Einführung in die Informatik

Klaus Knopper

26.10.2004

Syntax von Repräsentationen

Repräsentationen besitzen zwei wichtige Komponenten,

- die Semantik und
- die Syntax.

Bislang haben wir die Syntax einer Repräsentation nicht näher betrachtet. Dies soll nun in diesem Abschnitt geschehen. Wir beschäftigen uns auf den nächsten Folien mit der Frage, wann eine Repräsentation syntaktisch korrekt ist und wann nicht. Dabei beschränken wir uns auf eine Teilmenge aller möglichen Repräsentationen, indem wir nur Repräsentationen betrachten, die mit Hilfe von Zeichen dargestellt werden können.

Beispiele: Stellt die Zeichenfolge `35. Dezember 1888` ein syntaktisch korrektes Datum dar? Ist der Satz `Invormaticker schreiben dank Rechtschreibprüfunk fehlerfrei` syntaktisch korrekt? Wie ist die Abgrenzung zu semantischer Korrektheit?

Definitionen

Alphabet und Worte Ein **Alphabet** T ist eine endliche, nichtleere Teilmenge von Symbolen. Jede Folge $t_1 \dots t_n$ von Symbolen mit der Eigenschaft $t_i \in T$ wird **Wort** genannt. Anders ausgedrückt ist ein Alphabet eine Menge von Zeichen (Symbolen), die zur Darstellung zur Verfügung stehen. Das Alphabet der lateinischen Buchstaben unterscheidet sich vom Alphabet der chinesischen Symbole. Worte sind Repräsentationen, die nur Symbole eines bestimmten Alphabets zur Darstellung benutzen.

Sternhülle eines Alphabets Unter der Sternhülle T^* eines Alphabets T versteht man die Menge aller Worte, die mit Hilfe des Alphabets gebildet werden können.

Wortlänge

Aufbauend auf den Definitionen lässt sich nun eine Funktion

$$l : T^* \rightarrow I\!N$$

zur Bestimmung der Wortlänge definieren. Sei $w = t_1 \dots t_n$ ein beliebiges Wort aus der Sternhülle des Alphabets T, so ist

$$l(w) = n \text{ oder kurz } |w| = n$$

Das leere Wort Das leere Wort ε ist als Wort mit der Länge 0 definiert. Es gilt somit $|\varepsilon| = 0$

Mengen von Repräsentationen

In der Praxis ist es wichtig, Repräsentationen zu Mengen zusammenzufügen. Wir bezeichnen eine Repräsentation r als **syntaktisch korrekt** bezüglich der Menge von Repräsentationen R, wenn $r \in R$. Ansonsten bezeichnen wir r als **syntaktisch fehlerhaft** bezüglich R.

Nun stellt sich die Frage wie festgestellt werden kann, ob eine Repräsentation zu einer Menge gehört oder nicht. Warum wird uns der naive Ansatz, der Speicherung aller Elemente einer (möglicherweise unendlichen) Menge und vergleichen auf Übereinstimmung in der Praxis nicht praktikabel?

Anderer Ansatz: Beschreiben der Syntax mit Hilfe von Regeln.

Beschreibung von Syntax

Die Syntax kann mit Hilfe von Regeln beschrieben werden. Diese Regeln können sowohl

- umgangssprachlich oder
- mit Hilfe einer formalen Methode

beschrieben werden. Die Schwierigkeit besteht dabei nicht in der Beschreibung des Syntax einer einzelnen Repräsentation, sondern in der Beschreibung der Syntax von zusammengehörenden Gruppen von Repräsentationen. Wie ist beispielsweise die Syntax aller natürlichen Zahlen definiert?

