252-0027
Einführung in die Programmierung

1.0 EBNF

Thomas R. Gross

Department Informatik
ETH Zürich

- **EBNF Regel besteht aus:**

LHS ⟸ RHS

  - Linke-Seite (Left-Hand Side, LHS)
  - Rechte-Seite (Right-Hand Side, RHS)
  - ⟸ (trennt LHS von RHS, ausgesprochen «ist definiert als»)
- **LHS**
  - Ein Wort (kursiv, kleingeschrieben) – der Name der EBNF Regel
- **RHS**
  - Die genaue Beschreibung für den Namen (d.h., der LHS) durch
    - Zeichen (stellen das Zeichen da, d.h. wir erwarten dieses Zeichen und kein anderes) – nicht kursiv
    - Namen (von EBNF Regeln) – kursiv und kleingeschrieben
    - Kombinationen der vier Kontrolelemente («control forms») (auf folgenden Seiten)

EBNF «control forms»

- **Vier Kombinationsmöglichkeiten («control forms») die Sie in Java wiederfinden werden**
  - Aufreihung («sequence»)
  - Entscheidung («decision») – Auswahl und Option
  - Wiederholung («repetition»)
  - Rekursion («recursion»)
- **Kombinieren (in der RHS) EBNF Regeln**

Kombinieren mit Auswahl und Optionen

- **Auswahl – aus Alternativen**
  - Eine Menge von Alternativen
    - Reihenfolge unwichtig
  - Durch **|** (gesprochen *senkrechter Strich*) («stroke») getrennt
  - Alternativen folgen den EBNF Bestimmungen für die RHS (Rechte-Seite)
- **Auswahl Beispiel**

digit **⟸ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0**

---

raum **⟸ E 12 | D 28**

Kombinieren mit Optionen

- **Option**
  - Element(e) in **[** und **]** (eckige Klammern) («square bracket»)
    - Element muss EBNF Bestimmungen für die RHS folgen
  - Kann gewählt werden, muss aber nicht

- **Drei Beispiele**
  - *initials* ⟸ **T [R] G**
  - *raum* ⟸ **M L [(D 28) | (E 12)]**
  - *zahl* ⟸ **0 x 0 [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9]**

Beispiel

- **Oft Auswahl (..|..) und Option ([..]) kombiniert**
  - EBNF Beschreibung *zahl*

    digit ⟸ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

    vorzeichen ⟸ [+ | -]

    zahl ⟸ *vorzeichen digit*
  - 1, +2, -3, +0, 4 … sind legale Symbole
- **Wie drücken wir aus, dass nicht gewählt wurde?**
  - **ε (**Ausgesprochen «epsilon»**) – die leere Zeichenfolge**
  - **ε erscheint nicht in Symbolen**

EBNF Beschreibungen

- **Nicht immer eindeutig**
  - *initials* ⟸ T **[** R **]** G
  - *initials* ⟸ **(** T G **) | (** T **[** R **]** G **)**
- **«Überspringen» einer Option**
  - *vorzeichen* ⟸ **[+ | -]**
  - *vorzeichen* ⟸ **+ | - | ε**

- Nach *digit* kommt – optional – etwas [...]
- Wenn die Option (...) genommen wird dann lässt [*digit*] (denn das war ...!) eine *digit* zu – wenn wir das wollen
- Beide Regeln lassen die selben Symbole zu

digit ⟸ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

number ⟸ *digit* **[** *digit* **]**

number ⟸ *digit* **[[** *digit* **]]**

1.1.3 Kombination mit Wiederholungen

- **Wiederholung**
  - Der zu wiederholende Ausdruck steht zwischen **{** und **}** (geschweifte Klammer) («curly braces»)
  - Kann 0, 1, … wiederholt werden
  - Immer daran denken: 0 Wiederholungen heisst – fehlt!

