Lexikalische Analyse

Eingabe: ***Programm als Zeichenfolge***

Aufgaben:	**Übersetzermodul:**
	Eingabeverarbeitung
Erkennen und klassifizieren der Symbole	Scanner (zentrale Phase, endlicher Automat)
Ausblenden bedeutungsloser Zeichen	
Kodieren der Symbole	Bezeichnermodul
Speichern von Symbolinformation	Literalmodule
	Zeichenspeicher
Konversion	

Ausgabe: ***Programm als kodierte Symbolfolge***

Symbolrepräsentation:

Syntax-Code	**Attribut**	**Quellposition**
Terminal der konkreten Syntax	Wert oder Referenz auf Datenmodul	für Fehlermeldungen in späteren Übersetzerphasen

Vorlesung Übersetzer WS 97/98 / Folie 12

Ziele:

- Aufgaben des Scanners und der Datenmodule,
- Symbolrepräsentation

in der Vorlesung:

- Informationsgehalt von Grundsymbolen an Beispielen,

nachlesen:

Kastens / Übersetzerbau, Abschnitt 3, 3.3.1

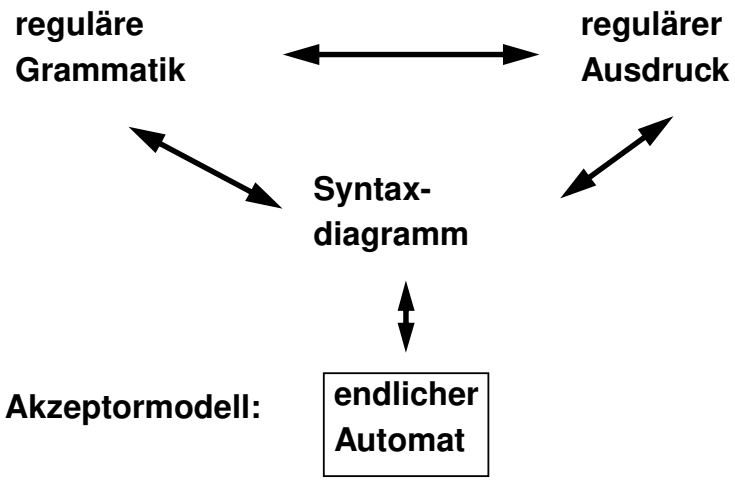
Übungsaufgaben:

- unkonventionelle Wortsymbole
- verschiedene Formen von Kommentaren
- Trennung von Symbolen in FORTRAN

Verständnisfragen:

- Was weiß die lexikalische Analyse nicht über Symbole?
- Bestimmte Entscheidungen beim Sprachentwurf können eine systematische lexikalische Analyse erschweren. Geben Sie Beispiele.
- Erläutern Sie das typedef-Problem in C.

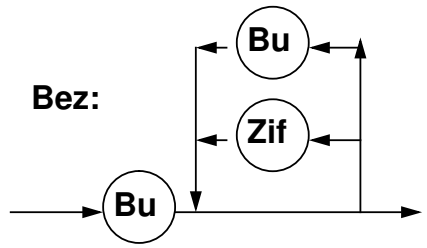
Spezifikation von Grundsymbolen

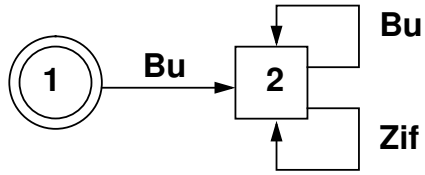


Beispiel Bezeichner:

Bez ::= Bu X
X ::= Bu X
X ::= Zif X
X ::=

Bez = Bu (Bu | Zif)*





Vorlesung Übersetzer WS 97/98 / Folie 13

Ziele:

- Äquivalente Spezifikationsformen

in der Vorlesung:

- unser Entwurfsweg: regulärer Ausdruck in Syntaxdiagramm, dieses in endlichen Automaten transformieren

nachlesen:

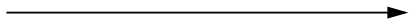
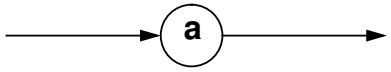
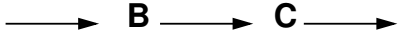
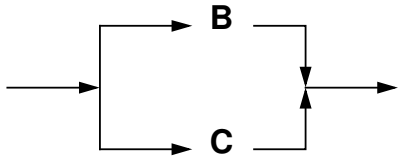
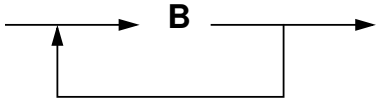
Kastens / Übersetzerbau, Abschnitt 3.1, Kalküle aus Einführung in die theoretische Informatik wiederholen

Verständnisfragen:

- geben Sie Beispiele für Unix-Werkzeuge, die reguläre Ausdrücke zur Beschreibung ihrer Eingabe verwenden.