Umgangssprachliche Beschreibung

Wir wollen eine Syntax für die Repräsentation von Daten (hier: =Mehrzahl von Datum) festlegen. Der Einfachheit halber verzichten wir auf eine Berücksichtigung von Schaltjahren. Dies hat zur Folge, dass der Februar auch in Nicht-Schaltjahren 29 Tage haben darf. Zur Modellierung der Regeln dienen folgende umgangssprachliche Sätze:

1. Ein Datum besteht aus einer Zahl gefolgt von einem Punkt ., einem Leerschritt ␣ und einer Zeichenfolge, die den Monatsnamen repräsentiert. Danach folgt ein weiterer Leerschritt ␣ und die vierstellige Jahreszahl.
2. Ist der Monatsname aus der Menge {Januar, März, Mai, Juli, August, Oktober, Dezember}, dann darf die Zahl vor dem Monatsnamen zwischen 1 und 31 liegen.
3. Ist der Monatsname aus der Menge {April, Juni, September, November}, dann darf die Zahl vor dem Monatsnamen zwischen 1 und 30 liegen.
4. Ist der Monatsname Februar, dann darf die Zahl vor dem Monatsnamen zwischen 1 und 29 liegen.

Diese Regeln müssen interpretiert werden. Eignet sich umgangssprachliche Beschreibung der Syntax, um Interpretationsfehler zu minimieren?

Formales Modell

Ein formales Modell zur Beschreibung der syntaktischen Struktur bestehen aus definierten Elementen, deren Semantik festgelegt ist. Wir schaffen somit ein zwei-stufiges Vorgehen:

Stufe 1: Wir legen eine Repräsentation zur Beschreibung der Syntax von Repräsentationen fest. Für diese Repräsentation definieren wir eine Interpretationsfunktion, die die Syntax und die Semantik festlegt. Stufe 1 stellt somit ein Werkzeug zur Beschreibung von syntaktischen Strukturen dar.

Stufe 2: Wir benutzen das Werkzeug aus Stufe 1, um syntaktische Strukturen zu beschreiben. Mit Hilfe der Interpretationsfunktion kann einfach die Beschreibung der syntaktischen Struktur „verstanden“ werden.

Syntaxdiagramme

Syntaxdiagramme stellen ein formales Modell zur Beschreibung der syntaktischen Struktur dar. Syntaxdiagramme besitzen mindestens

- einen Namen,
- einen Startpunkt und
- einen Endpunkt.

Weiterhin kann ein Syntaxdiagramm

- Schleifen,
- Alternativen und
- Ausgaben

enthalten.

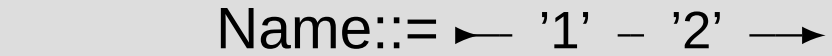
Syntaxdiagramme: Gerüst

Name::= 

Das minimale Syntaxdiagramm besteht aus einem Namen, einem Startpunkt und einem Endpunkt. Die „Bearbeitung“, also der Kontrollfluss, des Diagramms erfolgt indem ein Pfad vom Startpunkt zum Endpunkt durchlaufen wird. Die Menge der Ausgaben aller möglichen Pfade entspricht der Menge der von dem Diagramm erzeugbaren Repräsentationen. Dieses Diagramm besitzt lediglich einen Pfad, der keine Ausgaben erzeugt. Somit kann mit Hilfe dieses Diagramms lediglich die leere Repräsentation (auch leeres Wort ε genannt) erzeugt werden.

Syntaxdiagramme: Ausgaben

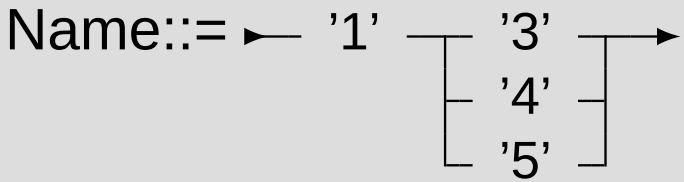
Innerhalb von Syntaxdiagrammen können Ausgaben erzeugt werden, Dies geschieht, indem Zeichenketten in das Syntaxdiagramm eingefügt werden. Erreicht der Kontrollfluss eine Zeichenkette, so wird diese ausgegeben. Die Ausgabe von Syntaxdiagramme beschränkt sich auf Zeichenketten, somit können nur eine Teilmenge alle möglichen Repräsentationen mit Hilfe von Syntaxdiagrammen beschrieben werden. Repräsentationen wie Töne oder ein gesprochenes Wort sind somit nicht modellierbar.



