

## 2. Beispiel: $L = \{w \mid |w|_0 = |w|_1\}$

Besprochene Grammatik:

$S \rightarrow \varepsilon, S \rightarrow 0S1S, S \rightarrow 1S0S$

- bison liefert: shift/reduce conflicts
- bedeutet: es gibt Situationen, wo sich nicht klar ist, ob ein Eingabezeichen zu lesen ist oder eine Regel anzuwenden ist.
- **Ursache:** Grammatik nicht eindeutig.

**Satz T8.4.10:** LR( $k$ )-Grammatiken sind eindeutig.

714

## Eindeutige Grammatik für $L$

$S \rightarrow \varepsilon$

$S \rightarrow 0T1S, S \rightarrow 1R0S$

$T \rightarrow 0T1T, T \rightarrow \varepsilon$

$R \rightarrow 1R0R, R \rightarrow \varepsilon$

erinnert an  $R \rightarrow (R)R, R \rightarrow \varepsilon$ ,  
also eine eindeutige  
Grammatik für korrekte  
Klammerausdrücke,  
wobei  $1 \hat{=} (, 0 \hat{=} )$ , s.Übungen

715

## Korrektheit

1. Grammatik erzeugt nur Wörter aus  $L$   
→ offensichtlich
  2. Grammatik erzeugt alle Wörter aus  $L$ .
  3. Grammatik ist eindeutig.
- } →

Induktion über Wortlänge

$S \rightarrow \varepsilon, S \rightarrow 0T1S, S \rightarrow 1R0S$   
 $T \rightarrow 0T1T, T \rightarrow \varepsilon, R \rightarrow 1R0R, R \rightarrow \varepsilon$

716

## Induktion über $|w|$

$|w|=0$ :

- Alle Wörter  $w \in L$  mit  $|w|=0$  herleitbar.
- Alle solchen Wörter haben nur eine Herleitung.

$S \rightarrow \varepsilon, S \rightarrow 0T1S, S \rightarrow 1R0S$   
 $T \rightarrow 0T1T, T \rightarrow \varepsilon, R \rightarrow 1R0R, R \rightarrow \varepsilon$

717

$|w| > 0$ , o.B.d.A. beginne  $w$  mit 0.

Sei  $i > 0$  kleinste Zahl m.  $|w_1 \dots w_i|_0 = |w_1 \dots w_i|_1$ .

Dann gilt:

$$w_1=0, w_i=1, |w_2 \dots w_{i-1}|_0 = |w_2 \dots w_{i-1}|_1$$

$$\text{und } |w_{i+1} \dots w_n|_0 = |w_{i+1} \dots w_n|_1$$

mit

- Jedes Anfangsstück von  $w_2 \dots w_{i-1}$  enthält nicht weniger Nullen als Einsen  
 $\Rightarrow T \xrightarrow{*} w_2 \dots w_{i-1}$  (eindeutig, vgl. Übungen)
- $w_{i+1} \dots w_n \in L \Rightarrow S \xrightarrow{*} w_{i+1} \dots w_n$  (nach IV eind.)

**Also:**  $S \rightarrow 0T1S \xrightarrow{*} 0w_2 \dots w_{i-1}1w_{i+1} \dots w_n$ .

Regel nicht anders anwendbar.

718

### 3. Bsp: Auswerten v.arithm. Ausdr.

Neu: Verwendung von sog. Token.

- Statt Zahlen explizit herzuleiten, enthält die Grammatik Terminale 'NUM'.
- Der Parser erzeugt für die Terminale einen „semantischen“ Wert, also den Wert der Zahl.
- Einlesen der Zahlen in der Prozedur yylex.