Reguläre Ausdrücke in Syntaxdiagramme

Transformationsregeln:

regulärer Ausdruck A	Syntaxdiagramm für A
leer	
a	
B C	
B | C	
B*	
B+	

Vorlesung Übersetzer WS 97/98 / Folie 14

Ziele:

- Konstruktion durch rekursives Ersetzen

in der Vorlesung:

- Verfahren am Beispiel Gleitpunktzahlen aus Pascal

nachlesen:

Kastens / Übersetzerbau, Abschnitt 3.1

Übungsaufgaben:

- Verfahren anwenden Aufgabe 4

Verständnisfragen:

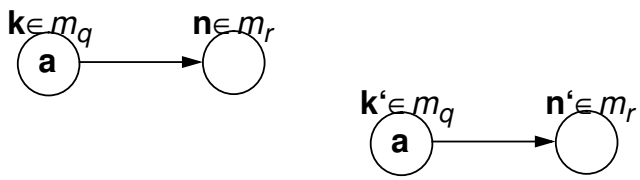
- Bei der Transformation in umgekehrter Richtung erfordern bestimmte Diagrammstrukturen die Verdopplung von Teilausdrücken. Geben Sie Beispiele.
- Stellen Sie für dieses Problem eine Analogie zu Programmabläufen mit Marken, Sprüngen und Schleifen her.

Automat aus Syntaxdiagramm konstruieren

Syntaxdiagramm	**Automat**
Knoten, Kanten	Übergänge, Zustände
Knotenmenge m_q	**Zustand** q
unter gleichem Zeichen a verbundene Knotenmengen m_q und m_r	Übergänge q ---> r mit Zeichen a

Konstruktion:

1. **Knoten numerieren**
   Diagrammausgang = *0*
2. **Anfangsmenge** m_1 — **Anfangszustand** *1*
   enthält alle vom Anfang erreichbaren Knoten
3. **konstruiere neue Knotenmengen** und **Übergänge**
   für ein Zeichen a und eine Menge m_q mit Knoten k
   bilde Knotenmenge m_r mit allen Knoten n, wobei gilt



4. **Schritt 3 wiederholen** bis keine neuen Knotenmengen
5. Alle Zustände q sind **Endzustand** gdw. *0* in m_q ist.

Vorlesung Übersetzer WS 97/98 / Folie 15

Ziele:

- Konstruktionsmethode verstehen

in der Vorlesung:

- Methode am Beispiel Gleitpunktzahlen erläutern

nachlesen:

Kastens / Übersetzerbau, Abschnitt 3.2

Übungsaufgaben:

- Konstruktionsbeispiele Aufgabe 4

Verständnisfragen:

- Warum liefert das Verfahren deterministische Automaten?
- Skizzieren Sie ein einfacheres Verfahren, das nicht notwendig deterministische Automaten liefert.

Teilautomaten zusammensetzen

- **Anfangszustände der Teilautomaten identifizieren**

  ggf. auch Übergänge unter gleichem Zeichen identifizieren
- **unterschiedliche Endzustände beibehalten**

  klassifizieren die Symbole
- **Teilautomaten für Kommentare und bedeutungslose Zeichen zufügen**

Übergänge aus Endzuständen:

wann hält der Automat? ··· → □ → ···

Regel des längsten Musters:

- **Automat hält in beliebigem Zustand, falls kein Übergang mit nächstem Zeichen möglich**
- **setzt zurück auf zuletzt durchlaufenen Endzustand**
- **Fehler, falls kein Endzustand durchlaufen**

Konsequenzen für die Trennung aufeinanderfolgender Symbole!

An der Sprachdefinition und der Syntax prüfen!