Dieses Syntaxdiagramm erzeugt die mit Hilfe von zwei Ausgaben (eine Ausgabe wäre auch möglich gewesen!) die Repräsentation 12.

Syntaxdiagramme: Alternativen

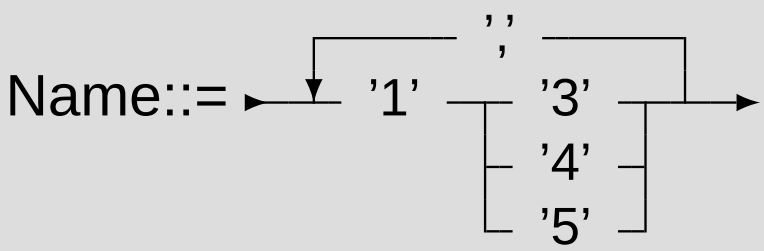
In Syntaxdiagramme können auch mehrere Pfade eingefügt werden. So können Alternativen realisiert werden. Alternative Pfade müssen keine Ausgaben erzeugen.



Dieses Syntaxdiagramm besitzt drei Pfade und erzeugt die Zeichenketten $13, 14, 15$.

Syntaxdiagramme: Schleifen

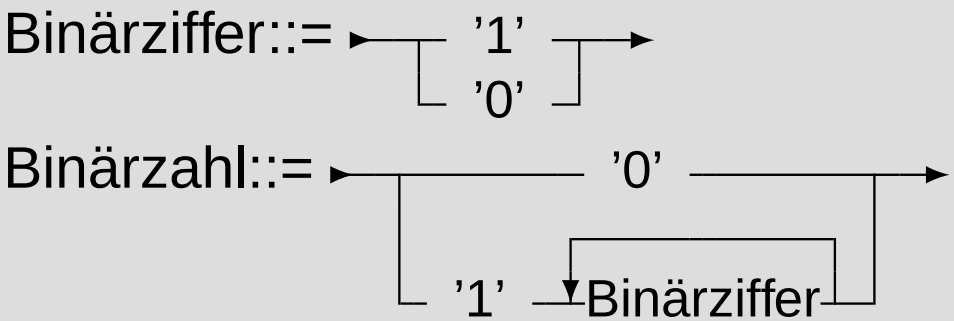
Bisher verlief der Kontrollfluss immer von links nach rechts. Durch die Einführung von Rückkopplungen können Schleifen realisiert werden. Diese Schleifen haben kein an die Schleife geknüpftes Prädikat, das bedeutet, durch eine Rückkopplung entstehen unendlich viele Pfade. In Rückkopplungen können auch Ausgaben erfolgen.



Es sind beliebige Folgen von $13, 14, 15$ erzeugbar. Besteht die Folge aus mehr als einem Element, so sind die Elemente durch ein Komma getrennt. Ein Pfad hat beispielsweise die Ausgabe 13, ein anderer die Ausgabe $15, 14$.

