoziierte Modul zu der geladenen Datei geöffnet. Achten Sie daher bei der Vergabe der Dateinamen auf möglichst eindeutige Bezeichner.

Anmerkung: In einer späteren Version wird direkt bei der Dateiauswahl das zugehörige Modul mit angezeigt.



Das Dialogfenster zum Öffnen

## 1. Das Hauptprogramm

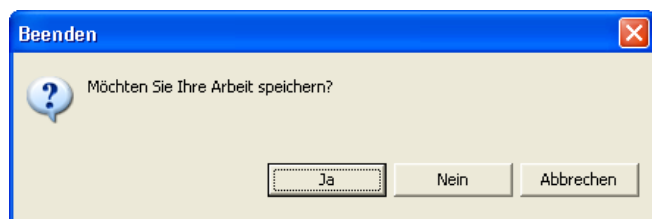
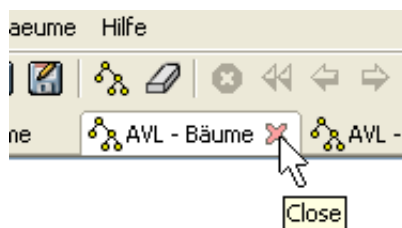
### 1.3.3. Sitzungsdaten speichern

Per Klick auf die Buttons <SPEICHERN> und <SPEICHERN UNTER> können Sie die Sitzungsdaten des gerade aktiven Moduls in einer Datei speichern. Wie beim Laden öffnet sich auch hier ein Dialog zur Dateiauswahl, in welchem Sie Zielpfad und Name der neuen Datei eintragen können. Die Angabe der Dateiendung ist nicht nötig, das Programm ergänzt diese automatisch.

Je nach Implementierung des aktiven Moduls steht die Speicherfunktion nur zur Verfügung, wenn gerade kein Algorithmus läuft. Sollte noch ein Algorithmus aktiv sein, so beenden Sie diesen bitte vorher oder brechen ihn ab.

### 1.3.4. Modul schließen

Sie haben die Möglichkeit, jede Modulinstanz durch Klick auf das Kreuz der dazugehörigen Registerkarte zu schließen. Dabei werden Sie gegebenenfalls gefragt, ob Sie Ihre Arbeit speichern wollen. Um das gesamte Programm zu schließen, ist es nicht nötig, die Module einzeln zu schliessen, das erledigt das Programm für Sie.

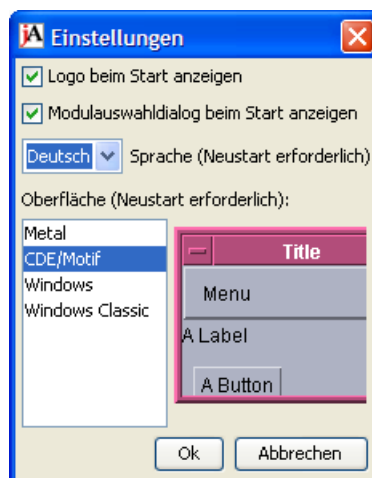


Der Knopf zum Schließen eines Moduls.

Die Abfrage, ob die Daten gespeichert werden sollen.

### 1.3.5. Einstellungen

**j-Algo** bietet ein paar Möglichkeiten an, das Programm den Bedürfnissen des Benutzers anzupassen. Den Dialog für die Grundeinstellungen erreichen Sie unter dem Menüpunkt <DATEI>→<EINSTELLUNGEN>



Der Dialog für die Grundeinstellungen

Unter anderem kann hier die Sprache eingestellt werden. Außerdem bietet sich die Möglichkeit, die Art der graphischen Oberfläche zu ändern, da unter manchen Betriebssystemen diverse Oberflächenelemente nicht immer vorteilhaft aussehen.

#### 1.3.6. Hilfe

Die Hilfe stellt ein wichtiges Nachschlagewerk für all diejenigen dar, die nicht auf Anhieb mit allen Funktionen von **j-Algo** und seinen Modulen klar kommen. Hier können Sie noch einmal eine genaue Beschreibung zu den einzelnen Programmelementen nachlesen.

Die Hilfe ist kontextspezifisch aufgebaut, d.h. ist ein Modul geöffnet, so wird in der Hilfe automatisch an die entsprechende Stelle gesprungen.

Sie erreichen die Hilfe über den Menüpunkt `<HILFE>→<INHALT>` oder indem Sie einfach auf die Taste `<F1>` Ihrer Tastatur drücken.

#### 1.3.7. Hinweis-Tipps

Zusätzlich wird zu den meisten Kontrollelementen, also Buttons, Menüeinträge, etc., ein kurzer Hinweistext neben dem Mauszeiger bzw. in der Statuszeile des Programmes angezeigt. Dies sollte als schnelle Hilfestellung den meisten Anforderungen genügen.

## 1. Das Hauptprogramm

### 1.4. Impressum

Die **j-Algo** Software wurde im Sommersemester 2004 von der Praktikumsgruppe SWT04-PROG1 im Rahmen des externen Softwarepraktikums entwickelt. Mitwirkende waren die

#### Teammitglieder

- Michael Pradel — Chief of Algorithms
- Cornelius Hald — Chief of Framework
- Malte Blumberg
- Stephan Creutz
- Christopher Friedrich
- Anne Kersten
- Hauke Menges
- Babett Schalitz
- Benjamin Scholz
- Marco Zimmerling

Die Webseite des Praktikums finden Sie unter <http://web.inf.tu-dresden.de/~swt04-p1/>. Komplette Überarbeitung erfuhr die Software und die Dokumentation unter anderem durch Alexander Claus und Matthias Schmidt. Weitergehende Informationen über **j-Algo** erhalten Sie unter <http://j-algo.binaervarianz.de/>.

## Teil II.

# Die Module





## 2. Das Modul AVL-Bäume

### 2.1. Einleitung

Das Modul **AVL-Bäume** realisiert die Darstellung von binären Such- und AVL-Bäumen. Für eine detaillierte Beschreibung dieser Baumtypen lesen Sie bitte nach im Vorlesungsskript von Prof. Vogler „Algorithmen, Datenstrukturen und Programmierung“. Eine mehr oder weniger kurze Einführung in diese Thematik ist auch zu finden im Anhang [A](#).

**Anmerkung:** In dieser Version von **j-Algo** ist das Modul **AVL-Bäume** unter Linux aus Kompatibilitätsgründen nicht installiert.

### 2.2. Funktionsübersicht

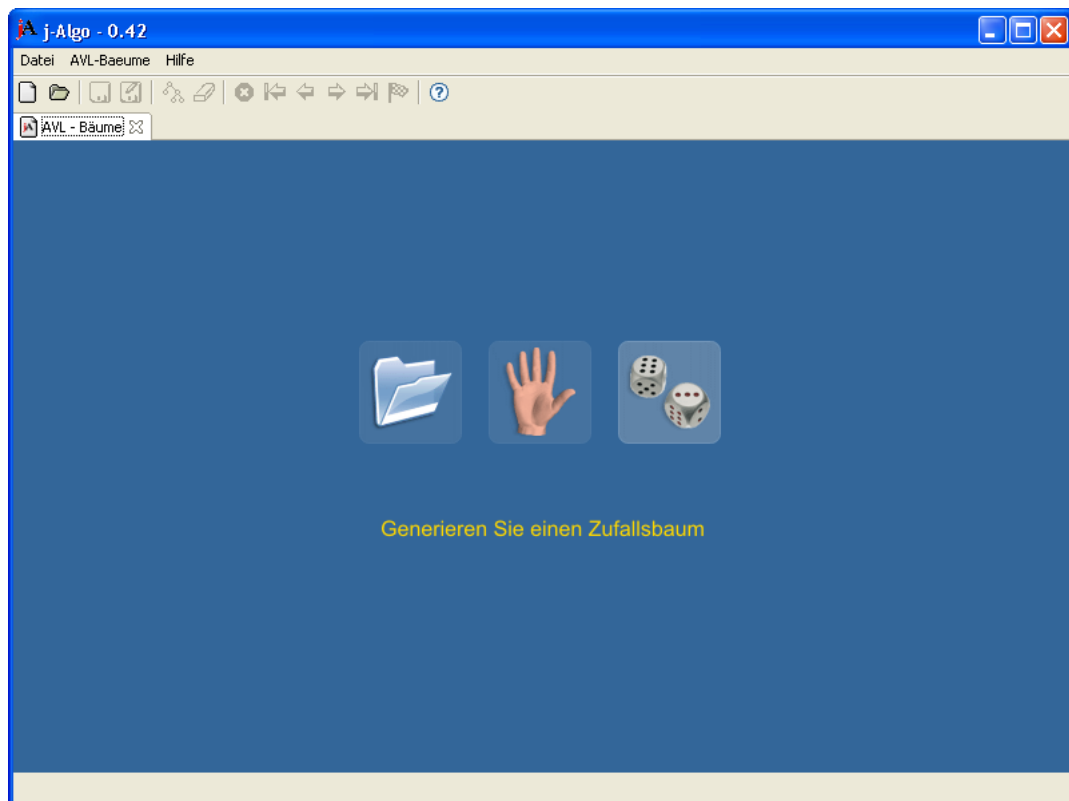
Das Modul **AVL-Bäume** realisiert folgende Funktionen:

- Visualisieren von binären Suchbäumen mit und ohne AVL-Eigenschaft
- Einfügen, Suchen und Löschen von Baumknoten
- Testen eines Baumes auf die AVL-Eigenschaft
- Generieren von zufälligen Suchbäumen
- Speichern und Laden von Bäumen
- Informationen zum Baum und zu den laufenden Algorithmen

## 2. Das Modul AVL-Bäume

### 2.3. Programmstart - Der Willkommensbildschirm

Nach Starten des Hauptprogramms **j-Algo** können Sie über den Button <NEU> oder mit dem Menüpunkt <DATEI>→<NEU>→<**AVL-BÄUME**> eine neue Instanz des Moduls **AVL-Bäume** öffnen. Anschließend öffnet sich der Willkommensbildschirm des Moduls, der Ihnen verschiedene Möglichkeiten eröffnet.



Der Willkommensbildschirm des Moduls **AVL-Bäume**

#### 2.3.1. Baum laden

Mit Klick auf das Ordner-Symbol öffnet sich ein Dialogfenster, in dem Ihnen die Möglichkeit gegeben wird, eine „\*.jalgo“-Datei auszuwählen, in welcher ein Baum gespeichert wurde. Im Prinzip ist die Bedeutung dieses Buttons die gleiche wie des <ÖFFNEN>-Buttons in der Werkzeugleiste. Der Unterschied besteht darin, dass der Button in der Werkzeugleiste eine neue Modulinstanz öffnet, in welcher die Datei geladen wird, der Button im Startbildschirm von **AVL-Bäume** jedoch die Datei in die aktuell geöffnete Modulinstanz lädt.

#### 2.3.2. Baum von Hand erstellen

Mit Klick auf das Hand-Symbol gelangen Sie sofort zur leeren Arbeitsfläche des Moduls **AVL-Bäume**. Sie können jetzt mit der knotenweisen Generierung eines neuen Suchbaumes beginnen.

### 2.3.3. Zufallsbaum erstellen lassen

Mit Klick auf das Würfel-Symbol beginnen Sie die Generierung eines zufällig erzeugten Suchbaumes. In dem folgenden Dialogfenster können Sie verschiedene Daten zum Baum und die Art der Visualisierung festlegen.



Eingabe der Zufallsbaumdaten

- **Anzahl der Knoten**

Geben Sie hier die Anzahl der Knoten ein. Der entstehende Baum muss mindestens einen Knoten enthalten, höchstens aber 99.

- **AVL-Eigenschaft**

Aktivieren Sie dieses Kästchen, wenn der zu erstellende Baum die AVL-Eigenschaft besitzen soll.

- **Visualisierung**

Wählen Sie hier die Art der Visualisierung der Erstellung aus.

- KEINE

- Der Baum wird sofort erstellt.

- SCHRITTWEISE

- Jeder Algorithmusschritt kann von Ihnen per Hand bestätigt werden.

- AUTOMATISCH

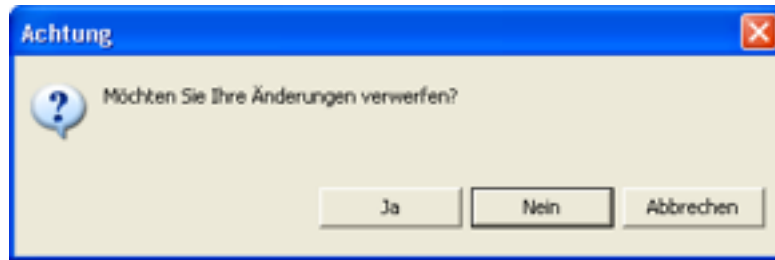
- Lassen Sie die Erstellung des Baumes als Animation ablaufen, die Geschwindigkeit ist dabei einstellbar.

Haben Sie schrittweise oder automatische Visualisierung gewählt, können Sie den Ablauf jederzeit abbrechen. Dabei wird das gerade aktive Knoteneinfügen abgebrochen, und der Baum steht mit entsprechend weniger Knoten zur Verfügung.

### 2.3.4. Willkommensbildschirm anzeigen

Mit Klick auf diesen Button in der Werkzeugleiste des Modulbildschirms kann der Willkommensbildschirm später jederzeit wieder angezeigt werden. Dabei werden Sie eventuell gefragt, wie Sie mit Ihren Änderungen verfahren wollen. Sollten Sie Ihre Änderungen nicht verwerfen wollen, so wird eine neue Instanz des Moduls geöffnet.

## 2. Das Modul AVL-Bäume



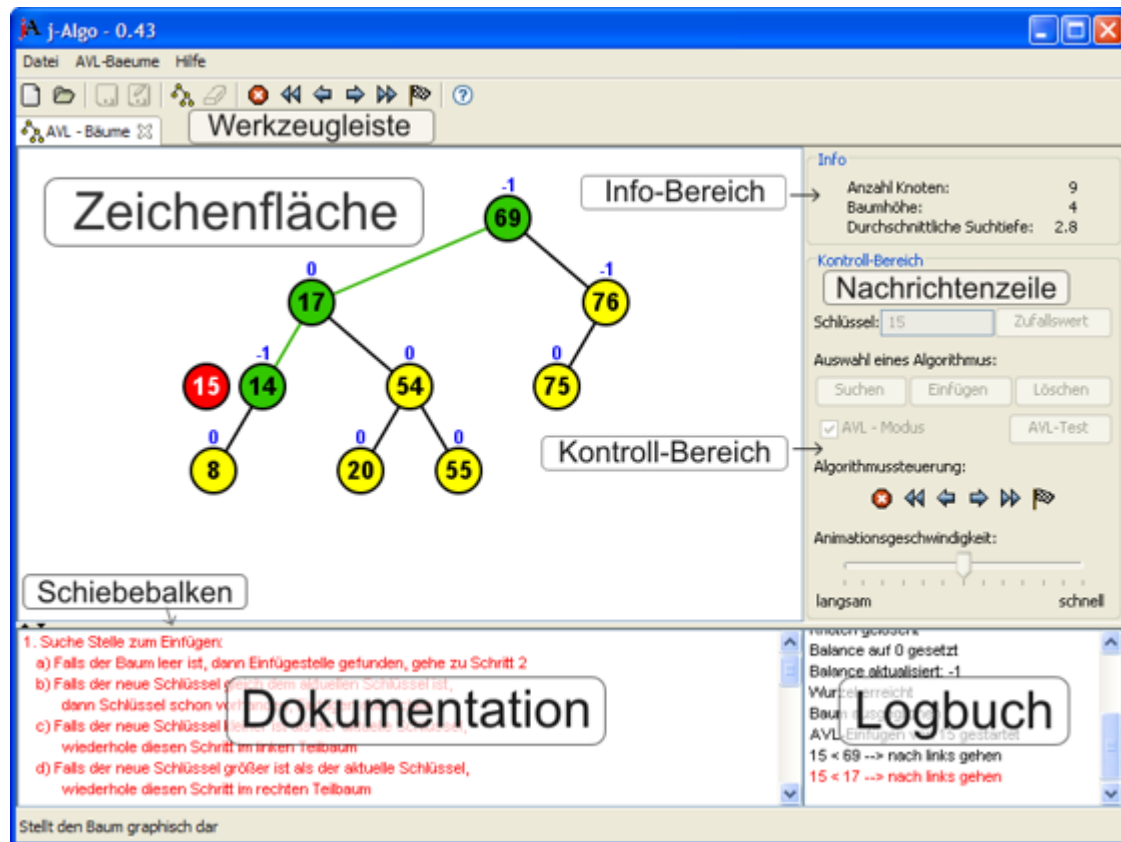
Dialog mit der Frage, ob der ganze Baum gelöscht werden soll.

### 2.4. Die Arbeitsfläche

Die Arbeit mit **AVL-Bäume** spielt sich auf der Arbeitsfläche ab. Sie bietet alle Funktionalitäten des Moduls und ist in fünf wichtige Bereiche unterteilt:

- **ZEICHENFLÄCHE**  
Der Baum und alle Algorithmen werden hier visualisiert.
- **INFOBEREICH**  
Wichtige Baumdaten wie Anzahl der Knoten, Baumhöhe und Suchtiefe sind hier zu finden.
- **KONTROLL-BEREICH**  
In diesem Bereich erfolgt der Start und die Steuerung der Algorithmen.
- **DOKUMENTATIONSBEREICH**  
Hier läuft der Text zum jeweiligen Algorithmus mit. Der aktuelle Schritt wird dabei farbig hervorgehoben. Der Text ist dem Skript „Algorithmen, Datenstrukturen und Programmierung“ von Prof. Vogler, Version vom 2. Oktober 2003, entnommen.
- **LOGBUCH**  
Hier werden erfolgte Einzelaktionen protokolliert und dabei der jeweils aktuelle Schritt farbig hervorgehoben.

Die Aufteilung zwischen dem unteren und dem oberen Bereich kann mit dem Schiebebalken verändert werden. Per Klick auf die schwarzen Pfeile können Sie den Textbereich wahlweise maximieren, um einen Überblick über den Algorithmustext zu gewinnen, oder minimieren, um die Zeichenfläche zu vergrößern.