Kombination mit Wiederholungen

- **Wiederholung**
  - Der zu wiederholende Ausdruck steht zwischen **{** und **}** (geschweifte Klammer) («curly braces»)
  - Kann 0, 1, … Mal wiederholt werden
  - Immer daran denken: 0 Wiederholungen heisst – fehlt!

- **Beispiel**

digit ⟸ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

folge ⟸ *digit* { *digit* }

<digit> ⟸ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

<folge> ⟸ <digit> { <digit> }

1.1.4 EBNF Beispiele

EBNF Beispiel (i1)

EBNF Description: *integer*

$$digit \Leftarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$integer \Leftarrow [+|-]digit\{digit\}$$

Was bestimmt diese EBNF Beschreibung?

- **Einfache (ganze) Zahlen**
- **Umgangssprachlich:**
  - Eine *digit* ist definiert als eines der Zeichen/Buchstaben 0 … 9
  - Eine *integer* ist definiert als eine Folge von 3 Elementen
    - Ein optionales Vorzeichen (wenn es vorhanden ist, dann muss es eine der Alternativen + oder – sein)
    - Eine *digit*
    - Eine Wiederholung von 0 oder mehr Auftreten von *digit* wobei jede *digit* eine der Alternativen der *digit* Regel ist (und die Alternativen unabhängig gewählt werden)

EBNF Beschreibungen

- **Reihenfolge der Regeln und gewählte Namen unwichtig**

  EBNF Description *integer* (i2)

  integer ⟸ [+ | -] *digit* { *digit* }

  digit ⟸ 0| 1| 2| 3| 4| 5| 6| 7| 8| 9

- **oder auch**

  EBNF Description *zahl* (i3)

  zahl ⟸ [+ | -] *ziffer* { *ziffer* }

  ziffer ⟸ 0| 1| 2| 3| 4| 5| 6| 7| 8| 9

- **<u>Konvention</u>: von einfach nach komplex, relevante Namen**
- **Für jeden Namen eine (und nur eine) Regel**
- **Name der letzten Regel ist der Name der relevanten Beschreibung**

Übersicht

- **Es gibt vier elementare Ausdrucksmöglichkeiten in EBNF**
  - Schon gesehen: Aufreihung, Auswahl/Option, Wiederholung
- **Sie lernen EBNF Beschreibungen zu lesen und verstehen**
- **Sie lernen zu entscheiden ob ein Symbol legal ist (für eine EBNF Beschreibung)**
- **Sie können entscheiden ob zwei EBNF Beschreibungen äquivalent sind**
- **Sie lernen EBNF Beschreibungen zu erstellen**
- **Sie lernen den Unterschied zwischen Syntax und Semantik**

1.2 Symbole und EBNF Beschreibungen

- **Wie können wir eine EBNF Beschreibung wie ein Schiedsrichter interpretieren?**
  - Gegeben eine EBNF Beschreibung und ein Symbol
  - Symbol: eine Folge von Zeichen
  - Schiedsrichter entscheidet ob das Symbol *legal* ist oder nicht (für diese EBNF Beschreibung)
- **Symbol legal gemäss einer Regel: alle Zeichen des Symbols stimmen mit den Elementen der Regel überein**

Zeichen im Symbol – Elemente der Regel

- **Genaue Übereinstimmung: legal**
  - Vergleich Zeichen im Symbol mit Elementen der Regel
    - Von Links nach Rechts
    - Zeichen für Zeichen
- **Ende des Symbols**
  - Es darf kein (nicht-optionales) Element der Regel übrig bleiben
- **Keine weitere Regel**
  - Es darf kein Zeichen im Symbol übrig bleiben
- **Nur dann sprechen wir von Übereinstimmung**
  - Sonst: Symbol *nicht legal* oder *illegal*

- **Beispiel (mit *digit*):** 6 legal, 86 nicht legal
- **Beispiel (mit *digit*):** ***sign*** $\Leftarrow$ **[+ | -]**

signed_number $\Leftarrow$ ***sign digit*** 9, +9, -9 legal, 09, +-9 nicht

Informeller Beweis:

- **Genaue Übereinstimmung: legal**
  - Es darf kein Zeichen im Symbol übrig bleiben
  - Es darf kein (nicht-optionales) Element der Regel übrig bleiben
  - Nur dann sprechen wir von Übereinstimmung
- **Sonst: Symbol nicht legal, illegal**

Animation

1.2.1 Informelle Beweise

- **Zeige dass X mit *integer* übereinstimmt (Beschreibung i1)**
  - Start: 1. Element (optionales Vorzeichen)
    - Option gewählt oder nicht gewählt
  - nächstes Zeichen des Symbols muss mit Zeichen übereinstimmen
    - Zeichen durch *digit* Regel bestimmt
    - Wähle auf der RHS von *digit*
  - Keine oder mehr Wiederholungen

Informelle Beweise

- **Zeige dass +70 mit *integer* übereinstimmt (Beschreibung i1)**
  - Start: 1. Element (optionales Vorzeichen)
    - Option *gewählt* oder nicht gewählt
  - nächstes Zeichen des Symbols muss mit Zeichen übereinstimmen
    - Zeichen durch *digit* Regel bestimmt
    - Wähle 7 auf der RHS von *digit*
  - Keine oder *mehr* Wiederholungen
    - Eine Wiederholung – Zeichen durch *digit* Regel bestimmt
    - Wähle 0 auf der RHS von *digit*
- **+70 ist legal**

Animation

Informelle Beweise

- **Wissen (oder probieren) richtige Anzahl von Wiederholungen**
- **Wenn Beschreibung nicht eindeutig dann finden wir die richtige Alternative**

Animation

Mehr Beispiele

- **1249**
- **-320**
- **+445**

Mehr Beispiele

- **1’249**
- **A15**
- **345-**

1.2.2 Tabellen

- **Formaler als Umgangssprache**
- **Kompakter**
- **1. Zeile: Name der EBNF Regel, mit der das Symbol übereinstimmen soll**
- **Letzte Zeile: Symbol**

Tabellen

- **Jede Zeile wird aus der Vorgängerzeile durch eine dieser Regeln abgeleitet:**
  1. Ersetze einen Namen (LHS) durch die entsprechende Definition (RHS)
  2. Wahl einer Alternative
  3. Entscheidung ob ein optionales Element dabei ist oder nicht
  4. Bestimmung der Zahl der Wiederholungen
- **Manchmal werden 1&2 in einem Schritt gemacht**

Ableitungsbäume

- **Graphische Darstellung eines Beweises durch eine Tabelle**
  - Oben: Name der EBNF Regel, mit der das Symbol übereinstimmen soll
  - Unten: Symbol
- **Kanten zeigen welche Regeln es uns erlauben von einer Zeile zur nächsten (in der Tabelle) zu gehen**

Schnellübung

Poll

- **Welche dieser Symbole sind legal gemäss der *integer* Beschreibung (i1)?**

1. +28
2. +0
3. -
4. IX
5. 333-111
6. -354
7. two
8. a2
9. 0
10. $100
11. 007
12. 824

EBNF Description: *integer*

$$digit \Leftarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$integer \Leftarrow [+|-]digit\{digit\}$$

aus dem Archiv…..