Syntaxdiagramme: Unterdiagramme

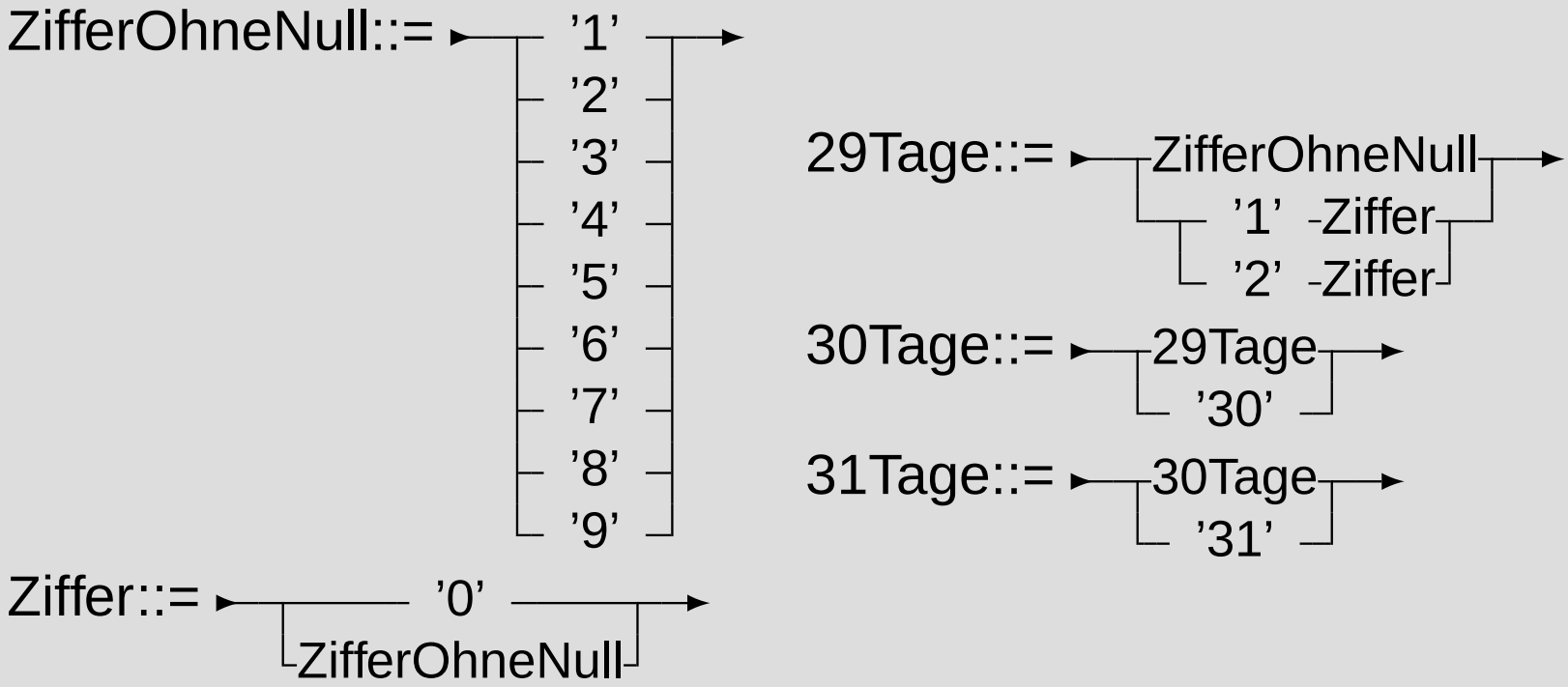
Zur Steigerung der Übersichtlichkeit können in Syntaxdiagrammen andere Syntaxdiagramme eingebunden werden. Dies geschieht mit Hilfe des Namens der Syntaxdiagramme. Zur Unterscheidung zwischen einem Namen und einer Ausgabe sollte die Ausgabe mit Hilfe von Anführungszeichen als Ausgabe gekennzeichnet sein.



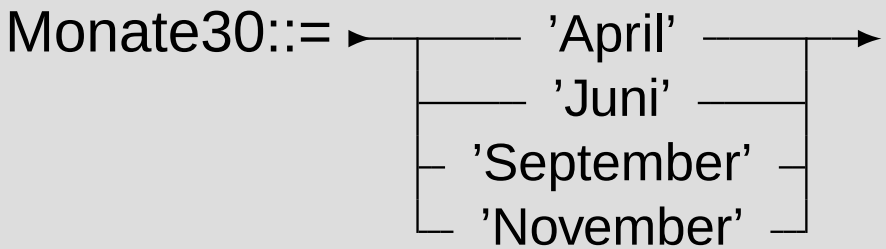
Das Syntaxdiagramm `Binärzahl` stellt Pfade zum Erzeugen aller möglichen Binärzahlen zur Verfügung. Dabei sollen keine fürenden Nullen ausgegeben werden. Mit Ausnahme der Binärzahl '0' fangen also alle Binärzahlen mit einer '1' an.