Vorlesung Übersetzer WS 97/98 / Folie 16

Ziele:

- Vollständiger Automat als Grundlage für den Scanner-Algorithmus
- Regel des längsten Musters verstehen

in der Vorlesung:

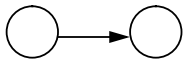
- Beispiel für einen Automaten für eine Programmiersprache
- Beispiele für Anwendung der Regel des längsten Musters

nachlesen:

Kastens / Übersetzerbau, Abschnitt 3.2

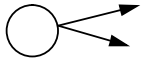
Scanner: Implementierungsaspekte

- **Laufzeit proportional zur Zahl der Zeichen des Programms**
- **Operationen pro Zeichen schnell - sonst dominiert der Scanner die Übersetzungszeit**
- **Tabellen-gesteuerte Automaten zu langsam**
  Schleife, 2-fache Array-Indizierung, Verzweigung
- **direkt programmierter Automat schneller**
  Übergänge in Programmstrukturen umsetzen:



Sequenz

Schleife



Verzweigung o. Sprung

- **schnelle Schleife für bedeutungslose Zwischenräume**
- **Klassen gleichartiger Zeichen: Bitmuster** oder **Indizierung -** statt teurer Mengenoperationen
- **Zeichen nicht aus Eingabepuffer kopieren, sondern**
  Zeiger in Eingabepuffer

Vorlesung Übersetzer WS 97/98 / Folie 17

Ziele:

- Laufzeiteffizienz steht im Vordergrund
- direkt programmierter Automat

in der Vorlesung:

- Statistik zu Vorkommen von Symbolen

nachlesen:

Kastens / Übersetzerbau, Abschnitt 3.3

Übungsaufgaben:

- Direkt programmierten Automaten generieren Aufgabe 5

Verständnisfragen:

- Gibt es Vorteile für Tabellen-gesteuerte Automaten? Argumente kritisch hinterfragen!

Bezeichner- und Literalmodule

- **einheitliche Schnittstelle:**

  Eingabeparameter: Zeiger auf Symboltext

  Ausgabeparameter: Attribut, Syntax-Code

- **Bezeichnermodul kodiert Bezeichner bijektiv, erkennt Wortsymbole**

  Implementierung: Hash-Vektor, erweiterbare Tabelle,

  Kollisionslisten

- **Literalmodule**

  (Gleitpunktzahlen, ganze Zahlen, Zeichenreihen)

  Repräsentation zur Speicherung:

  Quellsymbol,

  Wert konvertiert in Übersetzerdatum

  Wert konvertiert in Zieldarstellung

  Vorsicht:

  Überlauf bei Konversion abfangen!

  Übersetzerdatum evtl. verschieden von Zieldarstellung!

- **Zeichenspeicher**

  speichert Zeichenfolgen ohne Längenbegrenzung

  wird von den übrigen Modulen benutzt

Vorlesung Übersetzer WS 97/98 / Folie 18

Ziele:

- Sichere und effiziente Standardverfahren sind verfügbar.

in der Vorlesung:

- Begründungen für Implementierungsverfahren

nachlesen:

Kastens / Übersetzerbau, Abschnitt 3.3

Verständnisfragen:

- Warum müssen ganze Zahlen meist als Übersetzerdatum verfügbar sein?
- Geben Sie Beispiele für die Konversion von Zeichenreihen.

Scanner-Generatoren

generieren die zentrale Funktion der lexikalischen Analyse

GLA	University of Colorado, Boulder; Komponente von Eli
Lex	Unix Standardwerkzeug
Flex	Nachfolgesystem zu Lex
Rex	GMD Karlsruhe

Symbolspezifikation: reguläre Ausdrücke

GLA	+ programmierte Teilautomaten
	+ Muster für vordefinierte Symbole
Lex, Flex, Rex	+ programmierbare Zustandsübergänge

Schnittstellen:

GLA	wie in diesem Kapitel beschrieben, kooperiert mit anderen Eli-Komponenten
Lex, Flex, Rex	Aktionen zu reg. Ausdrücken für Symbole (beliebige Anweisungen der Impl.sprache), Schnittstelle zu Parser-Generator Yacc

Implementierung:

GLA	direkt programmierter Automat in C
Lex, Flex, Rex	Tabellen-gesteuerter Automat in C
Rex	auch in Modula-2
Flex, Rex	schneller, kleiner als bei Lex

Vorlesung Übersetzer WS 97/98 / Folie 19

Ziele:

- Einige gebräuchlich Scanner-Generatoren kennenlernen

in der Vorlesung:

- Hinweise auf besondere Eigenschaften

nachlesen:

Kastens / Übersetzerbau, Abschnitt 3.4

Übungsaufgaben:

- GLA und Lex anwenden Aufgabe 5