Die Arbeitsfläche des Moduls **AVL-Bäume**

## 2.5. Modulfunktionen

Alle Funktionen des Moduls **AVL-Bäume** lassen sich über den Kontroll-Bereich der Arbeitsfläche bedienen. Sie stellen die verschiedenen Baumalgorithmen dar, deren Visualisierung Aufgabe dieses Moduls ist. Grundslegend läuft die Arbeit mit den Algorithmen immer nach dem gleichen Schema ab:

1. Schlüsseleingabe
2. Starten des Algorithmus per Klick auf den entsprechenden Button

Es gibt natürlich auch Algorithmen, wie der AVL-Test, die keinen Schlüssel benötigen und ohne Schritt 1 auskommen.

Es folgen nun die einzelnen Funktionen im Detail.

### 2.5.1. Schlüsseleingabe

Für die Eingabe der Schlüsselwerte steht ein Textfeld und ein Button für zufällige Werte zur Verfügung. Es sind nur ganzzahlige Schlüsselwerte von 1 bis 99 erlaubt.

Über dem Textfeld befindet sich eine Nachrichtenzeile, in welcher Sie auf eventuelle Fehleingaben aufmerksam gemacht werden. Hier werden später ebenfalls kurze Ergebnismeldungen zu den Algorithmen eingeblendet.

## 2. Das Modul AVL-Bäume

### 2.5.2. Algorithmusfunktionen

#### Knoten einfügen

Der eingegebene Wert wird als Schlüssel für einen neuen Knoten verwendet, der in den Baum eingefügt werden soll. Ist bereits ein Knoten mit dem gleichen Schlüssel im Baum enthalten, so bricht der Algorithmus erfolglos ab.

#### Knoten suchen

Nach dem Starten dieses Algorithmus beginnt die Suche nach dem eingegebenen Schlüssel im Baum.

#### Knoten löschen

Nach dem eingegebenen Schlüssel wird gesucht, und wenn ein entsprechender Knoten gefunden wurde, wird dieser aus der Baumstruktur entfernt.

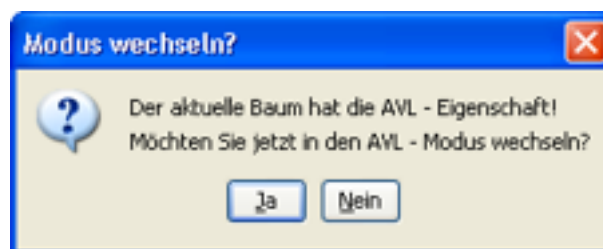
### 2.5.3. AVL-Modus

Ist dieses Kästchen aktiviert, werden die entsprechenden Algorithmen so ausgeführt, dass die AVL-Eigenschaft gewahrt bleibt.

Achtung: Es ist keine Funktion implementiert, die an einem beliebigen Suchbaum die AVL-Eigenschaft herstellt!

Ist das Kästchen deaktiviert, ist es daher nicht immer ohne weiteres wieder zu aktivieren. Dazu muss zuerst getestet werden, ob der Baum die AVL-Eigenschaft hat. Es ist jedoch jederzeit möglich, das Kästchen zu deaktivieren und einen unbalancierten Baum zu erzeugen.

### 2.5.4. Baum auf AVL-Eigenschaft testen




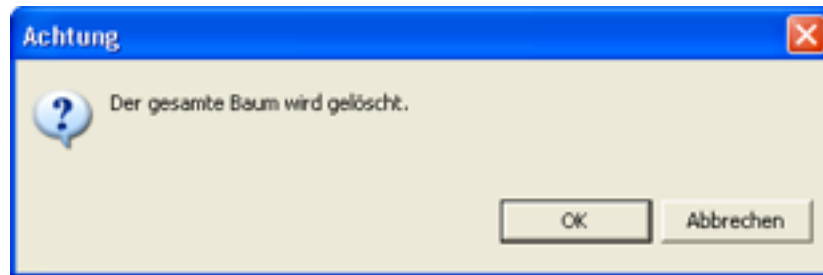
Hinweisfenster des AVL-Tests

Wenn der AVL-Modus einmal deaktiviert sein sollte, so ermöglicht das Programm einen Test des Baumes auf die AVL-Eigenschaft. Dabei erfolgt eine Berechnung und Anzeige der Balancen aller Knoten und das eventuelle Markieren von Knoten, deren Balance sich nicht mehr im Rahmen der AVL-Eigenschaft bewegt.

Es wird ein Hinweis-Dialog geöffnet, der Ihnen das Ergebnis des Tests präsentiert. Sollte der Baum tatsächlich die AVL-Eigenschaft besitzen, so wird Ihnen angeboten, direkt in den AVL-Modus zu wechseln.

### 2.5.5. Baum löschen

Mit einem Klick auf den Button  in der Werkzeugleiste können Sie nach einer Sicherheitsabfrage die gesamte Baumstruktur löschen und mit einer leeren Arbeitsfläche neu beginnen.



Sicherheitsabfrage beim Löschen des Baumes

## 2.6. Algorithmussteuerung

Aufgabe des Moduls **AVL-Bäume** ist es, Baumalgorithmen, wie das Einfügen und Löschen von Knoten, zu visualisieren. Jeder Algorithmus ist in verschiedene Teilschritte unterteilt, die nacheinander angezeigt werden. Das Visualisieren erfolgt dabei durch das Zeichnen des Baumes, durch die Erklärung der Schritte im Dokumentationsbereich und im Logbuch und durch die Neuberechnung der baumspezifischen Daten, die im Infobereich präsentiert werden. Nachdem Sie einen Algorithmus gestartet haben, verweilt er in einem Initialzustand und wartet auf Ihre Eingabe. Nun haben Sie die Möglichkeit, den Algorithmus in kleinen oder großen Schritten zu durchlaufen; Sie können ihn sofort beenden oder direkt abbrechen. Dafür bietet die Algorithmussteuerung die entsprechenden Werkzeuge.

### 2.6.1. Schritt-Pfeile

Mittels der Schritt-Pfeile steuern Sie die Abfolge der Einzelschritte und bekommen so eine detaillierte Sicht auf die Arbeitsweise des Algorithmus. Das Programm bietet Ihnen die Möglichkeit, einen Teilschritt rückgängig zu machen und damit gewisse Abläufe zu wiederholen. Die Schritt-Pfeile, welche die Rückgängigfunktion anbieten, weisen in ihrer Richtung nach links und sind dadurch intuitiv von den Vorwärts-Pfeilen zu unterscheiden. Zusätzlich gibt es für jede Richtung einen großen und einen kleinen Schritt, der per Knopfdruck ausgeführt wird.

Kleine Schritte beim Einfügen eines Knotens stellen Schlüsselvergleiche, Balancenberechnungen und Rotationen dar. Große Schritte hingegen sind zum Beispiel das Suchen der Einfügestelle, das Einfügen an dieser und die gesamte Balancenaktualisierung.

#### Einzel-Schritt-Pfeile

Ein Klick auf diese Buttons realisiert einen kleinen Algorithmusschritt zurück bzw. nach vorn.

## 2. Das Modul AVL-Bäume

### ⏮ ⏭ Block-Schritt-Pfeile

Ein Klick auf diese Buttons realisiert einen großen Schritt zurück bzw. nach vorn. Sollte der Algorithmusablauf an eine Stelle geraten, an der es nur noch einen kleinen Schritt nach vorn bzw. zurück gibt, so hat der Block-Schritt die selbe Funktionalität wie ein Einzel-Schritt.

### 2.6.2. 🚫 🏁 Abbruch und Beenden-Buttons

Klicken Sie auf den Beenden-Button 🏁 um den laufenden Algorithmus bis zum Ende auszuführen.

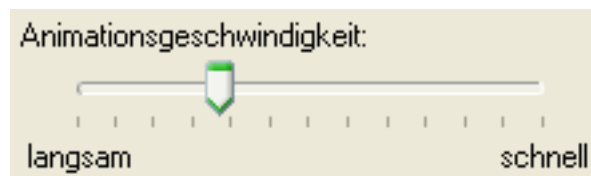
Klicken Sie auf den Abbruch-Button 🚫 um den laufenden Algorithmus abzubrechen. Der Baum hat danach den gleichen Status wie vor Beginn des Algorithmus.

Ist ein Algorithmus beendet, so steht Ihnen diese Option nicht mehr zur Verfügung, weil nur der *laufende* Algorithmus abgebrochen werden kann.

### 2.6.3. Animationsgeschwindigkeit

Beim Generieren eines Zufallsbaumes haben Sie die Option, den Ablauf der Baumerzeugung als Animation ablaufen zu lassen. Starten Sie in diesem Modus, so beginnt die Animation sofort und kann mit dem Geschwindigkeitsregler schneller oder langsamer abgespielt werden. Zu Beginn steht dieser auf der mittleren Position. Verschieben Sie den Regler nach links, um die Animation zu verlangsamen bzw. nach rechts, um sie zu beschleunigen.

Eine Animation der anderen Algorithmusabläufe ist in dieser Version von **AVL-Bäume** nicht integriert.



Der Regler für die Animationsgeschwindigkeit

## 2.7. Dokumentation

Da das Modul **AVL-Bäume** vor allem zu Lehr- und Lernzwecken eingesetzt werden soll, ist eine detaillierte Dokumentation der Algorithmen unumgänglich. Für die Einzelheiten des Algorithmustextes steht ein Auszug aus dem Vorlesungsskript von Prof. Vogler zur Verfügung. Ein Logbuch in der rechten unteren Ecke des Bildschirms führt Protokoll über den Stand und die Beschaffenheit des einzelnen Algorithmusteilschrittes.

Zu guter Letzt wird ein Infobereich angeboten, in dem wichtige Baumdaten zusammengefasst sind.

### 2.7.1. Skript

Der Dokumentationsbereich, der das Skript enthält, befindet sich am unteren Bildschirmrand. Es handelt sich hierbei um einen Auszug des Skripts zur Vorlesung „Algorithmen und Datenstrukturen“ von Prof. Vogler (TU Dresden), Version vom 2. Oktober 2003. Im Rahmen dieser



Vorlesung soll das Modul vorwiegend eingesetzt werden.

Bei dem jeweils aktuellen Algorithmustext handelt es sich um die Aktion, die als nächstes im Ablauf des Algorithmus erfolgen wird. Sie wird rot markiert angezeigt.

### 2.7.2. Logbuch

Das Logbuch ist eine weitere Möglichkeit, den Ablauf des Algorithmus zu verfolgen. Es bezieht sich in erster Linie auf baumspezifische Daten und verwendet zum Beispiel konkrete Schlüsselwerte, anhand deren die Aktionen des Algorithmus besser verstanden werden sollen.

Auch hier wird der aktuelle Eintrag rot markiert dargestellt. Dieser bezieht sich aber auf die zuletzt ausgeführte Aktion.

### 2.7.3. Infobereich

Der Infobereich ist hauptsächlich dafür gedacht, Ihnen schnell wichtige Daten des Baumes bereit zu stellen. Hier finden Sie folgende Punkte:

- **ANZAHL DER KNOTEN**  
Dieser Punkt fasst für Sie die Anzahl der Knoten im Baum zusammen.
- **BAUMHÖHE**  
Hier finden Sie die Anzahl der Level des Baumes.
- **DURCHSCHNITTLLICHE SUCHTIEFE**  
Dieser Wert berechnet sich durch die Summe der Level aller Knoten geteilt durch die Anzahl dieser. Der Wert ist ein Indiz dafür, wie gut der Baum ausbalanciert ist bzw. wie groß der Suchaufwand im Durchschnitt ist.

## 2.8. Zusatzfunktionen

Dieses Kapitel widmet sich den Eastereggs des Moduls **AVL-Bäume**.

Sollten Sie sich lieber selber gerne auf die Suche nach diesen Zusatzfunktionen machen wollen, so überspringen Sie besser dieses Kapitel.

Für alle Anderen folgt nun eine Übersicht zum Baumnavigator und dem Beamermodus.

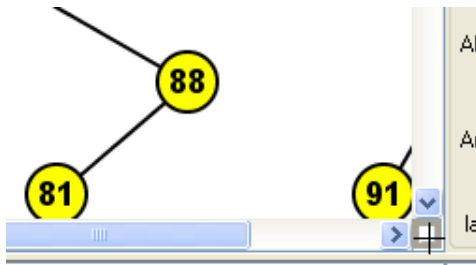
### 2.8.1. Navigator

Der Navigator ist eine kleine, versteckte Zusatzfunktion, die die Arbeit mit großen Bäumen erheblich vereinfachen kann. Er stellt eine willkommene Hilfe für das Scrollen der Zeichenfläche dar, ist aber nicht so einfach zu finden.

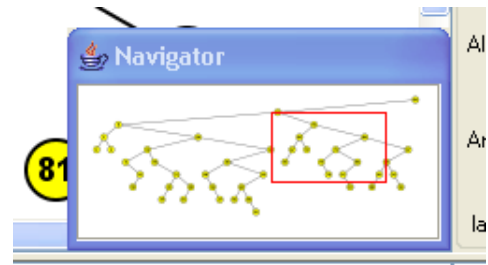
- Wenn der Baum, der auf der Zeichenfläche angezeigt wird, zu groß für diese wird, so erscheinen Schiebebalken, mit denen Sie den Bildausschnitt verschieben können.
- Klicken Sie nun auf das kleine Quadrat in der rechten unteren Ecke der Zeichenfläche, genau zwischen den beiden Schiebebalken. Halten Sie dabei die linke Maustaste gedrückt.

## 2. Das Modul AVL-Bäume

- Eine kleine Übersichtskarte des Baumes mit einem Ausschnittfenster erscheint. Bewegen Sie die Maus (mit gedrückter Taste) und das Ausschnittfenster, das den Bildschirminhalt der Zeichenfläche repräsentiert, folgt Ihren Bewegungen.



Ein Klick auf das kleine Kästchen...

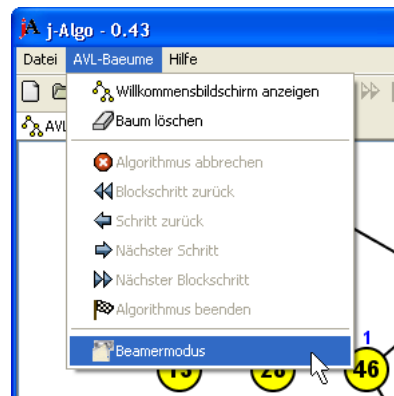


...öffnet den Navigator!

### 2.8.2. Beamermodus

Der Beamermodus ist in erster Linie für die Präsentation in Vorlesungen oder ähnlichen Veranstaltungen gedacht. Ist dieser Modus aktiv, so werden die Knoten des Baumes und die Einträge des Logbuches vergrößert dargestellt. Der Algorithmustext aus dem Skript von Prof. Vogler bleibt dabei unverändert, weil davon ausgegangen wird, dass die interessierten Studenten der Vorlesung über ein (eventuell aktuelleres) Skript verfügen.

Sie erreichen den Beamermodus über den Menüpunkt **<AVL-BÄUME>** → **<BEAMERMODUS>**. Ist der Modus aktiv, so erscheint neben diesem Menüeintrag ein Häkchen. Um den Modus wieder auszuschalten, entfernen Sie einfach den Haken per Klick.



Das Menü **<AVL-BÄUME>** mit dem Eintrag **<BEAMERMODUS>**

## 2.9. Impressum

Das Modul **AVL-Bäume** wurde im Sommersemester 2005 von der Praktikumsgruppe SWT05-PROG1 im Rahmen des externen Softwarepraktikums entwickelt. Mitwirkende waren die

### Teammitglieder

- Alexander Claus — Chefprogrammierer
- Ulrike Fischer — Assistent
- Sebastian Pape — Administrator
- Jean Christoph Jung — Testverantwortlicher
- Matthias Schmidt — Sekretär

sowie der betreuende Tutor Marco Zimmerling.

Die Webseite des Projektes finden Sie unter <http://web.inf.tu-dresden.de/~swt05-p1/>.

Das Handbuch zu diesem Modul wurde erstellt von Matthias Schmidt und Jean Christoph Jung.

## 2. *Das Modul AVL-Bäume*

## 3. Das Modul Dijkstra

### 3.1. Einleitung

Das Modul **Dijkstra** visualisiert den bekannten Algorithmus von E. W. Dijkstra zum Finden der kürzesten Wege von einem Startknoten in einem Distanzgraphen. Der Algorithmus selbst ist unter anderem im Vorlesungsskript von Prof. Vogler „Algorithmen, Datenstrukturen und Programmierung“ zu finden. Aber auch im Internet existieren zahlreiche Quellen dazu.

Soweit es möglich gewesen ist, wurde beim Design des Moduls darauf geachtet, es weitgehend intuitiv und selbst-dokumentierend zu gestalten. Nichtsdestotrotz findet sich hier eine kurze Einführung in das **Dijkstra** - Modul.

### 3.2. Funktionsübersicht

Das Modul **Dijkstra** realisiert folgende Funktionen:

- graphisches Erstellen / Bearbeiten eines Distanzgraphen
- Erstellen / Bearbeiten eines Graphen mittels Kanten- / Knotenliste oder Adjazenzmatrix
- Speichern und Laden von Graphen
- Visualisierung des Dijkstra-Algorithmus

### 3.3. Modul starten

Um das Modul zu starten, wählt man im Menü <DATEI> das Submenü <NEU> und dann den Menübefehl **DIJKSTRA** . Im Hauptfenster erscheint nun die Oberfläche des **Dijkstra** - Moduls im Eingabe-Modus.

### 3.4. Symbolleiste

Die Symbolleiste stellt die Funktionen **SPEICHERN**, **SPEICHERN UNTER**, **RÜCKGÄNGIG** und **WIEDERHERSTELLEN** bereit.

### 3.5. Graph erstellen

Nach dem Start des Moduls wird die Oberfläche für das Erstellen eines Graphen angezeigt. Sie ist in die Bereiche

1. „Werkzeuge“
2. „Graph“
3. „Knotenliste“
4. „Kantenliste“
5. „Distanzmatrix“

aufgeteilt.