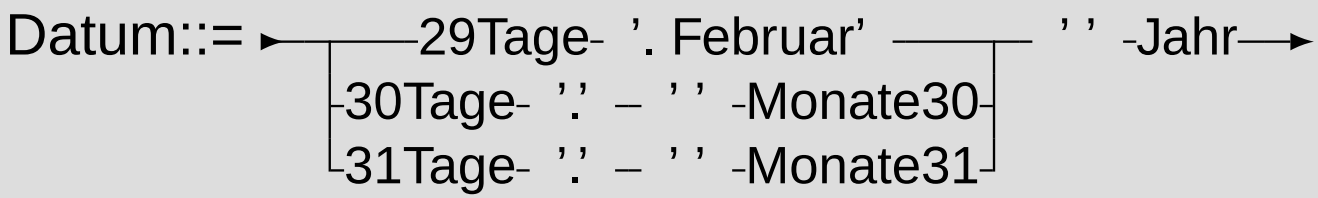
Syntaxdiagramme: Beispiel

Wir konstruieren nun ein Syntaxdiagramm für Daten:



Syntaxdiagramme: Beispiel





Gibt es einen Pfad für die Ausgabe „31. Januar 2001“?

Syntaxdiagramme: Zusammenfassung

+ Syntaxdiagramme stellen eine formale Methode zur Beschreibung von Mengen von Repräsentationen dar.
+ Syntaxdiagramme haben eine festgelegte Interpretationsfunktion.
+ Syntaxdiagramme visualisieren die Syntax, machen sie verständlich.
- Es ist schwer, Syntaxdiagramme mit Hilfe von Computern weiterzuverarbeiten (automatische Syntaxtests). → Notwendigkeit einer anderen formalen Beschreibung der Syntaxregeln.

Grammatiken

Grammatiken sind eine festgelegte formale Beschreibung von Syntaxregeln. Die Idee von Grammatiken ist es, ein System zu erschaffen, mit dem alle Wörter einer Sprache erzeugt werden können. Hierbei sind Grammatiken ähnlich den Syntaxdiagrammen, bei denen durch Durchlaufen aller Pfade alle Wörter erzeugt werden. Eine Grammatik G ist ein 4-Tupel (N, T, P, S). Die Komponenten dieses Tupels sind:

N bezeichnet die Menge der Nichtterminalsymbole. Nichtterminale sind vergleichbar mit den Namen der Syntaxdiagramme. Nichtterminale können weiter ersetzt werden. Ein Wort darf keine Nichtterminale besitzen.

T bezeichnet die Menge der Terminalsymbole. Terminale sind Symbole unseres Alphabets.

Grammatiken (2)

P bezeichnet die Menge der Produktionsregeln. Produktionsregeln sind Regeln zum Ersetzen von Terminal– und Nichtterminalsymbolen durch andere Terminal– und Nichtterminalsymbole.

S bezeichnet das Nichtterminal, mit dem die Ersetzung immer anfängt.

Der Sprachwissenschafter Chomsky hat Grammatiken wissenschaftlich untersucht und sie in verschiedene Klassen eingeteilt. Wir betrachten hier nicht alle Klassen, sondern nur sogenannte Typ-2-Grammatiken. Typ-2-Grammatiken stellen besondere Anforderungen an den Aufbau der Produktionen, was dazu führt, dass es Sprachen gibt, die nicht durch Typ-2-Grammatiken erzeugt werden können. Für unsere Zwecke sind diese Grammatiken jedoch ausreichend.

Grammatiken: Beispiel

Sei $G = (N, T, P, S)$ eine Grammatik mit:

$$\begin{aligned} N &= \{A\} \\ T &= \{a, b\} \\ P &= \{A \to aAa, A \to bAb, A \to \epsilon\} \\ S &= \{A\} \end{aligned}$$

Da wir uns auf Typ-2-Grammatiken beschränkt haben, darf in der Produktionenmenge P nicht jede Ersetzung enthalten sein. Erlaubte Produktionen ersetzen genau ein Nichtterminal durch eine beliebige Kombination von Terminalen und Nichtterminalen. Es ist ebenfalls erlaubt Nichtterminale auf das leere Wort abzubilden.