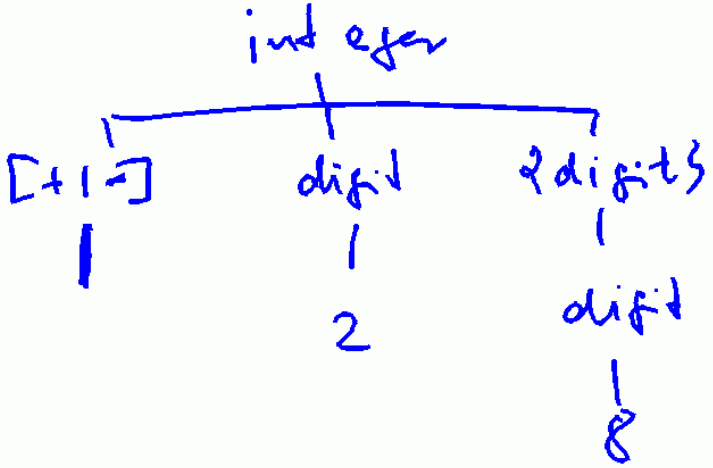
Poll

Schnellübung

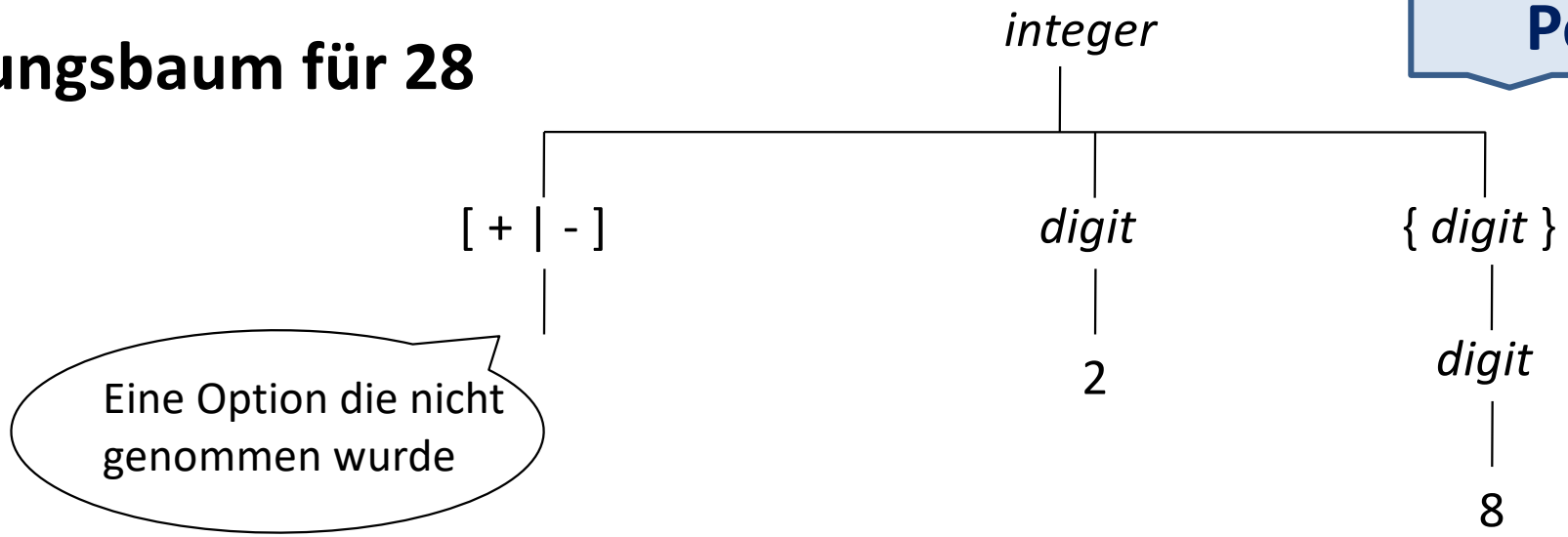
- **Welche dieser Symbole sind legal gemäss der *integer* Beschreibung (i1)?**