#### 3.5.1. Graphische Eingabe/Erstellen eines Graphen per Maus

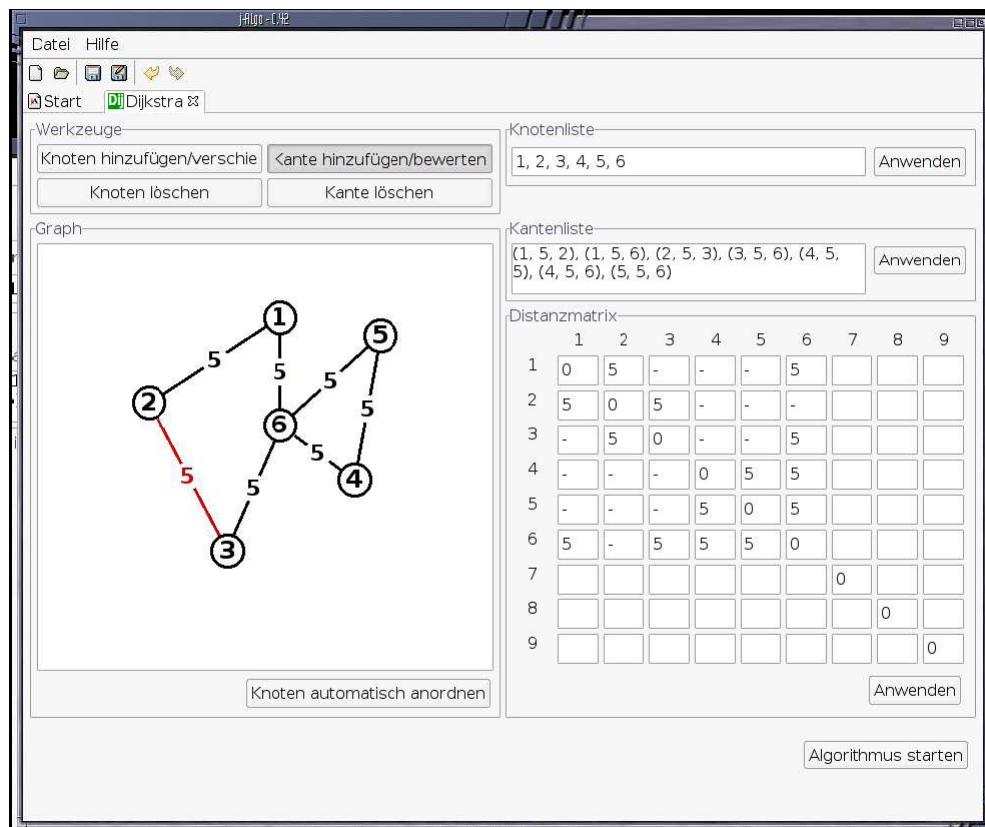
Das Erstellen eines Graphen per Maus wird durch die Werkzeuge

1. „Knoten hinzufügen/verschieben“ — Durch Klicken auf die Zeichenfläche wird ein neuer Knoten erzeugt. Ein bestehender Knoten kann durch Ziehen mit der Maus bewegt werden.
2. „Kante hinzufügen/bewerten“ — Indem man die Maus von einem Knoten zu einem anderen zieht, entsteht zwischen ihnen eine neue Kante. Die Kantenbewertung wird geändert, wenn man sie herauf- bzw. hinunterzieht.
3. „Knoten löschen“ — Ein angeklickter Knoten wird gelöscht.
4. „Kante löschen“ — Eine angeklickte Kante wird gelöscht.

unterstützt. Dabei geht man wie folgt vor: Nach der Auswahl des Werkzeugs „Knoten hinzufügen / verschieben“ kann man durch einfaches Klicken auf die weiße Zeichenfläche Knoten erstellen. Vorhandene Knoten können mit Drag&Drop verschoben werden.

Nachdem man alle Knoten angelegt hat, kann man nach Auswahl des Werkzeugs „Kante hinzufügen/bewerten“ den Graphen vervollständigen. Um eine Kante zu erstellen, klickt man erst den „Startknoten“ und dann den „Endknoten“ der Kante an. Es erscheint eine Kante zwischen den Knoten mit der Bewertung fünf. Diese Bewertung (auch Kantengewicht) kann verändert werden, indem man das Kantengewicht mit der Maus „festhält“ und nach oben (das Gewicht wird größer) oder unten (das Gewicht wird kleiner) zieht.

### 3.5. Graph erstellen



Graphisches Erstellen eines Graphen

#### 3.5.2. Die Knotenliste

Die Knotenliste zeigt die Indizes aller Knoten durch Kommata getrennt. Durch Hinzufügen von Indizes werden auch neue Knoten erzeugt. Änderungen in der Knotenliste werden nach Betätigen der Schaltfläche „Anwenden“ übernommen.

#### 3.5.3. Die Kantenliste

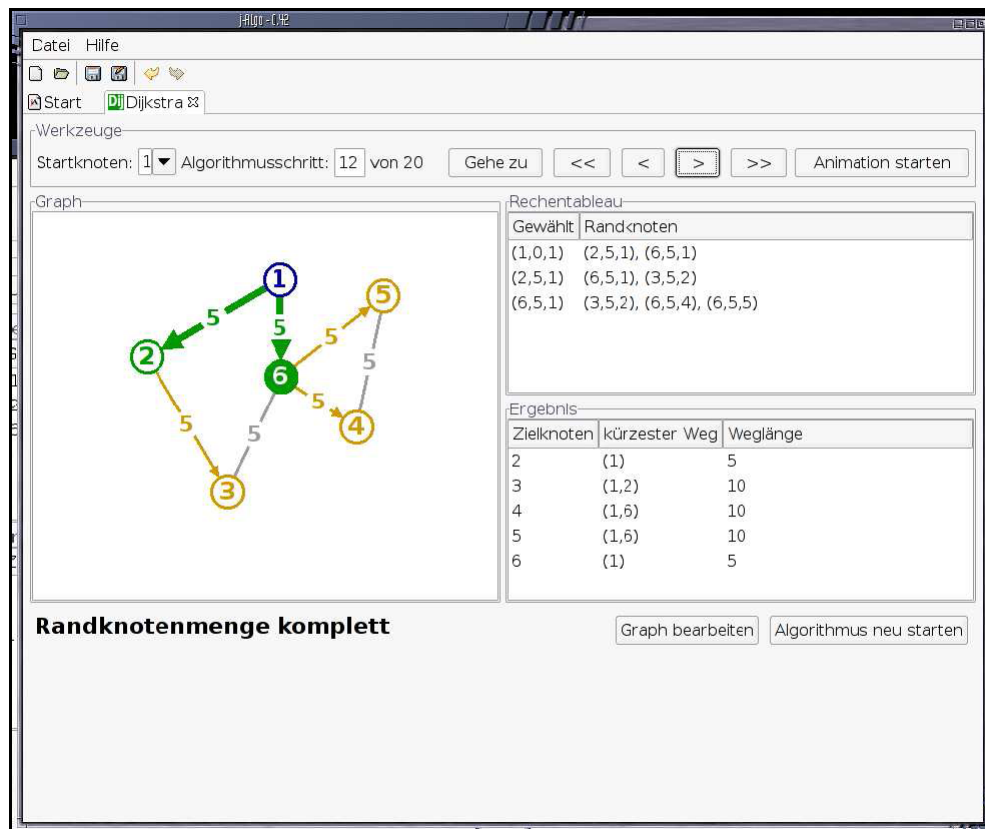
Die Kantenliste zeigt alle Kanten des Graphen im Format ( VON, WEG, ZU ). Durch Editieren dieser Liste können bestehende Kanten geändert und neue hinzugefügt werden. Auch hier ist zu beachten, dass Änderungen erst durch Klicken von „Anwenden“ übernommen werden.

#### 3.5.4. Die Adjazenzmatrix

Die Adjazenzrelation des Graphen ist in dieser Matrix dargestellt. Kanten und Knoten können in jener durch einfaches Eingeben einer Kantenbewertung erzeugt werden. Dabei muss nicht beachtet werden, daß die Matrix symmetrisch bleibt, da dies automatisch gewährleistet wird. Nach dem Bearbeiten der Matrix darf nicht vergessen werden, den „Anwenden“ - Button zu betätigen, damit die Änderungen übernommen werden.

### 3. Das Modul Dijkstra

#### 3.6. Ablauf des Algorithmus



Der Algorithmus läuft

Ist man mit dem Aussehen seines Graphen zufrieden, wird durch Auswählen des „Algorithmus starten“-Buttons in den Algorithmus-Modus des Moduls gewechselt. Der Graph erscheint nun zuerst grau und ändert sich farblich im weiteren Verlauf, um die verschiedenen Zustände des **Dijkstra** -Algorithmus darzustellen.

Um die einzelnen Schritte des Algorithmus abzuarbeiten, kann man man mit den „<-“ und „>-“Buttons zum vorherigen bzw. nächsten Schritt springen. „<<“ und „>>“ überspringen gleich mehrere Schritte.

Falls man sich entschließt, einen anderen Graphen untersuchen zu wollen, kann mit „Graph bearbeiten“ in den Editiermodus zurückgekehrt werden. „Algorithmus neu starten“ springt zurück an den Anfang des Algorithmus.



## 3.7. Impressum

Das Modul **Dijkstra** wurde im Sommersemester 2005 von der Praktikumsgruppe SWT05-PROG2 im Rahmen des externen Softwarepraktikums entwickelt. Mitwirkende waren die

### Teammitglieder

- Frank Staudinger — Chefprogrammierer
- Julian Stecklina — Assistent
- Hannes Straß — Administrator
- Martin Winter — Testverantwortlicher
- Steven Voigt — Sekretär

sowie der betreuende Tutor Marco Zimmerling.

Die Webseite des Projektes finden Sie unter <http://web.inf.tu-dresden.de/~swt05-p2/>.

### 3. *Das Modul Dijkstra*

## 4. Das Modul KMP

### 4.0.1. Einleitung

Dieses Modul behandelt den Suchalgorithmus '*Knuth Morris Pratt*' (KMP) und zeigt Arbeitsweise am Pattern und Text. Für eine detaillierte Beschreibung des Algorithmus sei auf das Vorlesungsskript von Prof. Vogler '*Algorithmen, Datenstrukturen und Programmierung*' verwiesen.

### 4.0.2. Funktionsübersicht

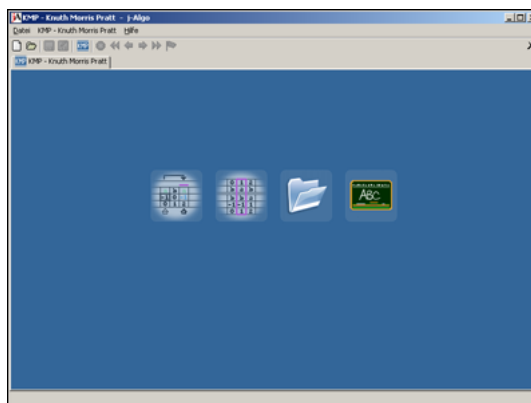
Das Modul KMP realisiert folgende Funktionen:

- Visualisierung und Erläuterung des Algorithmus
- Manuelle Eingabe von Pattern und Text
- Generieren von Pattern und passenden Texten
- Erstellen der Verschiebetabelle
- Suchen im Suchtext
- Präsentation von Lernbeispielen
- Speichern und Laden von KMP-Beispielen

## 4.1. Der Willkommensbildschirm

### 4.1.1. Modul KMP starten

Nach Starten des Hauptprogramms j-Algo können Sie über den Button 'Neu' oder mit dem Menüpunkt 'Datei' => 'Neu' eine neue Instanz des Moduls KMP öffnen. Anschließend öffnet sich der Willkommensbildschirm des Moduls, der Ihnen verschiedene Möglichkeiten eröffnet.



## 4. Das Modul KMP

Der Willkommensbildschirm des Moduls

### 4.1.2. Generierung der Verschiebetabelle

Mit Klick auf das erste Symbol gelangen Sie zur ersten Phase des KMP-Algorithmus, in dem für ein Pattern die Verschiebetabelle erstellt wird.

### 4.1.3. Suchen im Suchtext

Mit Klick auf das zweite Symbol gelangen Sie zur zweiten Phase des KMP-Algorithmus, in dem ein Text nach einem Pattern durchsucht wird.

### 4.1.4. Öffnen einer KMP-Sitzung

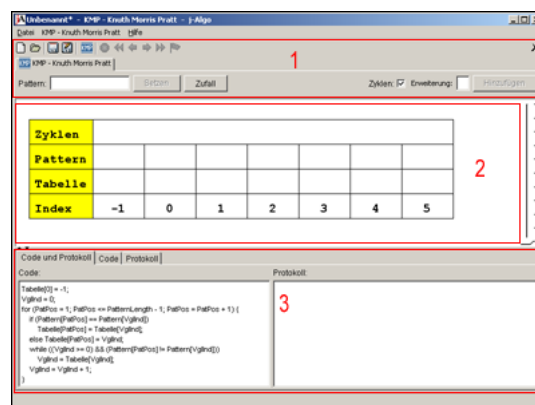
Mit Klick auf das Ordner-Symbol erhalten Sie die Möglichkeit, ein zuvor gespeichertes KMP-Beispiel zu laden (\*.jalgo Datei)

### 4.1.5. Präsentation von Lernbeispielen

Mit Klick auf das Tafel-Symbol erhalten Sie die Möglichkeit, anhand gegebener repräsentativer Lernbeispiele die Funktions- und Arbeitsweise des Algorithmus kennenzulernen.

## 4.2. Der Arbeitsbereich

Der Arbeitsbereich ist untergliedert in drei Bereiche, die Ihnen Zugriff auf alle wesentlichen Funktionen in den Algorithmus-Phasen ermöglicht:



Phase 1 Bildschirm

#### 1. Steuerung

Hier wird der Algorithmus gestartet und gesteuert.

#### 2. Visualisierung

Hier werden die Verschiebetabelle bzw. der Text dargestellt.

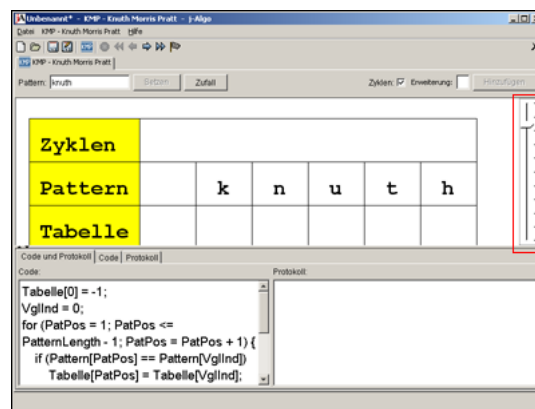
### 3. Dokumentation

Hier gibt es verschiedene Perspektiven, die miteinander kombiniert werden können: Quellcode, Text, Protokoll.

## 4.3. Anzeigeoptionen

### 4.3.1. Skalierung

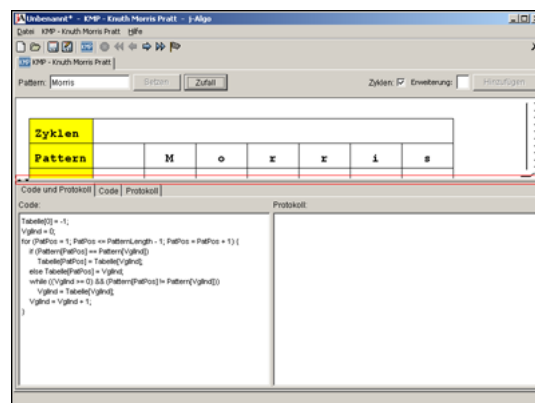
Die Größe der Elemente des Visualisierungs- und Dokumentationsbereichs kann eingestellt werden, klicken Sie dafür auf den Schieberegler im Visualisierungsbereich und ziehen Sie ihn nach oben bzw. unten.



Auswirkung des Schiebereglers zur Skalierung

### 4.3.2. Aufteilung der Bereiche

Die Aufteilung zwischen dem Visualisierungs- und dem Dokumentationsbereich kann mit einem Schieberegler verändert werden. Per Klick auf die Kante können Sie die Grenze verschieben.

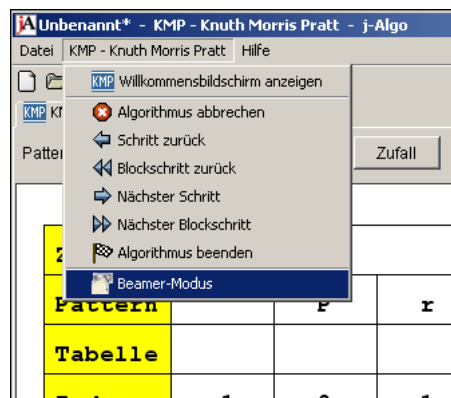


Auswirkung des Schiebereglers

## 4. Das Modul KMP

### 4.3.3. Beamer - Modus

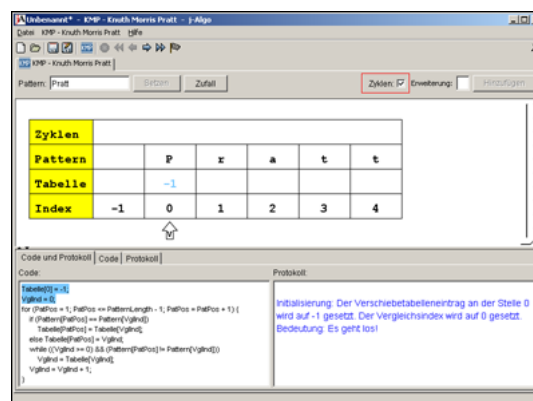
Der Beamer-Modus ermöglicht das schnelle Einstellen von Anzeigeeoptionen, die die Präsentation in Vorlesungen oder ähnlichen Veranstaltungen begünstigen. Dieser Modus ist zu finden unter 'Knuth Morris Pratt' => 'Beamermodus'. Die Anzeige ist für die Auflösung 1024x768 optimiert und vergrößert die Elemente um den Faktor 1,5. Im Dokumentationsbereich wird die Perspektive 'Code' angezeigt. Ist der Modus aktiv, so erscheint vor diesem Menüeintrag ein Häkchen. Um den Modus wieder auszuschalten, entfernen Sie einfach den Haken per Klick.



Der Beamermodus

### 4.3.4. Zyklus - Anzeige

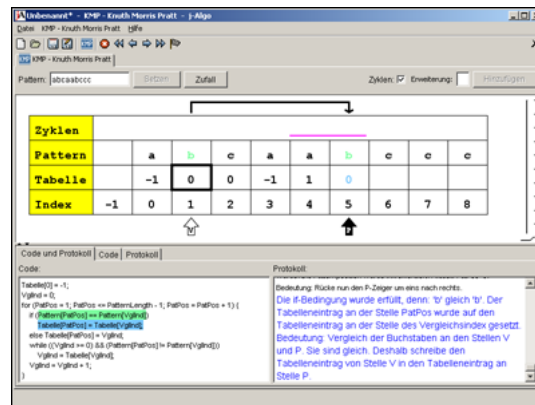
Im Visualisierungsbereich der Phase 1 'Generierung der Verschiebetabelle' können optional die Zyklen angezeigt werden, setzen Sie dazu das Häkchen 'Zyklen' im Steuerungsbereich. Es werden maximal drei Zyklen gleichzeitig angezeigt.