Grammatiken: Alternativen

Mit Hilfe des Symbols $|$ können Alternativen in der Menge der Produktionen beschrieben werden. Die Menge P lässt sich mit Alternativen folgendermaßen beschreiben:

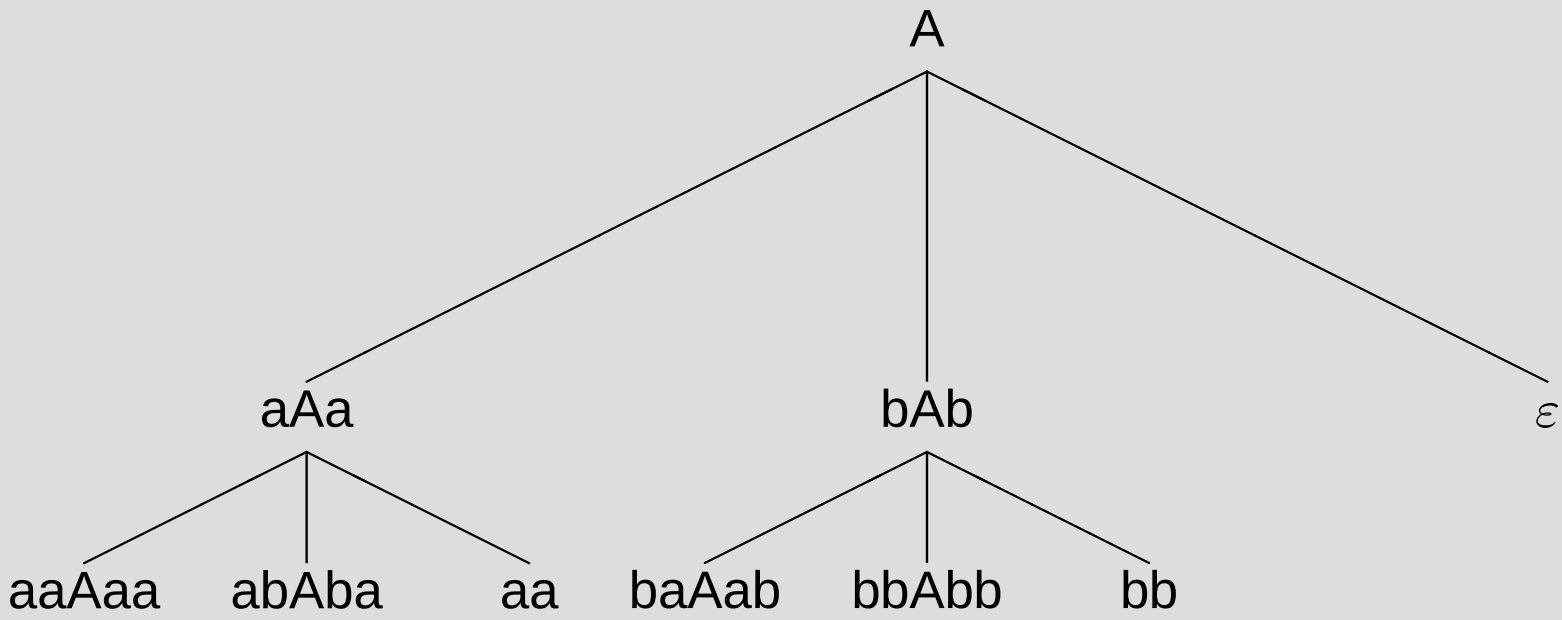
$$P \quad = \quad \{A \rightarrow aAa|bAb|\epsilon\}$$

Wichtig ist das Verständnis, das durch die Verwendung der Alternativen sich nicht die Menge der Produktionen veringert. Alternativen stellen lediglich eine Erleichterung beim Niederschreiben dar. Die Anzahl der Produktionen in P bleibt weiterhin 3.

Das Symbol $\rightarrow$ wird häufig durch die Zeichenfolge `::=` umschrieben. Dies kann sinnvoll sein, wenn das Symbol $\rightarrow$ nicht im Zeichenvorrat verfügbar ist.