1. +28 ✓
2. +0 ✓
3. - ✗
4. IX ✗
5. 333-111 ✗
6. -354 ✓
7. two ✗
8. a2 ✗
9. 0 ✓
10. $100 ✗
11. 007 ✓
12. 824 ✓

option nicht genommen

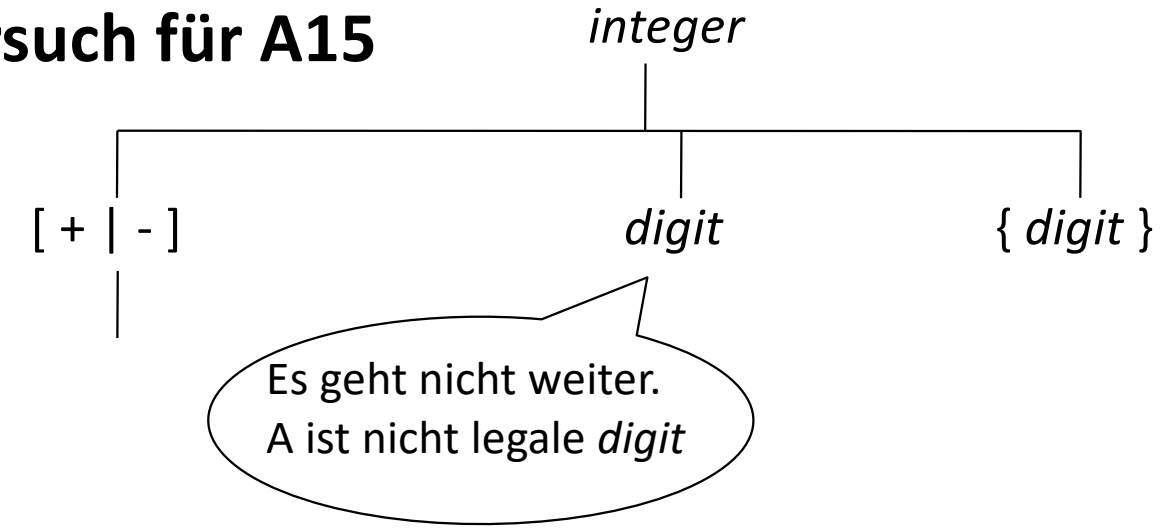


Ableitungsbaum für 28

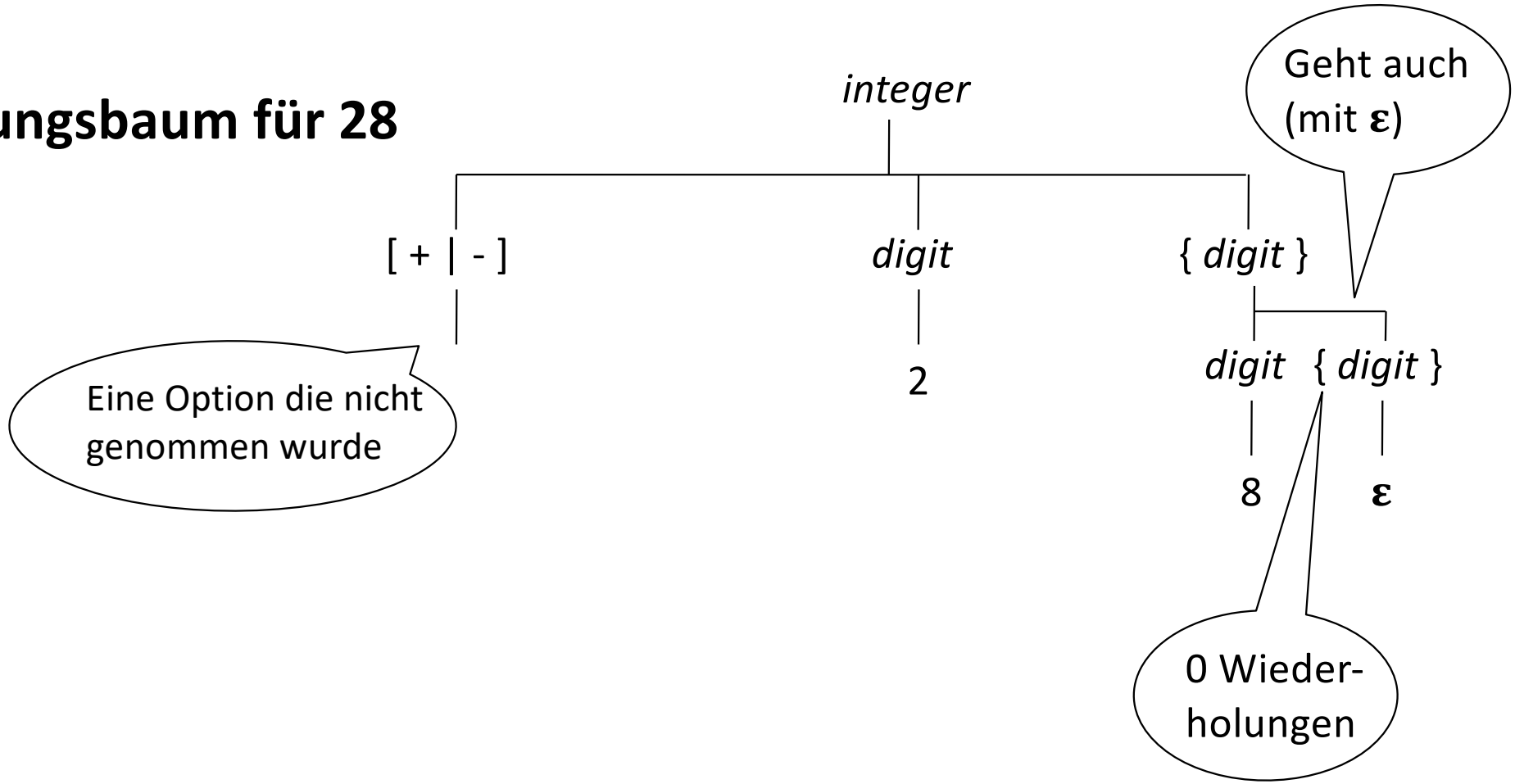


Poll

Ableitungsbaum Versuch für A15



Ableitungsbaum für 28



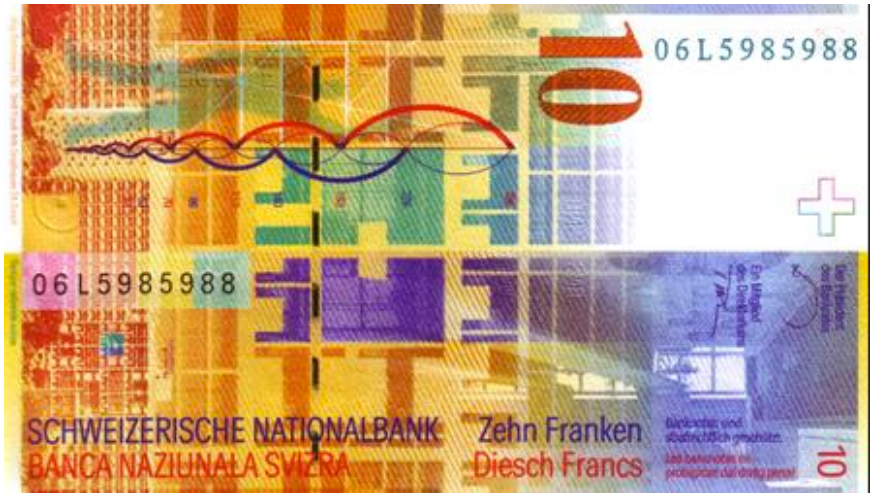
Sonderzeichen

- **Diese acht Zeichen (Buchstaben) haben eine besondere Bedeutung in EBNF Beschreibungen: {, }, [,], |, (,), ⟸**
  - Auch < und > wenn wir Namen nicht kursiv schreiben
- **Was machen wir wenn wir ein "{" in einem Symbol wollen?**
  - z.B. um eine Menge zu beschreiben
- **Antwort: Zeichen in Rahmen** **{**
    - Manchmal alle Zeichen die nicht eine EBNF Regel bezeichnen in einem Rahmen
  - Alternativen (in Textbüchern): In Anführungszeichen, z.B. "("
  - Dann ist " auch ein Sonderzeichen
  - Um ein " in einem Symbol zu bekommen: ""