Beispiel für Zyklen

## 4.4. Legende

### 4.4.1. Elemente in der Phase 'Generierung der Verschiebetabelle'



Beispiel für Phase 1

#### Visualisierungsbereich

**Pfeil schwarz mit weißem 'P'** - Zeiger auf die Variable 'patpos', die Patternposition

**Pfeil weiß mit schwarzem 'V'** - Zeiger auf die Variable 'VglInd', der Vergleichsindex

**schwarzer Pfeil über der Tabelle** - die verglichenen Zeichen

**gelber Hintergrund von Zellen** - Zellenkopf

**schwarzer Rahmen um Zelle in der Zeile 'Index'** - aktuell kopierte Verschiebeinformation

**roter Rahmen um Zellen** - negative boolesche Bedingung

**lila Strich** - Zyklen, die im Pattern auftreten

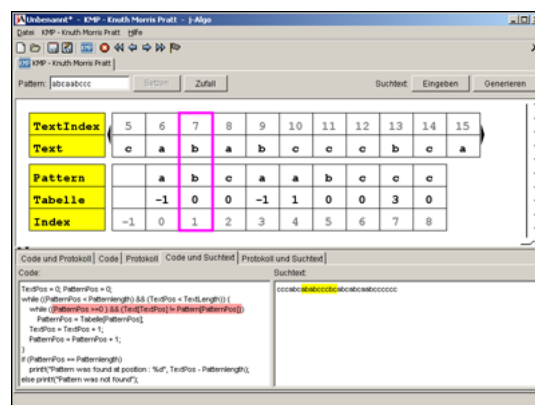
#### Schriftfarben

**blau** - aktuell geschriebene Verschiebeinformation (entspricht Zuweisungsstatement im Code)

**rot** - die verglichenen Zeichen stimmen nicht überein (entspricht booleschem Statement im Code)

**grün** - die verglichenen Zeichen stimmen überein (entspricht booleschem Statement im Code)

### 4.4.2. Elemente in der Phase 'Suchen im Suchtext'



#### 4. Das Modul KMP

Beispiel für diese Phase

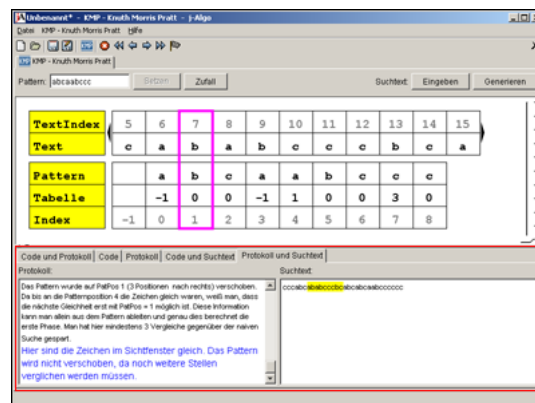
##### Visualisierungsbereich

**lila Rahmen** - das Sichtfenster, welches die aktuell betrachteten Zeichen des Textes und des Patterns justiert

##### Schriftfarben

**blau** - aktuell geschriebene Verschiebeinformation (entspricht Zuweisungsstatement im Code)

#### 4.4.3. Im Dokumentationsbereich



Beispiel für diese Phase

##### Perspektive 'Code'

**roter Hintergrund** - negative boolesche Bedingung

**grüner Hintergrund** - positive boolesche Bedingung

**blauer Hintergrund** - sonstige Statements

##### Perspektive 'Protokoll'

**blaue Schrift** - zuletzt vollzogener Schritt

**schwarze Schrift** - für den aktuellen Stand irrelevante vorausgegangene Schritte

##### Perspektive 'Suchtext'

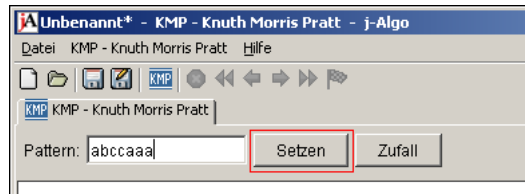
**gelber Hintergrund** - der Textausschnitt, an dem das Pattern momentan anliegt



## 4.5. Modulfunktionen

### 4.5.1. Eingabe von Pattern und Text

#### Pattern eingeben

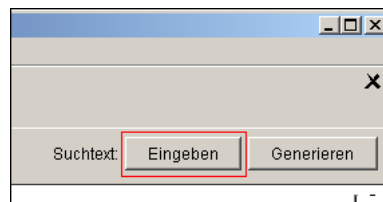


Pattern manuell eingeben

Das Pattern kann manuell eingegeben werden, hierbei ist die maximale Länge von 10 Zeichen zu beachten. Das Alphabet kann aus beliebigen Zeichen bestehen. Die Eingabe erfolgt im Steuerungsbereich in die Eingabezeile 'Pattern'. Um das Pattern zu übernehmen klicken Sie auf 'Setzen'.

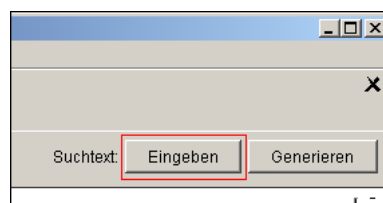
#### Suchtext eingeben

Der Text unterliegt keinen Beschränkungen und kann manuell eingegeben oder aus einer vorhandenen \*.txt-Datei importiert werden. Die Eingabe erfolgt im Steuerungsbereich, klicken Sie hierfür auf 'Eingeben'.



Text manuell eingeben

Es öffnet sich ein Fenster, in dem Sie den Text manuell eingeben (oder auch per Copy und Paste einfügen) oder eine \*.txt-Datei laden können. Um den Text zu übernehmen klicken Sie auf 'Anwenden'.

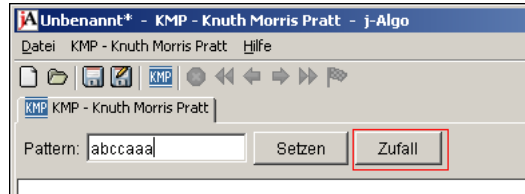


Text laden oder eingeben

## 4. Das Modul KMP

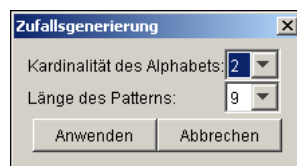
### 4.5.2. Generieren von Pattern und passenden Texten

#### Pattern eingeben



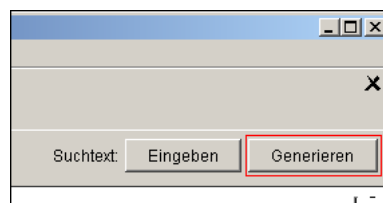
Pattern generieren lassen

Neben der manuellen Eingabe des Patterns gibt es die Möglichkeit, ein Pattern generieren zu lassen. Klicken Sie dafür auf den Knopf 'Zufall' (im Steuerungsbereich neben der Eingabezeile 'Pattern') und wählen Sie im folgenden Fenster die Kardinalität des zu nutzenden Alphabets und die gewünschte Länge des Patterns aus. Das Pattern wird mit Klick auf 'Anwenden' generiert und Sie kehren zum Arbeitsbereich zurück.



Pattern generieren lassen

#### Suchtext eingeben



Suchtext generieren lassen

Um einen zum Pattern passenden Text generieren zu lassen, klicken Sie auf 'Generieren' (im Steuerungsbereich neben 'Text'). Dadurch wird eine Text erstellt, der das Pattern enthält und nur aus den im Pattern genutzten Zeichen besteht. Dieser Text wird sofort übernommen, Sie haben aber die Möglichkeit, ihn zu bearbeiten, indem Sie auf 'Eingeben' klicken.

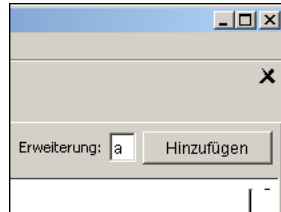
### 4.5.3. Generierung der Verschiebetabelle

Wenn ein Pattern gesetzt wurde, kann die Verschiebetabelle erstellt werden. Um den Algorithmus zu starten und zu steuern, benutzen Sie die Pfeile im Steuerungsbereich.



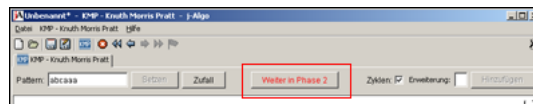
Algorithmussteuerung

Das Pattern kann nun noch erweitert werden ohne den aktuellen Algorithmus zu unterbrechen, geben Sie hierfür das gewünschte Zeichen in das Eingabefeld Erweiterung im Steuerungsbereich ein. Sollte das Pattern bereits die maximale Länge haben, ist eine Erweiterung nicht möglich.



Pattern erweitern

Nachdem die Verschiebetabelle erstellt wurde, können Pattern und Verschiebetabelle für eine Suche im Text genutzt werden. Klicken Sie hierfür auf das Feld 'Weiter zu Phase 2', welches erst am Ende des Algorithmus im Steuerungsbereich erscheint.



Von Phase 1 in Phase 2 wechseln

#### 4.5.4. Suchen im Suchtext

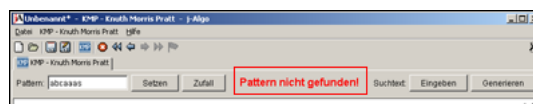
Erst wenn Pattern und Text gesetzt sind, kann das Anwenden der Verschiebetabelle an einem Text durchgeführt werden. Um Pattern und Text zu setzen, haben Sie verschiedene Möglichkeiten, siehe: Eingabe von Pattern und Text, Generieren von Pattern und passenden Texten. Um den Algorithmus zu starten und zu steuern, benutzen Sie die Pfeile im Steuerungsbereich.



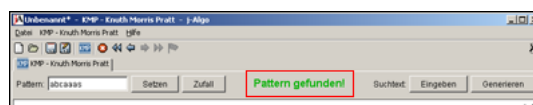
Algorithmussteuerung

Sollten Sie während der Durchführung des Algorithmus das Pattern und/oder den Text ändern, wird die aktuelle Durchführung des Algorithmus unterbrochen. Der Suchvorgang wird beendet,

wenn das Pattern gefunden wurde oder das Textende erreicht wurde, es erscheint eine Meldung dazu im Steuerungsbereich.



Nachricht, dass Pattern nicht gefunden wurde

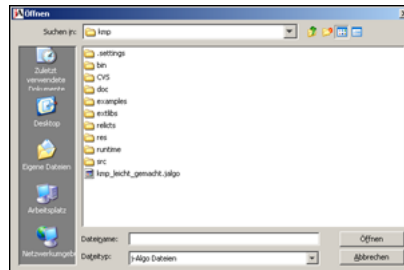


Nachricht, dass Pattern gefunden wurde

## 4. Das Modul KMP

### 4.5.5. Öffnen einer KMP-Sitzung

Mit Klick auf das Ordner-Symbol öffnet sich ein Dialogfenster, in dem Ihnen die Möglichkeit gegeben wird, eine '\*.jalgo' - Datei auszuwählen, in welcher eine KMP-Sitzung gespeichert wurde.

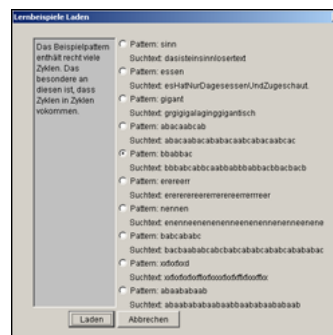


Dialogfenster zum Laden einer Sitzung

### 4.5.6. Präsentation von Lernbeispielen

Es stehen Ihnen zehn Beispiele zur Auswahl, die jeweils eine besondere Eigenschaft des KMP-Algorithmus repräsentieren. Wenn Sie mit dem Mauszeiger über die Beispiele fahren, werden diese Eigenschaften kurz beschrieben. Wählen Sie das gewünschte Lernbeispiel aus und klicken Sie auf 'Laden' um das Beispiel zu starten.

Die Beispiele bestehen aus Pattern- und passendem Text.



Präsentation von Lernbeispielen

Die Steuerung der Lernbeispiele erfolgt durch die Pfeile im Steuerungsbereich.

## 4.6. Algorithmussteuerung



Der Willkommensbildschirm des Moduls

**Startbildschirm** Hierbei wird der aktuelle Arbeitsbereich verlassen und der Benutzer kehrt zum Startbildschirm zurück.

**Abbruch** Hierbei wird zum Anfang des Algorithmus gesprungen.

**'Schnelles' Zurück** Hierbei wird ein Schritt vollzogen, der mehreren vorherigen Statements im Quelltext entspricht.

**Zurück** Hierbei wird einer Schritt vollzogen, der dem vorherigen Statement im Quelltext entspricht.

**Weiter** Hierbei wird ein Schritt vollzogen, der dem nächsten Statement im Quelltext entspricht.

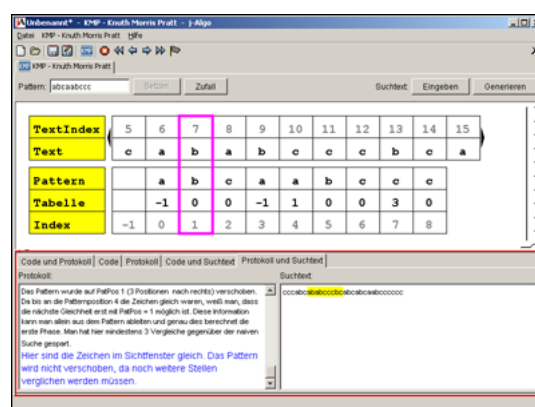
**'Schnelles' Weiter** Hierbei wird ein Schritt vollzogen, der mehreren nächsten Statements im Quelltext entspricht. Beim Erstellen der Verschiebetabelle entspricht dieser Schritt der Berechnung der Verschiebeinformation je eines Zeichens. Beim Anwenden der Verschiebetabelle an einem Text entspricht dieser Schritt dem Verschieben des Patterns unter Berücksichtigung der Verschiebetabelle.

**Ende** Hierbei werden alle Schritte, die bis zum Ende des Algorithmus durchgeführt werden, vollzogen.

## 4.7. Dokumentation

Im Dokumentationsbereich stehen verschiedene Perspektiven und ihre Kombinationen zur Auswahl, die Aktivierung der Perspektive erfolgt über Reiter.

- **Code** - der Quellcode des Algorithmus, entnommen dem Vorlesungsskript von Prof. Vogler 'Algorithmen, Datenstrukturen und Programmierung'.
- **Protokoll** - jeder vom Benutzer durchgeführte Schritt wird aufgelistet, egal ob es sich um Vorwärts- oder Rückwärts-Schritte handelt.
- **Suchtext** - der zu durchsuchende Text.



Der Dokumentationsbereich

Die Größe sowie die Aufteilung des Dokumentationsbereichs lassen sich vom Benutzer einstellen, siehe Anzeigoptionen.

Die Bedeutung der verwendeten Farben wird in der Legende erklärt.

#### 4. *Das Modul KMP*

## 5. Das Modul Ebnf und Syntaxdiagramme

### 5.1. Einleitung und Funktionsübersicht

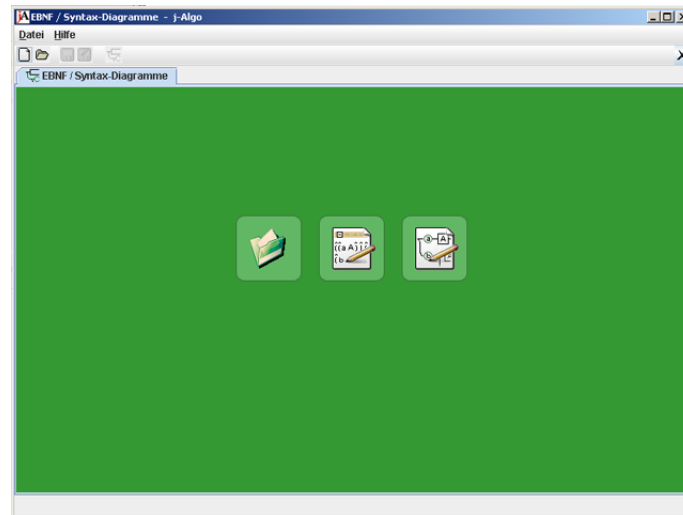
Das **j-Algo** Modul **EBNF und Syntaxdiagramme** erweitert **j-Algo** um die Möglichkeit, EBNF Definitionen und Syntaxdiagramme zu erzeugen und bearbeiten.

Darüberhinaus stellt das Modul die Möglichkeit bereit, EBNF-Definitionen in Syntaxdiagramme zu überführen; dies geschieht mit dem `trans()`-Algorithmus, der im Modul schrittweise nachvollzogen werden kann.

Außerdem wird eine graphische Darstellung des Rücksprungalgorithmus zur Verfügung gestellt, mit deren Hilfe der Benutzer Worte der von einem Syntaxdiagramm definierten Sprache erzeugen kann.