Grammatiken: Syntaxbäume

Um festzustellen, welche Sprache L von einer Grammatik G erzeugt wird, erstellt man einen Syntaxbaum (auch Ableitungsbaum genannt). Die Wurzel des Baumes ist immer das Startnichtterminal der Grammatik. Bei unendlichen Sprachen sind die Syntaxbäume ebenfalls unendlich groß. Der folgende Syntaxbaum zeigt die ersten zwei Ableitungen für die Grammatik G:



Grammatik für Datumsangabe

$$
\begin{aligned}
N &= \{ \quad \text{ZifferOhneNull}, \text{Ziffer}, \text{Monate29}, \text{30Tage}, \text{Monate31}, \\
&\quad\ \ \text{Monate30}, \text{Monate31}, \text{Jahr}, \text{JahrRest}, \text{Datum} \ \} \\
T &= \{ \quad 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, \text{Januar}, \text{Februar}, \text{März}, \text{April}, \text{Mai} \\
&\quad\ \ \text{Juni}, \text{Juli}, \text{August}, \text{September}, \\
&\quad\ \ \text{Oktober}, \text{November}, \text{Dezember}, ., , \ \} \\
S &= \{ \quad \text{Datum}\}
\end{aligned}
$$

Grammatik für Datumsangabe

$P = \{$ ZifferOhneNull $\rightarrow' 1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9'$,

Ziffer $\rightarrow$ ZifferOhneNull$|'0'$, 29Tage $\rightarrow'$ $1'$Ziffer$|'2'$Ziffer,

30Tage $\rightarrow$ 29Tage$|'30'$, 31Tage $\rightarrow$ 30Tage$|'31'$,

Monate30 $\rightarrow$ 'Januar'|'März'|'Mai'|'Juli'|'August',

Monate30 $\rightarrow$ 'Oktober'|'Dezember',

Monate31 $\rightarrow$ 'April'|'Juni'|'September'|'November',

Jahr $\rightarrow$ ZifferOhneNull|JahrRest,

JahrRest $\rightarrow$ Ziffer JahrRest$|\varepsilon$,

Datum $\rightarrow$ 29Tage '. ''Februar' ' 'Jahr,

Datum $\rightarrow$ 30Tage '. 'Monate30 ' 'Jahr,

Datum $\rightarrow$ 31Tage '. 'Monate31 ' 'Jahr $\}$

Ableiten von Wörtern

Ein Wort w heißt syntaktisch korrekt bezüglich der Grammatik G, wenn es eine Folge von Ableitungen gibt, die das Wort w erzeugen. Ein Wort w heißt syntaktisch falsch bezüglich der Grammatik G, wenn keine Folge von Ableitungen für w existiert.

Ist das Wort ’12. Februar 2001’ syntaktisch korrekt bezüglich der Grammatik G?

Ableiten von Wörtern

1.	29Tage. Februar Jahr	Ersetze 29Tage
2.	1Ziffer. Februar Jahr	Ersetze Ziffer
3.	12. Februar Jahr	Ersetze Jahr
4.	12. Februar ZifferOhneNullJahrRest	Ersetze ZifferOhneNull
5.	12. Februar 2JahrRest	Ersetze JahrRest
6.	12. Februar 2ZifferJahrRest	Ersetze Ziffer
7.	12. Februar 20JahrRest	Ersetze JahrRest
8.	12. Februar 20ZifferJahrRest	Ersetze Ziffer
9.	12. Februar 200JahrRest	Ersetze JahrRest
10.	12. Februar 200ZifferJahrRest	Ersetze Ziffer
11.	12. Februar 2001JahrRest	Ersetze JahrRest
12.	12. Februar 2001	Fertig

Grammatiken und Programmiersprachen

- Jede Programmiersprache besitzt eine Grammatik.
- Mit der Grammatik kann die Syntax von Programmen geprüft werden.
- Mit Hilfe der Grammatik kann eine Programmiersprache „syntaktisch“ verstanden werden.