Äquivalente EBNF Beschreibungen

- **Äquivalent: gleichwertig (sind immer gleich – in einem Kontext)**

äqui-valent



Äquivalent bzgl. Kaufkraft, nicht aber vor einem Automaten der keine Banknoten nimmt

Äquivalente EBNF Beschreibungen

- **Äquivalent: gleichwertig (sind immer gleich – in einem Kontext)**
- **Äquivalente EBNF Beschreibungen erkennen die selben legalen und illegalen Symbole**
  - Jedes mögliche Symbol wird von beiden Beschreibungen als legal (oder illegal) erkannt

Äquivalente EBNF Beschreibungen

- **Jede EBNF Beschreibung definiert eine *Sprache*: Menge der legalen Symbole**
- **Äquivalente EBNF Beschreibungen erkennen die selben legalen und illegalen Symbole**
  - Jedes mögliche Symbol wird von beiden Beschreibungen als legal (oder illegal) erkannt
  - Die Sprachen der EBNF Beschreibungen sind identisch

Äquivalenz von EBNF Beschreibungen

- **Zwei EBNF Beschreibungen B_1, B_2 definieren die selbe Sprache:**
  - Symbol legal für B_1: dann auch legal für B_2
  - Symbol illegal für B_1: dann auch illegal für B_2
  - Symbol legal für B_2: dann auch legal für B_1
  - Symbol illegal für B_2: dann auch illegal für B_1
- **B_1 und B_2 äquivalent**

Weitere EBNF Beschreibung für *integer* (i4)

EBNF Description: *integer*

$$
\begin{aligned}
sign &\Leftarrow + \mid - \\
digit &\Leftarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
integer &\Leftarrow [sign]\, digit\{digit\}
\end{aligned}
$$

Andere Beschreibung für *integer* (i4')

EBNF Description: *integer*

$$
\begin{aligned}
sign &\Leftarrow + \mid - \\
digit &\Leftarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
integer &\Leftarrow [sign]\{digit\}
\end{aligned}
$$

Noch eine andere Beschreibung

- **Äquivalent zu früherer Beschreibung?**

- **Symbol legal gemäss 1. Beschreibung: legal gemäss dieser Beschreibung**
- **Gilt auch die Umkehrung?**

Noch eine andere Beschreibung

- **Symbol legal gemäss 1. Beschreibung: legal gemäss dieser Beschreibung**
- **Gilt auch die Umkehrung?**
- **Nein: + ist jetzt legal, - ist jetzt legal**

- **Nein: ε (leere Zeichenfolge) jetzt eine legale *integer***

Andere Zahlendarstellungen

- **Wir möchten auch Zahlen mit Hochkomma (z.B., 1’412) als *integer* erkennen**
- **Fügen wir also ‘ zu *digit* als Alternative hinzu**
- **EBNF Beschreibung *comma_integer* (ci1)**
  - *sign* ⟸ + | -
  - *comma_digit* ⟸ 0| 1| 2| 3| 4| 5| 6| 7| 8| 9| ‘
  - *comma_integer* ⟸ [*sign*] *comma_digit* { *comma_digit* }

Frage

- **Beschreibung *comma_integer* lässt Symbole zu, die wir nicht akzeptieren wollen.**
- **Können wir (ci1) so ändern, dass Hochkommas richtig (zur Gruppierung in Tausenderblöcke) gesetzt werden?**
  - Jede Dreier-Gruppe von Ziffern ist durch ein Hochkomma von den links davor geschriebenen Ziffern getrennt.
    - Wenn eine Dreier-Gruppe am Anfang steht, dann steht dort kein Hochkomma (denn es gibt ja keine Ziffern links davor)