## 5. Das Modul Ebnf und Syntaxdiagramme

### 5.2. Der Willkommensbildschirm



Der Willkommensbildschirm

Der Willkommensbildschirm des **EBNF und Syntaxdiagramme** -Moduls bietet eine Auswahl der verschiedenen Möglichkeiten des Moduls:

-  Laden einer EBNF-Definition oder eines Syntaxdiagramms  
Wenn die Definition bzw. das Diagramm vollständig und korrekt ist, gelangt man in die jeweilige Anzeige, ansonsten in den Editor.
-  Erstellen einer neuen EBNF-Definition  
Hier gelangt man zum Editor für EBNF-Definitionen
-  Erstellen eines neuen Syntaxdiagramms  
Diese Schaltfläche führt zum Editor für Syntaxdiagramme



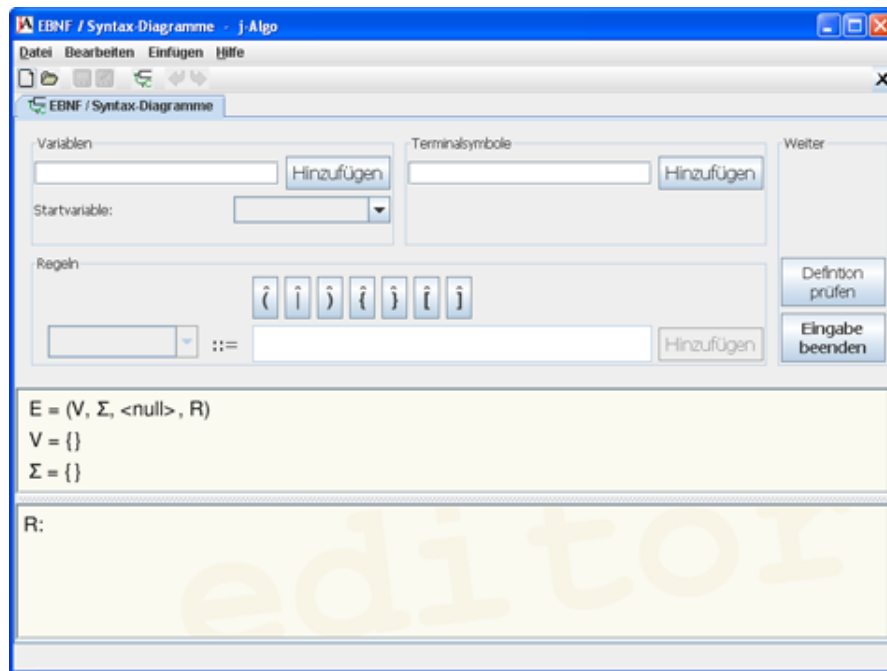
## 5.3. EBNF-Definitionen

### 5.3.1. Der EBNF-Editor

#### Die Arbeitsfläche

Die Arbeitsfläche der EBNF-Eingabe teilt sich in eine obere und eine untere Hälfte. Die obere Hälfte dient der Eingabe der Definition, die untere der Ansicht.

Ein nachträgliches Bearbeiten der Definition wird durch die Interaktivität der Definition im unteren Bereich ermöglicht.



Die Arbeitsfläche des Moduls **EBNF und Syntaxdiagramme**

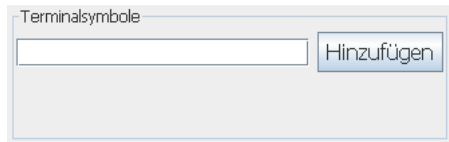
#### Eingabe einer Definition

Die Definition wird im oberen Bildschirmbereich eingegeben. Es stehen Steuerelemente für die Eingabe von Variablen, Terminalsymbolen sowie Regeln zur Verfügung.

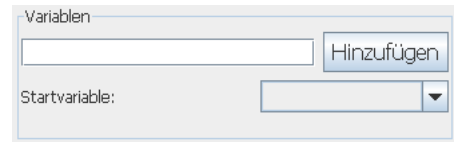
- Eingabe von Variablen und Terminalsymbolen
  - Zugelassene Symbole  
Grundsätzlich sind alle Symbole zugelassen, die die folgenden Bedingungen erfüllen:
    - \* Ein Symbol darf nicht der Anfang eines anderen Terminalsymbols oder einer anderen Variable sein
    - \* Ein Symbol darf nicht mit Leerzeichen beginnen oder enden
    - \* Ein Symbol darf nicht leer sein
    - \* Ein Symbol darf keine Metasymbole enthalten

## 5. Das Modul Ebnf und Syntaxdiagramme

- Hinzufügen von Terminalsymbolen und Variablen zur EBNF-Definition  
Das hinzuzufügende Symbol muss in das entsprechende Textfeld geschrieben werden und kann dann mit der Eingabetaste oder Klick auf Hinzufügen zur Definition hinzugefügt werden. Falls das Symbol nicht hinzugefügt werden kann, wird eine entsprechende Fehlermeldung angezeigt.



Eingabemaske für Terminalsymbole



...und für Variablen

- Eingabe von Regeln

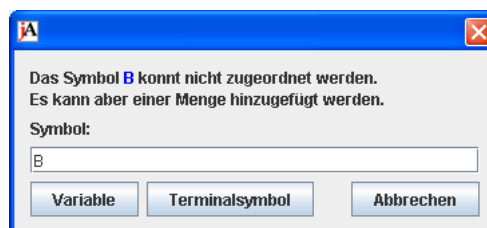
Mit einer EBNF-Regel wird einer Variablen ein EBNF-Term zugeordnet. Die Variable kann aus der Drop-Down-Liste ausgewählt werden. Da es zu jeder Variablen nur eine Regel geben darf, werden nur diejenigen Variablen angezeigt, zu denen es noch keine Regel gibt.

- Für einen korrekten EBNF-Term gelten folgende Regeln:
  - \* Ein EBNF-Term besteht aus Variablen, Terminalsymbolen und Metasymbolen
  - \* Die Metasymbole müssen korrekt geklammert sein
  - \* Der EBNF-Term darf nicht leer sein (genauso darf der in einer Klammerung enthaltene Sub-Term nicht leer sein)
- Eine neue Regel kann mit der Eingabetaste oder Klick auf Hinzufügen zur Definition hinzugefügt werden. Falls die Regel nicht hinzugefügt werden kann, wird eine entsprechende Fehlermeldung angezeigt.



Eingabemaske für Regeln

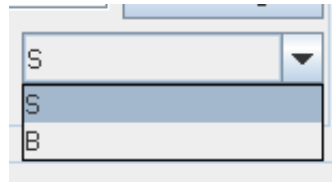
- Kommt in der Regel ein unbekanntes Symbol vor, so wird ein Dialogfenster angezeigt, welches die Möglichkeit bietet, das neue Symbol der Menge der Variablen oder Terminalsymbole hinzuzufügen. Man kann auch Teile des nicht erkannten Textes als Symbol hinzufügen, indem man den Text im Dialog bearbeitet. In diesem Fall ist es möglich, dass weitere unbekannte Symbole gefunden werden. Wählt man in diesem Dialog "Abbrechen", so schlägt das Hinzufügen der Regel fehl.



Dialog für unbekannte Symbole

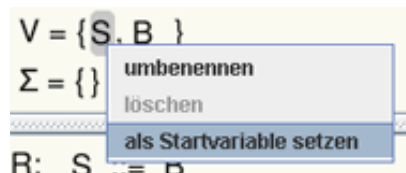
- Setzen der Startvariable

- Eine Definition besitzt genau dann eine Startvariable, wenn sie mindestens eine Variable enthält. Beim Hinzufügen der ersten Variablen wird diese automatisch zur Startvariable.



Setzen der Startvariable

- Die Startvariable kann geändert werden, indem man eine Variable aus der entsprechenden Drop-Down-Liste auswählt. Außerdem kann man eine Variable über ihr Kontextmenü in der Anzeige der Variablenmenge als Startvariable setzen.



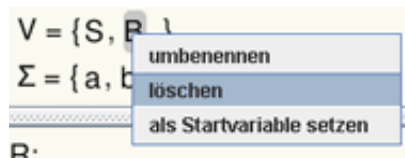
Setzen der Startvariable in der Ansicht

### Bearbeiten einer Definition

Möchte man an einer bereits eingegebenen Definition Änderungen vornehmen, zum Beispiel eine Regel löschen oder ein Terminalsymbol umbenennen, so kann dies über das Kontextmenü in der Ansicht geschehen.

- Löschen

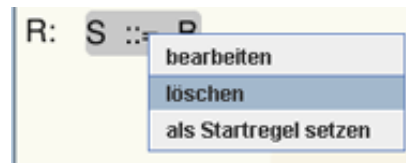
- Löschen von Variablen und Terminalsymbolen
  - \* Ein Symbol kann nur dann gelöscht werden, wenn es in keiner Regel mehr vorkommt.
  - \* Wenn dies der Fall ist, kann man im Kontextmenü des Symbols in der Variablen- oder Terminalsymbolmenge die Funktion „Löschen“ auswählen.



Löschen einer Variable

- Löschen von Regeln
  - Regeln können über das Kontextmenü in der Regel-Ansicht gelöscht werden.

## 5. Das Modul Ebnf und Syntaxdiagramme

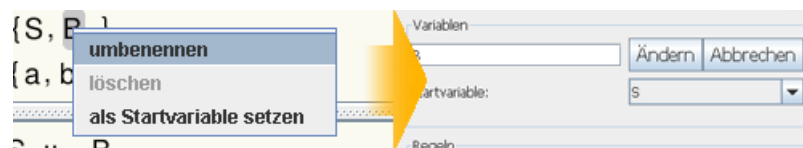


Löschen einer Regel

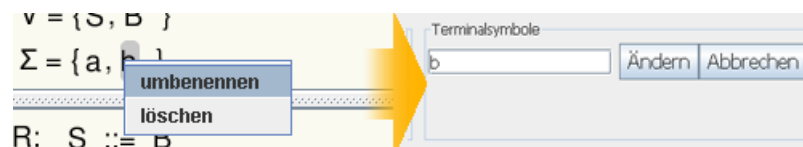
- Bearbeiten

- Bearbeiten von Variablen und Terminalsymbolen.

- \* Symbole können über ihr Kontextmenü umbenannt werden. Dabei gelten für die Namensänderung die gleichen Regeln wie für die Eingabe.
- \* Wählt man im Kontextmenü eines Symbols “Bearbeiten” aus, so erscheint in der entsprechenden Eingabemaske das Symbol, der Hinzufügen-Button ändert sich in einen “Ändern”-Button und es erscheint zusätzlich ein “Abbrechen”-Button. Im Textfeld kann man den Symbolnamen ändern und anschließend durch die Auswahl von “Ändern” übernehmen. Möchte man die Änderung abbrechen, so kann man dies über den entsprechenden Button vornehmen.
- \* Grundsätzlich kann man immer nur eine Variable und ein Terminalsymbol gleichzeitig bearbeiten; im Moment bearbeitete Symbole werden in der Ansicht gelb hervorgehoben.



Umbenennen einer Variable



Umbenennen eines Terminalsymbols

- Bearbeiten von Regeln

- \* Regeln können über ihr Kontextmenü bearbeitet werden.
- \* Wählt man im Kontextmenü einer Regel “Bearbeiten” aus, so erscheint in der Regel-Eingabemaske die Regel, der Hinzufügen-Button ändert sich in einen “Ändern”-Button und es erscheint zusätzlich ein “Abbrechen”-Button. Im Textfeld kann man den Term bearbeiten und anschließend durch die Auswahl von “Ändern” übernehmen. Das Ändern der Variablen der Regel ist nicht möglich. Möchte man die Änderung abbrechen, so kann man dies über den entsprechenden Button vornehmen.



Bearbeiten einer Regel

- \* Grundsätzlich kann man immer nur eine Regel bearbeiten; die im Moment bearbeitete Regel wird in der Ansicht gelb hervorgehoben.

### Überprüfen einer Definition und Beenden der Eingabe

Um eine Definition zu überprüfen kann der Button “Definition überprüfen” genutzt werden. Die Definition wird dann auf folgende Fehler und mögliche Probleme überprüft:

- Das Vorhandensein einer Startvariable (→ Fehlermeldung)
- Das Vorhandensein einer Regel für jede Variable (→ Fehlermeldung)
- Die Nutzung aller Variablen der Variablenmenge (→ Warnung)
- Die Nutzung aller Terminalsymbole der Terminalsymbolmenge (→ Warnung)
- Die Erreichbarkeit aller Regeln von der Startvariablen aus (→ Warnung)

Wird mind. ein Fehler gefunden, so werden Warnungen ignoriert und nicht angezeigt, um die Übersicht zu wahren.

Durch das Aktivieren des EINGABE BEENDEN-Buttons wird die Definition überprüft. Wird dabei mind. ein Fehler gefunden, so kann der Wechsel zur EBNF-Anzeige nicht stattfinden. Wenn kein Fehler auftritt (Warnungen können vorkommen, werden aber nicht angezeigt) geschieht ein Wechsel zur EBNF-Anzeige, in welcher die Eingabemaske nicht mehr zu sehen ist.

#### 5.3.2. Anzeige von EBNF-Definitionen

Die EBNF-Anzeige kann nur erreicht werden, wenn die Definition korrekt und vollständig ist. Eine Überprüfung dessen findet durch die vorausgegangene Nutzung des EINGABE BEENDEN-Buttons bzw. Beim Laden einer Datei statt.

Hintergrund der EBNF-Anzeige ist die übersichtliche Anzeige einer EBNF-Definition, ohne die Eingabemaske. Außerdem besteht hier die Möglichkeit, zum trans()-Algorithmus zu wechseln, um die EBNF-Definition in ein Syntaxdiagramm umzuwandeln.

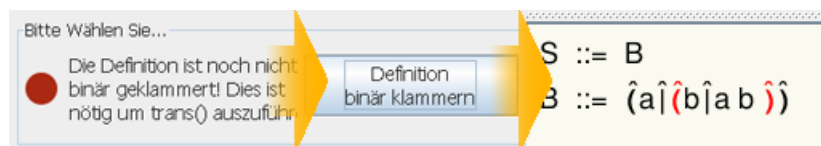
## 5. Das Modul Ebnf und Syntaxdiagramme

### Definition binär klammern

Um in den trans()-Algorithmus wechseln zu können, muss die Definition binär geklammert sein. Nach strenger Auslegung der Definition existieren nur binäre Alternativen in der EBNF:

- nicht binär geklammert:  $(a|b|c)$
- korrekt geklammert:  $(a|(b|c))$

Der Lesbarkeit halber ermöglicht das Programm eine Eingabe von nicht binär geklammerten Alternativen. Ob die Definition noch nicht korrekt geklammerte Terme enthält, wird durch eine entsprechende Meldung angezeigt:



Definition binär klammern

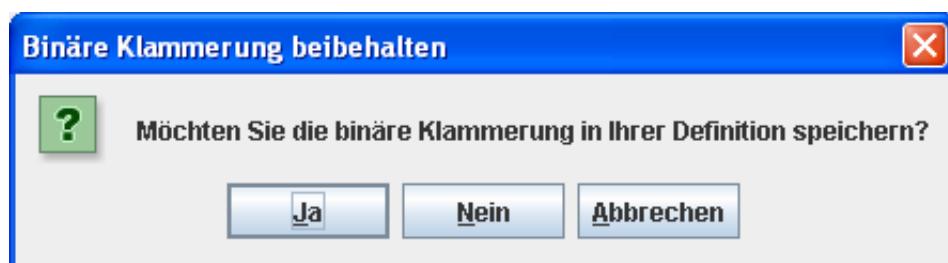
Um eine Definition binär zu klammern, muss der DEFINITION BINÄR KLAMMERN-Button genutzt werden. Dadurch werden die noch fehlenden Klammern hinzugefügt und rot hervorgehoben. Damit ist der Wechsel zum trans()-Algorithmus ermöglicht.

### Wechsel zum trans()-Algorithmus

Wenn die Definition binär geklammert ist, kann man in den trans()-Algorithmus durch Anklicken des TRANS()-ALGORITHMUS ANWENDEN-Buttons wechseln.

### Beibehalten der binären Klammerung

Falls in der EBNF-Anzeige nachträglich eine binäre Klammerung vorgenommen wurde, so wird beim Wechsel zur EBNF-Eingabe (mittels DEFINITION ÜBERARBEITEN-Button) und beim Speichern aus der EBNF-Anzeige heraus nachgefragt, ob diese in der Definition beibehalten werden soll.



Beibehalten der Klammerung

### 5.3.3. Speichern und Laden von EBNF-Defintionen

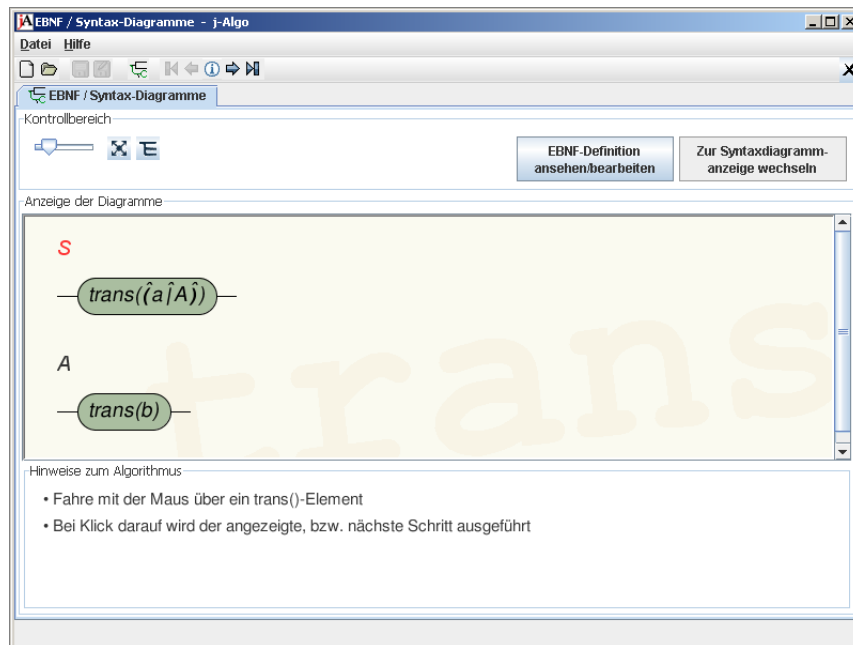
Man kann grundsätzlich Definitionen zu jedem Zeitpunkt in der EBNF-Eingabe speichern.

Dies geschieht entweder über das Menü DATEI oder die entsprechenden Symbole in der Werkzeugleiste. Außerdem wird gefragt, ob die Definition gespeichert werden soll, sobald man das Modul verlassen möchte und nicht gesicherte Veränderungen vorgenommen hat.

Falls in der EBNF-Anzeige nachträglich eine binäre Klammerung vorgenommen wurde, so wird beim Speichern aus der EBNF-Anzeige heraus nachgefragt, ob diese in der Definition beibehalten werden soll.

## 5.4. Der trans()-Algorithmus

### 5.4.1. Die Arbeitsfläche






Die Arbeitsfläche des trans()-Algorithmus

- Anzeigebereich






Hier werden die Syntaxdiagramme visualisiert, vollständig, sowie auch während des Algorithmus. Neben den Syntaxdiagrammelementen existieren hier zusätzlich noch die hinterlegten trans()-Elemente. Fährt man mit der Maus darüber, werden die zu trennenden Elemente farblich hervorgehoben und im Erklärungsbereich wird der dazugehörige Schritt angezeigt. Bei Klick, wird dieser Schritt ausgeführt.