719

## Grammatik

**Variablen:**

- s: Startsymbol
- a: additiver Term
- m: multiplikativer Term
- f: Faktor

**Regeln:**

$s \rightarrow a$

$a \rightarrow a+m \mid m$

$m \rightarrow m*f \mid f$

$f \rightarrow (a), \text{ NUM}$

720

## Grammatik in bison-Syntax

```
%token NUM
```

```
s: a      {printf("Ergebnis %d\n", $1);}
```

```
;
```

```
a: a '+' m  {$$=$1+$3;}
```

```
  | m      {$$=$1;}
```

```
;
```

```
m: m '**' f {$$=$1*$3;}
```

```
  | f      {$$=$1;}
```

```
;
```

```
f: '(' a ')' {$$=$2;}
```

```
  | NUM    {$$=$1;}
```

```
;
```

721

## Einlesen der Zahlen

```
int yylex(void)
```

```
{ int c;  
c=getchar();  
if (isdigit(c)  
{ ungetc(c,stdin);  
scanf("%ld",&yylval);  
return NUM; }  
if (c=='\n') return 0;  
return c;  
}
```

C-Prozedur zum  
Einlesen von Zahlen

Übergabe des Wertes  
an den Parser

Rückgabe des Tokens NUM

722

## Chomsky-1-Sprachen

**Satz T5.4.5:**  $L_1 = \text{NTAPE}(n)$ .

- Dabei ist  $\text{NTAPE}(n)$  ist die Menge der von NTMs mit linearem Platz berechenbaren Sprachen.
- Beweis des Satzes ähnlich zu  $L_0 = \{\text{Menge der rek. aufz. Sprachen}\}$  (weggelassen).

**Folgerung:** Alle Chomsky-1-Sprachen sind rekursiv und sind deterministisch in exponentieller Zeit berechenbar.

723

## Abschlusseigenschaften

Die Chomsky-1-Sprachen sind gegen

- Vereinigung (Bew. analog zu kontextfr. Spr.)
- Komplement (Satz v. Immerman/Szelépcsenyi)
- Durchschnitt (Anwendung der de-Morgan-Regeln)

abgeschlossen.

724

## Chomsky-1-Sprachen

$L_1$  enthält NP-schwierige Probleme, z.B.,  $\text{Clique}_{\text{dec}} \in L_1 (= \text{NTAPE}(n))$ :

Durchprobieren aller  $k$ -elementigen Mengen von Knoten und der Test, ob sie eine Clique bilden, geht auf linearem Platz (wenn auch in exponentieller Zeit).

**Fazit:** Chomsky-1-Sprachen zur Beschreibung von Programmiersprachen ungeeignet.

725

## Zusammenfassung

- **Teil 1:** Randomisierte und nichtdeterministische Komplexitätsklassen, NP-Vollständigkeitstheorie

### Hauptziele:

- Vergleich verschiedener Varianten von Randomisierung
- Anhaltspunkte erhalten, dass betrachtete Probleme keine effizienten Algorithmen haben.

726

## Zusammenfassung

- **Teile 2-4:** Chomsky-Hierarchie (vgl. Folie [627](#))

- Chomsky 0: rek. aufz. Sprachen
- Chomsky 1: kontextsensitive Sprachen
- Chomsky 2: kontextfreie Sprachen
- Chomsky 3: reguläre Sprachen

### Hauptziele:

- Beweise, dass Probleme nicht berechenbar sind.
- Konzepte zur Beschreibung von Sprachen/Programmiersprachen

727

## Was ist wichtig für die Prüfung?

**Prüfungsrelevant** ist der Inhalt der Vorlesung (Ausnahmen: LR( $k$ )-Sprachen, bison).

### Wichtig für **Prüfungsvorbereitung:**

- Realistische Zeitplanung
- Routine im Umgang mit dem Stoff
- Sprechen über den Stoff üben
- Sinnvolle Reihenfolge beim Lernen (erst die einfachen Dinge, dann die schwierigeren)
- Ähnliche Vorgehensweisen erkennen
- Richtige Literaturlauswahl

728

## Wie geht es weiter?

- **Effiziente Algorithmen u. Komplexitätstheorie (Schwerpunkt Komplexitätstheorie)**
- **Theorie des Logikentwurfs**
- Binary Decision Diagrams
- Effiziente Algorithmen und Komplexitätstheorie (Schwerpunkt effiziente Algorithmen)
- Randomisierte Algorithmen
- Approximationsalgorithmen
- Quantenrechner
- Kommunikationskomplexität
- Effiziente Algorithmen aus der Bioinformatik
- Online-Algorithmen

729