- Kontrollbereich

Im Kontrollbereich hat man die Möglichkeit, die Syntaxdiagramme grafisch zu verändern. Die Möglichkeiten beschränken sich auf das Skalieren der Diagramme, das automatische Anpassen der Größe an das Fenster, sowie die Entfernung von Treppenstrukturen. Auf der rechten Seite befinden sich zudem noch Navigationsbuttons.

-  Hier haben Sie die Möglichkeit, das Diagramm selbst zu vergrößern und zu verkleinern
-  Hier kann man einstellen, ob sich die Größe der Diagramme automatisch dem Fensterrahmen anpassen soll oder nicht.
-  Bei Aktivierung dieses Knopfes, werden Treppenstrukturen in den Diagrammen entfernt.



- Wechsel zur SyntaxDiagramm-Anzeige  
ACHTUNG! Ein Wechsel in die Syntaxdiagramm-Anzeige ist nur nach Beenden des Algorithmus möglich und man kommt zudem nicht wieder zurück! Wechselt man in die Ebnf-Ansicht und lässt die Ebnf-Definition unverändert, fährt man bei Rückkehr in den trans()-Algorithmus fort, ändert man die Ebnf-Definition, wird der Algorithmus neu gestartet.
- Steuerbereich  
Hier kann man neben dem direkten Arbeiten in der Anzeige noch über Buttons den Algorithmus steuern.
  -  Macht alle bisher ausgeführten Schritte rückgängig
  -  Macht den letzten ausgeführten Schritt rückgängig
  -  Bei Aktivierung dieses Buttons wird permanent der nächste Schritt hervorgehoben
  -  Führt den nächsten automatisch gewählten Schritt aus
  -  Führt den Algorithmus bis zum Ende aus
- Erklärungsbereich  
Hier werden die Erklärungen zu den einzelnen Schritten angezeigt oder es werden dem Nutzer Hinweise angezeigt

### 5.4.2. Steuerung des Algorithmus

Es existieren zwei Möglichkeiten, den Algorithmus zu steuern. Zum einen

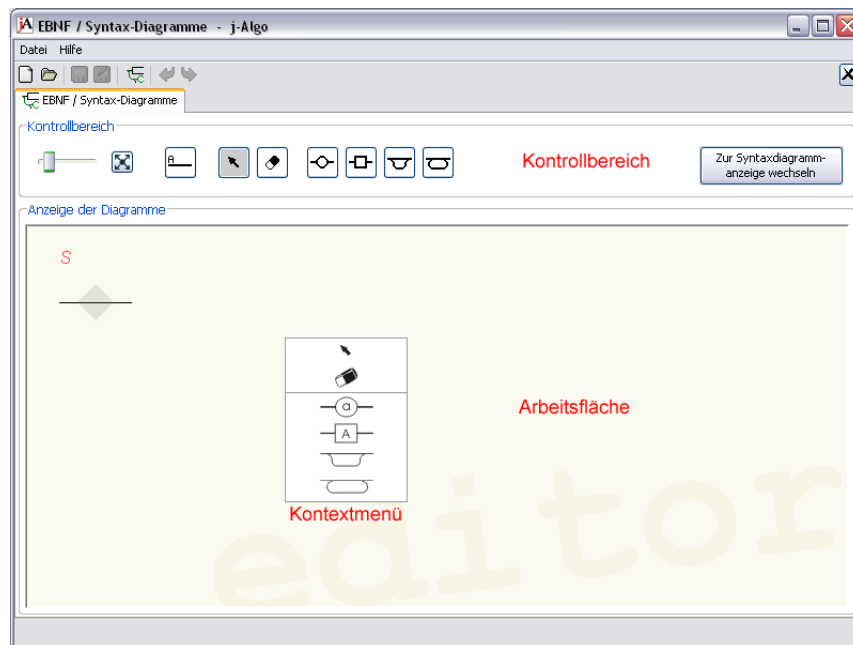
- Im Anzeigebereich  
Hier kann man direkt mit den farblich hinterlegten trans()-Elementen arbeiten. Befindet sich die Maus darüber, werden die zu trennenden Teilstücke hervorgehoben sowie der zugehörige Algorithmusschritt im Erklärungsbereich angezeigt. Klickt man darauf, wird dieser Schritt ausgeführt und das Element umgeformt.
- Über die Steuerbuttons  
Die Erklärung zu den Steuerbuttons befindet sich im Abschnitt [5.4.1](#) in der Erklärung zur Arbeitsfläche.

## 5.5. Syntaxdiagramme

### 5.5.1. Der Syntaxdiagramm-Editor

#### Die Arbeitsfläche



Die Arbeitsfläche des Syntaxdiagramm-Editors ist zweigeteilt: Im oberen Abschnitt befindet sich der Kontrollbereich, über den alle Aktionen, die für das Erstellen und Editieren von Syntaxdiagrammen nötig sind, anwählbar sind. Darunter befindet sich die Anzeige für Syntaxdiagramme - die Arbeitsfläche, auf der an den Diagrammen gearbeitet wird.



#### Hinzufügen von Elementen

Um einem Syntaxdiagrammsystem Elemente hinzufügen zu können, muss man sich in einem entsprechenden Modus befinden. Einzig das Hinzufügen neuer Syntaxdiagramme geschieht ohne Moduswechsel.

- Syntaxdiagramme

Um ein neues Syntaxdiagramm einzufügen genügt ein einfacher Klick auf das Symbol  im Kontrollbereich. In der Syntaxdiagramm-Anzeige erscheint ein Popupfenster, in dem sich u.a. ein Texteingabefeld befindet. In diesem bereits fokussierten Eingabefeld können Sie nun den gewünschten Namen ihres neuen Syntaxdiagramms angeben. Ihre Eingabe beenden Sie entweder mit dem OK-Button  rechts des Eingabefeldes oder über ENTER.

Falls der von Ihnen gewählte Name schon für ein anderes Diagramm vergeben ist, werden Sie im Rahmentext des Popupfensters darauf hingewiesen. Sie müssen einen anderen

Namen wählen.



Unterhalb des Eingabefeldes befindet sich eine weitere Zeile:

Startdiagramm ☐

Indem Sie die Checkbox auswählen, können Sie ihr neues Diagramm als Startdiagramm einfügen. Es ist immer nur ein Startdiagramm zulässig, d.h. das alte Startdiagramm verliert dieses „Privileg“ an das neue Syntaxdiagramm.

Sie können die Eingabe jederzeit mit ESC abbrechen. Die Eingabe eines leeren Diagrammnamens bewirkt ebenfalls einen Abbruch. Es wird kein Syntaxdiagramm hinzugefügt.

- Über die Einfüge-Modi

Um Elemente einfügen zu können, müssen Sie in den Modus „<Element> einfügen“ wechseln. Dieses Beispiel demonstriert dies für Terminalsymbole. Den Moduswechsel erreichen Sie auf zwei verschiedenen Wegen: entweder wählen Sie im Kontrollbereich den Button  aus oder Sie bewegen Ihre Maus in die Syntaxdiagramm-Anzeige. Dort können Sie jederzeit mit einem Rechtsklick ein Kontextmenü öffnen, in dem Sie ebenfalls den Modus „Terminalsymbole einfügen“ anwählen können. 

Ist dieser Modus gewählt, wird der zugehörige Button im Kontrollbereich ausgegraut. Auch ändert sich der Mauszeiger, wenn Sie über den Arbeitsbereich fahren.



Somit wissen Sie immer, in welchem Modus Sie sich gerade befinden.

- Über die Rhomben

Im Editiermodus bekommen die Syntaxdiagramme neue Elemente - die Rhomben. Sie sind Pseudoelemente, die das Einfügen von Elementen erleichtern bzw. erst ermöglichen. In ihrem Grundzustand sind die Rhomben grau. Befindet man sich in einem Einfügemodus, so leuchten sie grün auf, falls an dieser Stelle ein Hinzufügen möglich ist. Leuchten sie dagegen rot auf, kann das Element an dieser Stelle nicht eingefügt werden.


Hat man etwas eingefügt, erscheinen vor und hinter diesem Element neue Rhomben, an denen erneut hinzugefügt werden kann. In Wiederholungen und Verzweigungen erscheinen zudem in den Elementen neue Rhomben - schließlich soll man auch hier Einfügen können.

- Terminalsymbole und Variablen



Wechseln Sie als erstes in den Modus TERMINALSYMBOL EINFÜGEN oder VARIABLEN EINFÜGEN.   In der Arbeitsfläche können Sie nun mit der Maus über jeden beliebigen Rhombus fahren - er leuchtet grün auf. Mit einem Linksklick fügen Sie das Terminalsymbol / die Variable an dem aktuell grün leuchtenden Rhombus ein.

## 5. Das Modul Ebnf und Syntaxdiagramme

Es erscheint ein Textfeld, in dem Sie einen Namen für das Element eingeben müssen. Als Standard erscheint ausgewählt „a“ / „A“.

Zum Bestätigen einer Eingabe drücken Sie entweder auf den OK-Button  oder ENTER. Falls Sie das Erstellen abbrechen möchten, drücken Sie ESC oder geben einen leeren Namen ein.

- Verzweigungen und Wiederholungen

Wechseln Sie als erstes in den Modus „Verzweigungen einfügen“ / „Wiederholungen einfügen“  . In der Arbeitsfläche können Sie nun mit der Maus über jeden beliebigen Rhombus fahren - er leuchtet grün auf. Mit einem Linksklick fügen Sie den Anfang der Verzweigung / Wiederholung ein. Daraufhin erscheint grau gepunktet eine neue Verzweigung / Wiederholung. Diese zeigt, von wo bis wo eingefügt werden kann. Anfangs ist sie auf den angeklickten Rhombus beschränkt.

Sie können diese „unfertige“ Verzweigung / Wiederholung nun sowohl nach links als auch nach rechts in die Breite ziehen. Fahren Sie dazu mit der Maus über andere Rhomben. Leuchtet ein Rhombus grün auf, so kann bis zu dieser Stelle eingefügt werden. Dies wird durch die veränderte grau gepunktete Verzweigung / Wiederholung angezeigt. Dabei können auch schon vorhandene Elemente eingeschlossen werden.

Leuchtet ein Rhombus dagegen rot auf, so darf an dieser Stelle nicht eingefügt werden. Dies ist der Fall, wenn das Ende nicht auf der gleichen Linie liegt wie der Startpunkt. Auch dürfen sich Linien nicht kreuzen, so dass das Einfügen eines Endes in das Innere einer anderen Verzweigung / Wiederholung ebenfalls nicht erlaubt ist.

Haben Sie den Mauszeiger über ein grün leuchtendes Ende gefahren, so können Sie nun mit einem weiteren Linksklick Einfügen. Die grau gepunktete Verzweigung / Wiederholung verschwindet und ein vollständig neues Element erscheint.

### Bearbeiten und Löschen von Elementen

- Editieren

Im Editiermodus ist es möglich, Terminalsymbol-, Variablen- und Syntaxdiagrammnamen zu ändern.

Wechseln Sie dazu zuerst in diesen Modus. 

Fahren Sie nun mit der Maus über Terminalsymbole oder Variablen, so leuchten diese blau auf. Mit einem Linksklick auf diese Elemente erscheint ein Textfeld mit dessen Name. Sie können den Namen nun beliebig ändern. Mit dem OK-Button oder ENTER schließen Sie die Änderung ab. Mit ESCAPE oder der Eingabe eines leeren Namens brechen Sie den Editiervorgang ab.

Fahren Sie mit der Maus über den Namen eines Syntaxdiagramms, so erscheint dieser fett. Mit einem Linksklick können Sie nun den Namen des Diagramms ändern. Dazu erscheint der Dialog, der auch zum Einfügen neuer Elemente angezeigt wird. Einziger Unterschied ist, dass, falls Sie das Startdiagramm editieren, Sie diesem Diagramm diesen Status nicht nehmen können - die entsprechende Checkbox ist ausgegraut.

- Löschen

Im Löschmodus ist es möglich, sowohl alle Syntaxdiagrammelemente als auch leere Diagramme zu löschen.

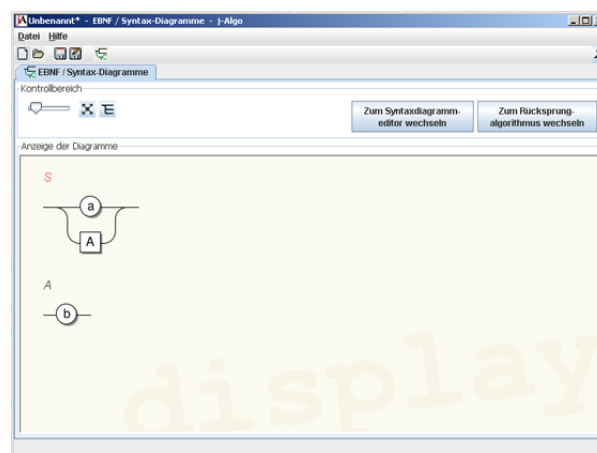
Wechseln Sie dazu zuerst in diesen Modus. 

Fahren Sie nun mit der Maus über Terminalsymbole oder Variablen, so leuchten diese rot auf. Mit einem Linksklick werden diese gelöscht. Nachfolgende Elemente rücken nach links.

Verzweigungen und Wiederholungen können ebenfalls gelöscht werden. Bewegen Sie dazu den Mauszeiger auf die obere oder untere Linie eines dieser Elemente. Die Linie und alle darauf liegenden Elemente leuchten rot auf. Mit einem Linksklick bestätigen Sie den Löschvorgang. War die untere Linie markiert, so werden Elemente der oberen Linie beibehalten. Löschen Sie die obere Linie einer Verzweigung, so werden die Elemente der unteren Linie - sofern vorhanden - auf die obere Linie eingefügt. Die untere Linie verschwindet. Dies geschieht auch beim Löschen des unteren Teils einer Wiederholung; nur werden hier die Elemente in umgekehrter Reihenfolge oben eingefügt (d.h. ihre „Lese-richtung“ bleibt erhalten).

Befinden Sie sich mit dem Mauszeiger über einem leeren Syntaxdiagramm (d.h. nur ein Rhombus befindet sich in diesem Diagramm), so erscheint um dieses ein roter Rahmen. Bestätigen Sie mit einem Linksklick, wird das Diagramm gelöscht. Handelt es sich dabei um das Startdiagramm, so erscheint ein Hinweis, dass das Löschen des Startdiagramms nicht möglich ist.

### 5.5.2. Die Syntaxdiagramm-Anzeige






## 5. Das Modul Ebnf und Syntaxdiagramme

### Anzeigebereich

Hier werden die Syntaxdiagramme visualisiert.

### Kontrollbereich

Im Kontrollbereich hat man die Möglichkeit, die Syntaxdiagramme grafisch zu verändern. Die Möglichkeiten beschränken sich auf das Skalieren der Diagramme, das automatische Anpassen der Größe an das Fenster, sowie das Entfernen von Treppenstrukturen. Auf der rechten Seite befinden sich zudem noch Navigationsbuttons.

-  Hier haben Sie die Möglichkeit, das Diagramm selbst zu vergrößern und zu verkleinern
-  Hier kann man einstellen, ob sich die Größe der Diagramme automatisch dem Fensterrahmen anpassen soll oder nicht.
-  Bei Aktivierung dieses Knopfes, während Treppenstrukturen in den Diagrammen entfernt.

### 5.5.3. Speichern und Laden von Syntaxdiagrammen

#### Speichern

Natürlich haben Sie auch die Möglichkeit, Syntaxdiagramme zu speichern.

Dies ist möglich in der Syntaxdiagramm-Anzeige sowie im Syntaxdiagramm-Editor. Das Speichern ist nur mit der Endung „\*.jalgo“ möglich. Das Programm erkennt jedoch automatisch, dass es sich um ein Syntaxdiagramm handelt.

Hinweis! Speichern sie die Diagramme in einem eindeutigen Ordner oder geben Sie dem Diagramm einen geeigneten Namen, da die Speicherendungen im gesamten **j-Algo** „\*.jalgo“ sind.

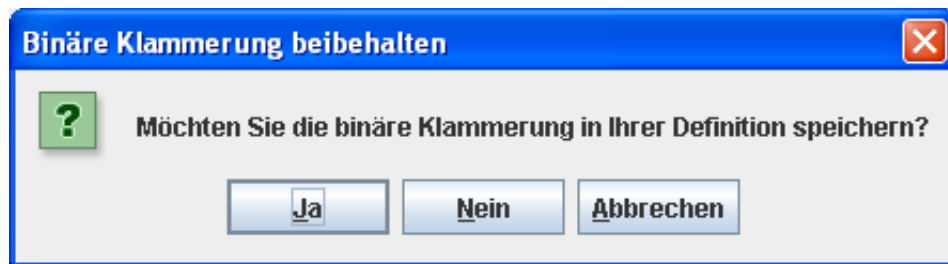
#### Laden von Syntaxdiagrammen

Beim Laden von Syntaxdiagrammen unterscheidet das Modul, ob sie vollständig sind oder nicht. Vollständig bedeutet, dass zu jeder Variable im Diagramm ein Diagramm mit diesem Namen existiert. Ist es vollständig, befinden Sie sich nach dem Laden in der Syntaxdiagrammanzeige. Ist es nicht vollständig wird der Editor geöffnet und das Diagramm kann bearbeitet werden.

## 5.6. Der Rücksprung-Algorithmus

### 5.6.1. Die Arbeitsfläche

Die Arbeitsfläche des Rücksprungalgorithmus ist in vier Bereiche aufgeteilt.



Beibehalten der Klammerung

- **Syntaxdiagramm-Ansicht**  
die Syntaxdiagramme werden hier grafisch dargestellt. Darüber hinaus erfolgt die Steuerung während des Algorithmus' durch Anklicken der Diagramme in dieser Ansicht.
- **Keller**  
Hier werden alle Rücksprungadressen dargestellt, die während des Algorithmus' auf den Keller gelegt werden.
- **Kontrollbereich**  
Hier wird das Wort ausgegeben, das während des Algorithmus' erzeugt wurde. Darüber hinaus kann hier ein Wort eingegeben werden, das erzeugt werden soll, der Algorithmus wird von hier gestartet. Zusätzlich bietet der Kontrollbereich die Möglichkeit, die Darstellung der Syntaxdiagramme in der Syntaxdiagramm-Ansicht anzupassen.
- **Erklärungsbereich**  
Hier werden während des Ablaufes des Algorithmus' Erklärungen zu den einzelnen Algorithmusschritten ausgegeben.

### 5.6.2. Steuerung des Algorithmus

Aufgabe des Bereichs ‚Rücksprungalgorithmus‘ ist es, den Algorithmus der Worterzeugung zu visualisieren. Er repräsentiert den Ablauf eines Weges durch ein Syntaxdiagramm-System und erzeugt während des Ablaufes ein Wort, das sich aus den passierten Terminalsymbolen zusammensetzt. Das Modul ist dabei nicht in der Lage, Entscheidungen über die Wegewahl bezüglich der Erzeugung eines Wortes selbst zu treffen. Vielmehr soll er dem Nutzer helfen, eigene Durchläufe und Sprünge zwischen den einzelnen Diagrammen zu überprüfen.

Im Folgenden wird die Steuerung des Algorithmus' erläutert:

## 5. Das Modul Ebnf und Syntaxdiagramme

### Eingabe eines Wortes

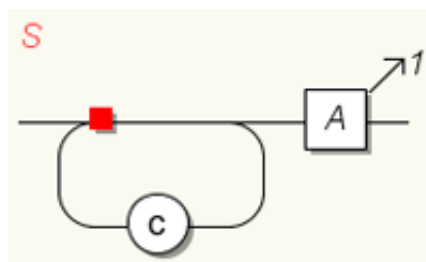
Bevor der Algorithmus gestartet wird, besteht die Möglichkeit, im Kontrollbereich ein Wort einzugeben, welches während des Algorithmus erzeugt werden soll. Dieses kann einfach in das Textfeld eingegeben werden, welches mit ‚Zu erzeugendes Wort‘ bezeichnet ist. Die Eingabe eines Wortes, welches erzeugt werden soll, ist optional. Es ist genauso möglich, kein Wort einzugeben und die Generierung von Wörtern so zu testen, oder den Button ‚Wort erzeugen‘ im Kontrollbereich zu klicken, der automatisch ein Wort - welches auch mit dem Algorithmus erzeugt werden kann - generiert (Dabei kann es sich aber auch um das leere Wort handeln, falls das Syntaxdiagramm-System dieses ermöglicht).

### Start des Algorithmus

Nach der (optionalen) Eingabe eines Wortes kann der Algorithmus gestartet werden. Dies ist über einen Klick auf den Button ‚Algorithmus starten‘ im Kontrollbereich, oder über das Menü Rücksprung-Algorithmus-> Algorithmus starten möglich.

Anmerkung: Besteht das im Schritt 1 eingegebene Wort nicht ausschließlich aus in den Syntaxdiagrammen vorhandenen Terminalsymbolen, so ist die Erzeugung eines solchen Wortes prinzipiell nicht möglich. Das Programm wird ein solches Wort zurückweisen und den Algorithmus nicht starten.

Ist der Algorithmus gestartet, so wird die aktuelle Position in einem der gegebenen Diagramme durch einen roten Punkt gekennzeichnet.



Zu Beginn des Algorithmus wird dieser Punkt am Anfang des Startdiagrammes stehen.

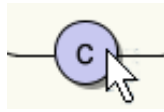
### Wählen eines Weges durch ein Syntaxdiagramm

Um einen Weg durch ein Syntaxdiagramm zu wählen, können einzelne Elemente des Diagrammes in der Syntaxdiagramm-Anzeige direkt angeklickt werden. Folgende Elemente können angeklickt werden: Terminalsymbole, Variablen, leere Verzweigungen und Ausgänge von Syntaxdiagrammen. All diese Elemente müssen von der aktuellen Position aus erreichbar sein.

### Passieren eines Terminalsymbols

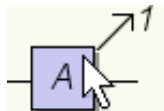
Um ein Terminalsymbol zu passieren, kann dieses einfach angeklickt werden. Dies ist nur möglich, wenn das entsprechende Terminalsymbol von der aktuellen Position im Syntaxdiagramm zu erreichen ist. Ist ein Terminalsymbol zu erreichen, so wird es blau hervorgehoben, sobald man mit der Maus über das Symbol fährt. (BILD) Passierte Terminalsymbole werden an das Ende des erzeugten Wortes im Kontrollbereich angehängt.





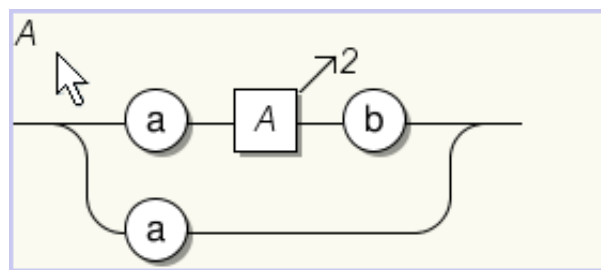
### Passieren einer Variable

Um eine Variable zu passieren, kann diese einfach angeklickt werden. Dies ist nur möglich, wenn die entsprechende Variable von der aktuellen Position im Syntax-Diagramm zu erreichen ist. Ist eine Variable zu erreichen, so wird sie blau hervorgehoben, sobald man mit der Maus über die Variable fährt. (BILD) Wird eine Variable passiert, so wird ihre Rücksprungadresse auf den Keller gelegt. Dem Passieren einer Variable folgt stets ein Sprung in ein Syntaxdiagramm.



### Sprung in ein Syntaxdiagramm

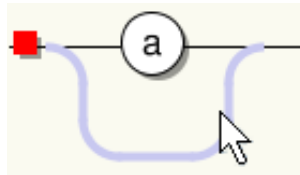
Dem Sprung in ein Syntaxdiagramm geht stets das Passieren einer Variable voraus. Um in ein Syntaxdiagramm zu springen, muss einfach mit der Maus über das entsprechende Syntaxdiagramm in der Syntaxdiagramm-Ansicht gefahren werden, dieses wird dann mit einem blauen Rahmen umrandet. (BILD) Durch den Klick auf das Diagramm wird der Sprung ins Diagramm durchgeführt. Ein Sprung ist nur möglich, wenn der Name des Diagrammes, mit dem Inhalt der zuvor passierten Variable übereinstimmt. Andernfalls wird der Algorithmus den Sprung zurückweisen.



### Passieren einer leeren Verzweigung

Um eine leere Verzweigung zu passieren, kann diese einfach angeklickt werden. Dies ist nur möglich, wenn die entsprechende Verzweigung von der aktuellen Position im Syntaxdiagramm zu erreichen ist. Ist eine leere Verzweigung zu erreichen, so wird sie blau hervorgehoben, sobald man mit der Maus über die Variable fährt. (BILD) Beim Klick auf eine leere Verzweigung wird das erzeugte Wort nicht verändert. Es ändert sich lediglich die Position im Syntaxdiagramm.

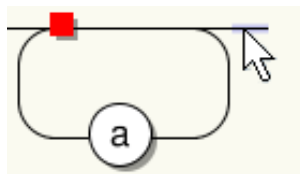
## 5. Das Modul Ebnf und Syntaxdiagramme



### Verlassen eines Syntaxdiagrammes

Ist nach dem Passieren eines Terminalsymbols, einer Variable oder leeren Verzweigung, sowie dem Rücksprung in ein Syntaxdiagramm nur noch der Ausgang des Diagrammes zu erreichen, so wird der Algorithmus das Diagramm automatisch verlassen. Doch nicht immer ist der Weg zum Ausgang die einzige Möglichkeit. In solchen Situationen muss man selbst entscheiden, ob das Diagramm verlassen werden soll.

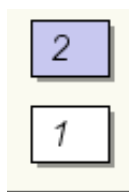
Um das Diagramm zu verlassen, muss einfach in der Syntaxdiagramm-Anzeige auf den Ausgang des Diagrammes geklickt werden. Das Verlassen des Diagrammes wird nur durchgeführt, wenn der Ausgang des Diagrammes von der aktuellen Position im Diagramm erreicht werden kann. Der Ausgang des Diagrammes wird dann beim Fahren mit der Maus über den Ausgang blau hervorgehoben.



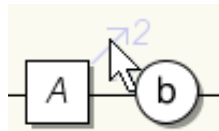
Dem Verlassen des Diagrammes folgt entweder der Rücksprung in ein Diagramm, oder das Ende des Algorithmus.

### Rücksprung in ein Syntaxdiagramm

Wurde ein Syntaxdiagramm verlassen, so kann ein Rücksprung in ein Diagramm folgen. Der Rücksprung folgt immer dann, wenn noch Rücksprungadressen auf dem Keller liegen. (BILD Keller) Um einen Rücksprung durchzuführen, muss einfach in der Syntaxdiagramm-Anzeige die Rücksprungmarke einer Variable in einem Diagramm angeklickt werden. Diese werden, falls ein Rücksprung möglich ist, beim Fahren mit der Maus über die Marke blau hervorgehoben.



Ein Rücksprung wird nur durchgeführt, wenn die Nummer der angeklickten Rücksprungmarke mit der obersten Adresse auf dem Keller übereinstimmt. Die oberste Adresse wird dann vom Keller gelöscht, der Weg kann hinter der Variable fortgesetzt werden, zu der zurückgesprungen wurde. Stimmen Rücksprungmarke und oberste Adresse auf dem Keller nicht überein, so wird der Rücksprung vom Algorithmus zurückgewiesen.




### Ende des Rücksprungalgorithmus

Wird ein Syntaxdiagramm verlassen und es befindet sich keine Rücksprung-Adresse mehr auf dem Keller, so endet der Algorithmus. Er endet jedoch nur dann erfolgreich, wenn das Ende des Startdiagrammes erreicht wurde, und das vor Beginn des Algorithmus eingegebene Wort erzeugt wurde. Andernfalls wird der Algorithmus mit einer Fehlermeldung enden.


### Fehler während des Algorithmusdurchlaufes

Nicht immer werden Fehler während des Laufens durch ein Syntaxdiagramm sofort erkannt und vom Algorithmus zurückgewiesen. So kann es z.B. passieren, dass das eingegebene Wort komplett erzeugt, der Ausgang des Startdiagrammes aber noch nicht erreicht wurde. Genauso kann ein anderes Wort als die Eingabe erzeugt worden sein. In einem solchen Fall wird der Algorithmus die Weiterarbeit verweigern. Es bestehen die Möglichkeiten, einzelne, falsche Schritte rückgängig zu machen, oder den Algorithmus komplett zurück zu setzen.

### Schritte rückgängig machen

Um einen Schritt im Algorithmus rückgängig zu machen, gibt es zwei Möglichkeiten. Zum einen den Klick auf das entsprechende Symbol in der Toolbar am oberen Fensterrand , oder die Auswahl über das Menü Rücksprung-Algorithmus -> Schritt rückgängig.

### Schritte wiederherstellen

Um einen Schritt im Algorithmus wiederherzustellen, gibt es zwei Möglichkeiten. Zum einen den Klick auf das entsprechende Symbol in der Toolbar am oberen Fensterrand , oder die Auswahl über das Menü Rücksprung-Algorithms -> Schritt wiederherstellen.

### Den Rücksprungalgorithmus zurücksetzen

Der Klick auf den Button ALGORITHMUS ZURÜCKSETZEN im Kontrollbereich ermöglicht das Zurücksetzen des Algorithmus in die Ausgangsposition. Einzelne Schritte können von hier aus wieder hergestellt werden. Es kann aber auch ein neues Wort eingegeben und der Algorithmus neu gestartet werden.

## 5. Das Modul Ebnf und Syntaxdiagramme

### **Verlassen des Rücksprunyalgorithmus-Bereiches**

Der Rücksprunyalgorithmus-Bereich kann über den Button ZUR SYNTAXDIAGRAMMANZEIGE WECHSELN im Kontrollbereich verlassen werden. Das Programm kehrt dann in die Syntaxdiagramm-Anzeige zurück. Die gleiche Funktionalität bietet die Auswahl im Menü Rücksprung-Algorithmus -> Algorithmus verlassen.

## 5.7. Impressum

Das Modul **EBNF und Syntaxdiagramme** wurde im Sommersemester 2006 von der Praktikumsgruppe 12 im Rahmen des externen Softwarepraktikums entwickelt. Mitwirkende waren die

### Teammitglieder

- Tom Kazimiers — Administrator
- Johannes Mey — Testverantwortlicher
- Michael Thiele — Chefprogrammierer
- André Viergutz — Assistent
- Claas Wilke — Sekretär

sowie der betreuende Tutor Matthias Schmidt.

Die Webseite des Projektes finden Sie unter <http://web.inf.tu-dresden.de/~swt06-12/>.

## 5. *Das Modul Ebnf und Syntaxdiagramme*

# A. Einleitung zu Datenstrukturen

**Anmerkung:** Der Autor der folgenden Seiten (Kapitel A, B und C dieses Anhangs) ist Jean Christoph Jung (Teammitglied AVL-Modul)

Eine häufige Anwendung auf großen Datenmengen ist das Suchen: Das Wiederfinden eines bestimmten Elements oder bestimmter Informationen aus einer großen Menge früher abgelegter Daten. Um die Suche zu vereinfachen, werden den (möglicherweise sehr komplexen) großen Datensätzen eindeutige Suchschlüssel (Keys) zugeordnet. Der Vergleich zweier solcher Schlüssel ist in der Regel viel schneller als der Vergleich zweier Datensätze. Wegen der Schlüsselzuordnung reicht es aus, alle vorkommenden Algorithmen nur auf den Schlüsseln zu betrachten; in Wirklichkeit verweisen erst die Schlüssel auf die Datensätze.

Eine grundlegende Idee ist nun, die Daten in Form einer Liste oder eines Feldes abzulegen. Diese Datenstruktur ist höchst einfach. Jedoch ist der Aufwand für das Suchen relativ hoch, nämlich  $O(n)$ . Genauer gesagt benötigt man für eine erfolglose Suche  $n$  Vergleiche (bei  $n$  Elementen in der Datenstruktur), denn man muss jedes Element überprüfen, und für eine erfolgreiche Suche durchschnittlich  $(n + 1)/2$  Vergleiche durchführen, da nach jedem Element mit der gleichen Wahrscheinlichkeit gesucht wird.

Eine Verbesserung dieses Verfahrens wäre, die Daten sortiert abzulegen, was zu einer binären Suche führt. Die Suche hat jetzt nur noch Komplexität  $O(\log_2 n)$ , da bei jedem Suchschritt das Feld halbiert wird. Der Nachteil ist, dass durch die Sortierung das Einfügen erschwert wird, da unter Umständen viele Datensätze bewegt werden müssen. Das Verfahren sollte also nur angewandt werden, wenn sehr wenige oder gar keine Einfügeoperationen ausgeführt werden müssen. Dann können die Daten anfangs mit einem schnellen Verfahren sortiert werden und müssen danach nicht mehr verändert werden.

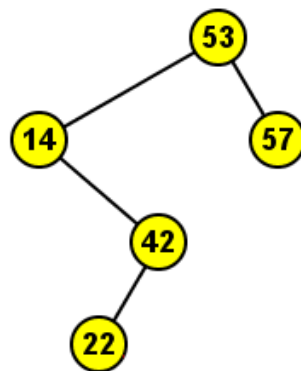
Eine weitere Datenstruktur, die der Suchbäume, wollen wir hier vorstellen.

## *A. Einleitung zu Datenstrukturen*



## B. Suchbäume

Suchbäume sind binäre Bäume (jeder Knoten hat höchstens 2 Kinder) mit der Eigenschaft: Für jeden Knoten gilt: alle Schlüssel im rechten Teilbaum sind größer als der eigene Schlüssel und alle Schlüssel im linken Teilbaum sind kleiner. Diese Eigenschaft wird hier immer Suchbaumeigenschaft genannt.



Ein Beispiel für einen Suchbaum

An dieser Stelle noch eine Bemerkung zu den Schlüsseln: Die Schlüssel können Elemente einer beliebigen Menge sein, unter der Bedingung, dass auf dieser Menge eine Ordnungsrelation definiert ist. Die Ordnungsrelation wird offensichtlich für die Suchbaumeigenschaft benötigt, da dort die Begriffe „kleiner“ und „größer“ vorkommen. Eine häufig verwendete Menge sind die natürlichen Zahlen mit ihrer normalen Ordnungsrelation  $\leq$ . Alle Operationen auf Suchbäumen kann man sich also anhand der natürlichen Zahlen vorstellen.

Aus der Suchbaumeigenschaft kann man sich leicht rekursive Algorithmen für das Einfügen und Suchen in einem Suchbaum herleiten:

**Algorithmus 1** (*Suchen eines Schlüssels  $s$  in einem Suchbaum*)

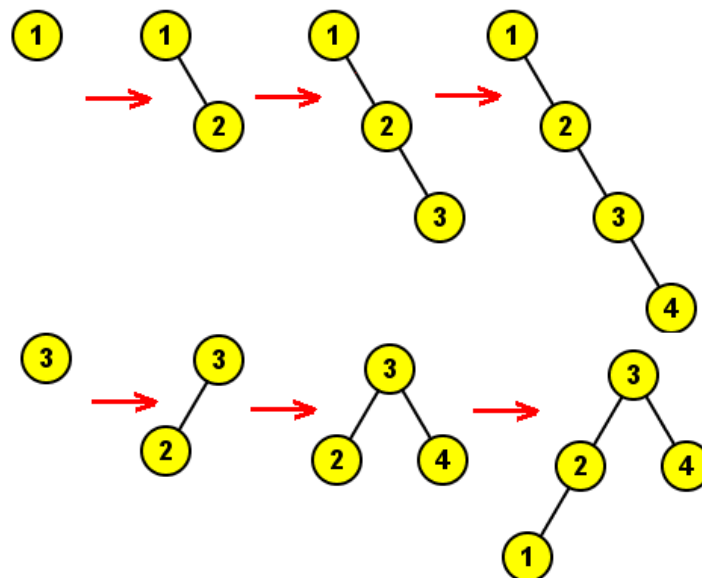
1. Falls Teilbaum leer, dann Schlüssel nicht im Baum vorhanden
2. Falls  $s$  gleich dem Schlüssel des aktuellen Knoten, dann Suche erfolgreich.
3. Falls  $s$  größer als Schlüssel des aktuellen Knotens, dann suche (rekursiv)  $s$  im rechten Teilbaum.
4. Falls  $s$  kleiner als Schlüssel des aktuellen Knotens, dann suche (rekursiv)  $s$  im linken Teilbaum.

## B. Suchbäume

### Algorithmus 2 (Einfügen eines Schlüssels $s$ in einen Suchbaum)

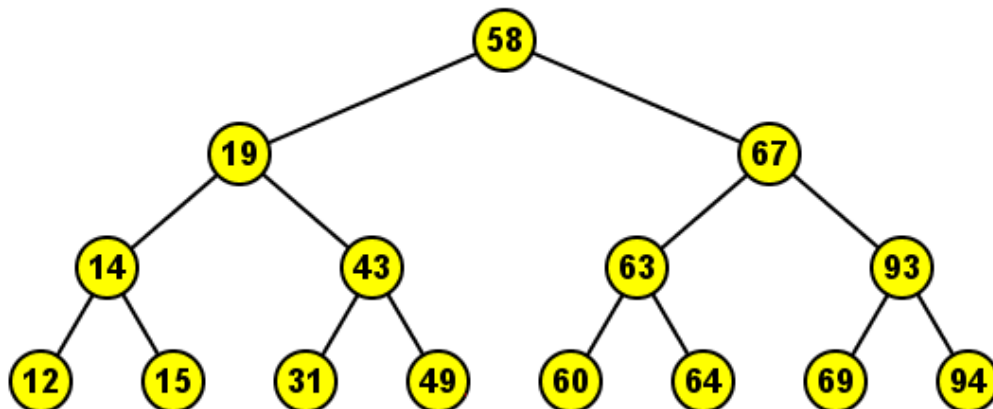
1. Falls Teilbaum leer, dann neuen Schlüssel hier einfügen.
2. Falls  $s$  gleich dem Schlüssel des aktuellen Knotens, dann Schlüssel bereits vorhanden, Einfügen nicht nötig.
3. Falls  $s$  größer als Schlüssel des aktuellen Knotens, dann füge (rekursiv)  $s$  in den rechten Teilbaum ein.
4. Falls  $s$  kleiner als Schlüssel des aktuellen Knotens, dann füge (rekursiv)  $s$  in den linken Teilbaum ein.

Mit Algorithmus 2 ergibt sich folgende Eigenschaft der Struktur von Suchbäumen: im Gegensatz zur sortierten Liste hängt die Struktur eines Baumes davon ab, in welcher Reihenfolge die Elemente eingefügt werden. So erhält man verschiedene Bäume, wenn man 1, 2, 3, 4 in dieser Reihenfolge und in der Reihenfolge 3, 2, 4, 1 einfügt.



Unterschiedliche Suchbäume bei unterschiedlicher Einfügereihenfolge

Im ersten Fall erhält man einen Suchbaum, der zur linearen Liste entartet ist. Das ist nicht nur in dieser speziellen Reihenfolge so, es gibt viele Möglichkeiten einen entarteten Baum zu erzeugen. Der Algorithmus hat also zwei Nachteile: zum einen kann der Suchaufwand linear zur Anzahl der Knoten im Baum sein, was keine Verbesserung zur linearen Liste darstellt; zum anderen hängt die Güte des Verfahrens von der Eingabefolge ab. Der Vorteil von Suchbäumen wird klar, wenn man einen „vollen“ Baum betrachtet, d.h. alle Pfade zu Blättern haben dieselbe Länge. Dann hat sowohl das Suchen (unabhängig davon, ob der Schlüssel im Baum vorhanden ist) als auch das Einfügen eine Komplexität von  $O(\log n)$ .

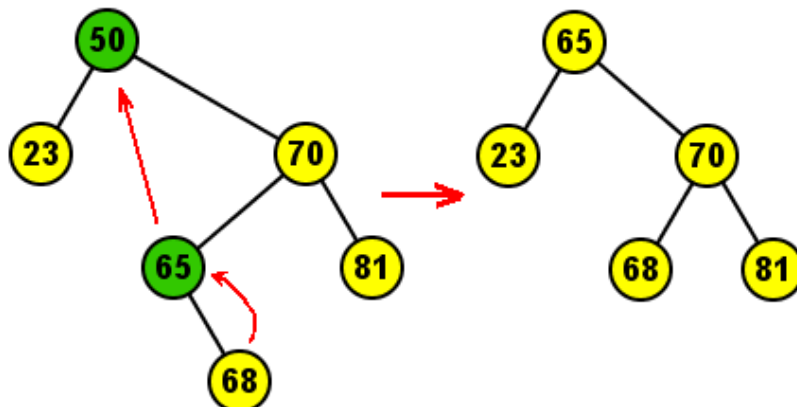


Ein Beispiel für einen vollen Baum

Es gibt einige Algorithmen, bei denen der Einfügealgorithmus so modifiziert ist, dass die entarteten Fälle vermieden werden und immer nahezu volle Bäume entstehen. Einen davon, den Algorithmus nach Adelson-Velskij und Landis (AVL), werden wir später betrachten.

Doch zunächst wollen wir noch eine weitere wichtige Operation auf Suchbäumen untersuchen: das Löschen. Leider ist es nicht ganz so einfach wie Suche und Einfügen.

Zuerst muss der zu löschende Schlüssel gesucht werden. Ist der betreffende Knoten ein Blatt, kann er einfach entfernt werden. Hat der zu löschende Knoten nur ein Kind, kann er durch dieses ersetzt werden. Der schwierige Fall ist, wenn er zwei Kinder hat. Damit die Suchbaumeigenschaft erhalten bleibt, muss man ihn durch den nächst größeren Schlüssel ersetzen. Der nächst größere Schlüssel befindet sich offensichtlich im rechten Teilbaum.



Löschen eines Knoten im Suchbaum

Nach dem Ersetzen bleibt die Suchbaumeigenschaft erhalten, weil alle Schlüssel aus dem linken Teilbaum ohnehin kleiner sind als die aus dem rechten. Außerdem sind auch alle Schlüssel aus dem rechten Teilbaum größer, sonst wäre es nicht der nächst größere Schlüssel gewesen. Aus diesen Überlegungen erhält man folgende verbale Beschreibung des Löschen-Algorithmus:

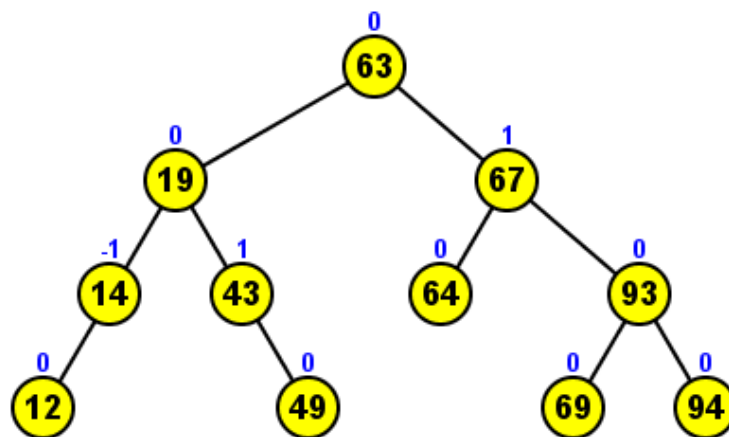
**Algorithmus 3** (*Löschen eines Schlüssels  $s$  aus einem Suchbaum*)

1. Suche  $s$  nach Algorithmus 1. Falls  $s$  nicht im Baum enthalten, terminiert der Algorithmus.
2.
  - a) Ist der zu löschende Knoten ein Blatt, dann entferne ihn einfach aus dem Baum.
  - b) Hat der zu löschende Knoten nur ein Kind, dann ersetze ihn durch dieses.
  - c) Sonst suche den kleinsten Schlüssel im rechten Teilbaum: Gehe zum rechten Kind und dann immer zum linken Teilbaum, solange dieser nicht leer ist. Ersetze den zu löschenden Schlüssel durch den des so gefundenen Knotens. Ersetze den gefundenen Knoten durch sein rechtes Kind.

Man kann anstelle des nächst größeren Schlüssels genausogut den nächstkleineren nehmen, der Algorithmus funktioniert trotzdem. Zur Komplexität ist zu sagen, dass der Algorithmus maximal  $h$  Vergleiche macht, wobei  $h$  die Höhe des Baumes ist. Auch hier ist also die Komplexität abhängig von der Struktur des Baumes.

## C. AVL-Bäume

AVL-Bäume (benannt nach Adelson-Velskij und Landis) sind spezielle Suchbäume: In jedem Knoten unterscheiden sich die Höhen des linken Teilbaums und des rechten Teilbaums um höchstens 1. Um diese Eigenschaft (AVL-Eigenschaft) abzusichern, wird für jeden Knoten ein Balancefaktor eingeführt. Der Balancefaktor ist die Differenz der Höhen des rechten Teilbaums und des linken Teilbaums. Also gilt für jeden AVL-Baum, dass alle Balancefaktoren aus  $\{-1, 0, 1\}$  sind. Bäume mit AVL-Eigenschaft sind niemals als lineare Liste entartet (sofern sie denn mehr als 2 Knoten haben), sondern sind immer fast vollständig. Zwischen der Höhe  $h$  eines Baumes und der Anzahl  $n$  seiner Knoten besteht folgender Zusammenhang:  $h \leq 2 \cdot \log_2 n$ . Das bedeutet, dass für das Suchen logarithmische Komplexität garantiert werden kann (das Suchen erfolgt gemäß Algorithmus 1).



Ein Beispiel für einen AVL-Baum mit Balancen

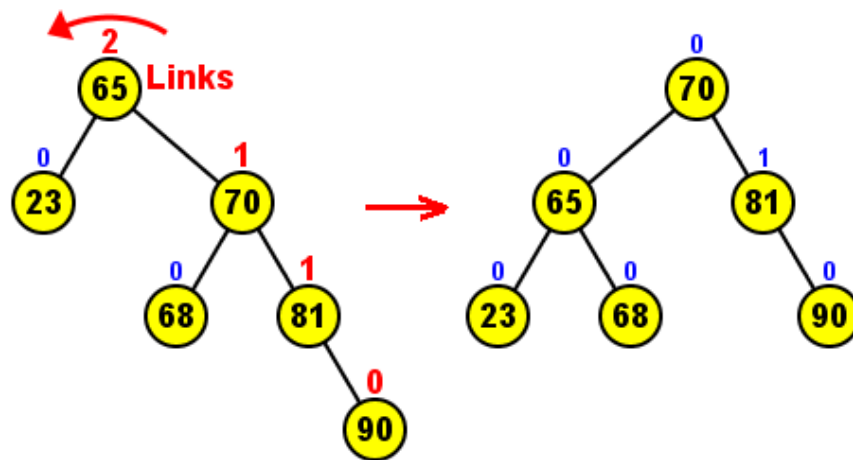
Jetzt muss noch untersucht werden, wie groß der zusätzliche Aufwand beim Einfügen ist, um die AVL-Eigenschaft zu wahren. In jedem Fall wird der neue Knoten als Blatt eingefügt (nach demselben Algorithmus wie bei Suchbäumen). Dabei kann sich der Balancefaktor ändern. Allerdings kann man sich leicht klarmachen, dass das nur entlang des Suchpfades passieren kann. Die Aktualisierung von Balancefaktoren erfolgt von der Einfügestelle zur Wurzel. Hier der Algorithmus:

## C. AVL-Bäume

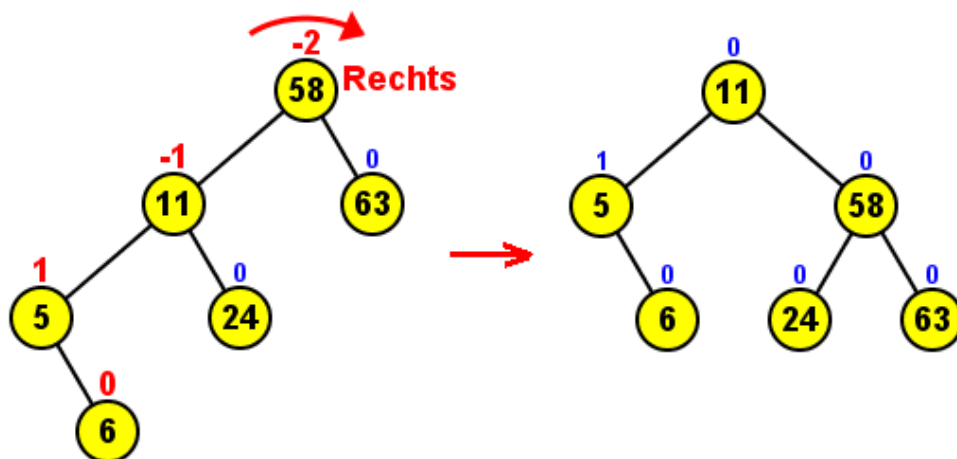
### Algorithmus 4 (Algorithmus zum Einfügen eines Elements $x$ in einen AVL-Baum)

1. Füge das neue Element  $x$  als direkten Nachfolger des Knotens  $n$  als Blatt ein, sodass die Suchbaumeigenschaft erfüllt bleibt. Aktualisiere  $n.balance$ .
2. Setze  $n$  auf den Vorgängerknoten von  $n$ .
  - a) Falls  $x$  im linken Unterbaum von  $n$  eingefügt wurde
    - i. wenn  $n.balance == 1$  dann  $n.balance = 0$  und gehe nach 3.
    - ii. wenn  $n.balance == 0$ , dann  $n.balance = -1$  und gehe nach 2.
    - iii. wenn  $n.balance == -1$  und
      - wenn  $n.left.balance == -1$ , dann Rechts( $n$ )-Rotation.
      - wenn  $n.left.balance == 1$  dann Links( $n.left$ )-Rechts( $n$ )-Rotation.
 Gehe zu 3.
  - b) Falls  $x$  im rechten Unterbaum von  $n$  eingefügt wurde
    - i. wenn  $n.balance == -1$  dann  $n.balance = 0$  und gehe nach 3.
    - ii. wenn  $n.balance == 0$ , dann  $n.balance = 1$  und gehe nach 2.
    - iii. wenn  $n.balance == 1$  und
      - wenn  $n.left.balance == 1$ , dann Links( $n$ )-Rotation.
      - wenn  $n.left.balance == -1$  dann Rechts( $n.left$ )-Links( $n$ )-Rotation.
 Gehe zu 3.
3. Gehe zurück zur Wurzel.

Zur Analyse dieses Algorithmus: Das reine Einfügen erfolgt gemäß Einfügen im Suchbaum (Algorithmus 2), allerdings ist hier sichergestellt, dass sich die Höhe logarithmisch zur Anzahl der Knoten verhält, d.h. auch der Aufwand für das Einfügen ist garantiert logarithmisch. Wie gesagt können sich jedoch Balancefaktoren geändert haben, sodass die AVL-Eigenschaft nicht mehr erfüllt ist. Das wird durch sogenannte Rotationen behoben. Es gibt zwei Typen von Rotationen – Linksrotation und Rechtsrotation, jeweils um einen Knoten  $n$ .



Linksrotation um den Knoten 65



Rechtsrotation um den Knoten 58

Falls durch das Einfügen irgendwo ein Balancefaktor 2 ( $-2$ ) entsteht (größere Änderungen können beim Einfügen eines Knotens offensichtlich nicht auftreten), heißt das, dass sich im rechten (linken) Teilbaum die Höhe um 1 erhöht hat. Durch Rotation(en) wie im Algorithmus angegeben, wird aber genau diese Höhe wieder reduziert. Somit ist klar, dass man maximal zweimal rotieren muss, der Aufwand ist also noch erträglich, im Gegensatz zum Löschen, wie man gleich sehen wird.

Genauso wie Einfügen verändert auch Löschen eines Knotens aus einem AVL-Baum die Balancefaktoren, also müssen auch hier Rotationen ausgeführt werden. Hier der Algorithmus:

**Algorithmus 5** *Löschen eines Knotens aus einem AVL-Baum)*

1. Lösche den Knoten analog zum Löschen im Suchbaum (Algorithmus 3). Falls der Knoten ein Blatt war oder nur einen linken Nachbarn hatte, setze aktuellen Knoten auf den Vater. Sonst setze aktuellen Knoten auf den Vater des Knotens mit dem nächst größeren Schlüssel.
2. Berechne den Balancefaktor des aktuellen Knotens neu. Falls
  - a) Balance 2 und rechte Balance -1, dann Rechts( $n.right$ )-Links( $n$ )-Rotation.
  - b) Balance 2 und rechte Balance nicht -1, dann Links( $n$ )-Rotation.
  - c) Balance -2 und linke Balance 1, dann Links( $n.left$ )-Rechts( $n$ )-Rotation.
  - d) Balance -2 und linke Balance nicht 1, dann Rechts( $n$ )-Rotation.
  - e) sonst keine Rotation.

Wiederhole diesen Schritt solange, bis die Wurzel erreicht ist.

Auch hier wollen wir den Aufwand des Algorithmus etwas genauer untersuchen. Schritt 1 entspricht dem Löschen aus dem Suchbaum, nur garantiert mit logarithmischem Aufwand. Die Frage ist nun, ob, wie beim Einfügen, der Algorithmus mit maximal zwei Rotationen auskommt. Leider ist das nicht der Fall. Das liegt an der erwähnten Eigenschaft der Rotationen, sie verringern die Höhe eines Teilbaums. Beim Einfügen war das gut, da durch das Anhängen

### *C. AVL-Bäume*

eines Knotens gerade die Höhe vergrößert wurde. Hier jedoch ist das unvorteilhaft, es kann passieren, dass man mehrere Rotationen auf dem Weg zur Wurzel durchführen muss. Die Anzahl der Rotationen ist nur durch die Höhe des Baums beschränkt. Das ist in Anwendungsfällen nicht wünschenswert.

Bei zeitkritischen Anwendungen muss man also entweder auf einen anderen Algorithmus ausweichen, oder Varianten wie etwa Lazy-Delete implementieren, d.h. der Knoten wird nur als gelöscht markiert und wird später (wenn Zeit ist) aus dem Baum entfernt.



# Literaturverzeichnis

- [1] R. Sedgewick, „Algorithmen in C++“, Addison-Wesley, 5. Auflage, 1999
- [2] Prof. Vogler, „Vorlesungsskript Algorithmen, Datenstrukturen und Programmierung“, TU Dresden, 2003
- [3] <http://de.wikipedia.org/wiki/AVL-Baum>
- [4] <http://dbs.uni-leipzig.de/de/skripte/ADS1/HTML/kap6-11.html>