EBNF Beschreibung *comma_integer* (ci2)

sign	⟸	+ | -
digit	⟸	0| 1| 2| 3| 4| 5| 6| 7| 8| 9
comma_separated_digits	⟸	*digit digit digit*
number	⟸	*digit* | *digit digit* | *digit digit digit*
comma_integer	⟸	[*sign*] *number* { **‘** *comma_separated_digits* }

Zurück zu *integer*

- **Welche dieser EBNF Beschreibungen ist äquivalent zur (früheren) Beschreibung (i1)?**

A (i5)

$$
\begin{aligned}
sign &\Leftarrow [+|-] \\
digit &\Leftarrow 0|1|2|3|4|5|6|7|8|9 \\
integer &\Leftarrow sign\ digit\{digit\}
\end{aligned}
$$

B (i6)

$$
\begin{aligned}
sign &\Leftarrow [+|-] \\
digit &\Leftarrow 0|1|2|3|4|5|6|7|8|9 \\
integer &\Leftarrow sign\{digit\}digit
\end{aligned}
$$

- **Beide (A und B)**

1.3.1 Syntax und Semantik

- **Syntax: Form**
- **Semantik («semantics»): Bedeutung («meaning»)**

- **Syntax legt nur die Form fest.**

Syntax und Semantik

- **Syntax: Form**
- **Semantik («semantics»): Bedeutung («meaning»)**

- **Syntax legt nur die Form fest.**

Alle lesenden Schiffe riechen gelb.

Der Herrscher dachte an sich selbst zuletzt.

Syntax und Semantik

- **EBNF beschreibt nur die Syntax**
- **Für Programmiersprachen: zwei wichtige Semantik Fragen:**
  1. Können unterschiedliche Symbole die selbe Bedeutung haben?
  2. Kann ein Symbol verschiedene Bedeutungen haben?

Illustration

- **Symbole die wir untersuchen: Namen**
  - Herr Wirth
  - Professor Wirth
  - Niklaus Wirth **können sich auf selbe Person beziehen**
- **Symbol das wir untersuchen: Ausdruck «nächste Vorlesung»**
  - Die «nächste Vorlesung» fällt aus
    - 252-0027, (gestern): keine Vorlesung am Freitag
    - 252-0025, (gestern): keine Vorlesung am Montag

Semantik von *integer*

- **Bedeutung einer Zahl: ihr Wert**
  - 1, +1
  - -0, +0, 0
- **Sollen 0012 und 12 die selbe Bedeutung haben?**
  - Mathematik: ja
  - PIN code: nein

EBNF Beschreibung *integer_set*

- **Mengen von Zahlen**
- **{ Aufzählung von Zahlen }**
- **Zwischen { und } keine, eine oder Reihe von Zahlen, durch Komma getrennt**
  - { 1 } {3, 2} {3, 2, 3} { }
- **In EBNF Regeln, müssen unterscheiden zwischen { und** **{**

EBNF Beschreibung *integer_set*

- **Mengen von Zahlen**
- **{ Aufzählung von Zahlen }**
- **Zwischen { und } keine, eine oder Reihe von Zahlen, durch Komma getrennt**
  - { 1 } {3, 2} {3, 2, 3} { }

EBNF Beschreibung

integer_list ⟸ ***integer*** **{ ,** ***integer*** **}**

integer_set ⟸ **{ [** ***integer_list*** **] }**

Diskussion

- ***integer_list*** **Regel – ähnlich vielen Regeln für Java**
- **Beispiele**
  - { }
  - { 1 }
  - { 2, -5, 18 }
- **Kann durch Tabelle (oder Ableitungsbaum) gezeigt werden**

Diskussion

{ 2, -5, 18 }

- **Lemma: 2 ist eine *integer***
- **Lemma*:* -5 ist eine *integer***
- **Lemma: 18 ist eine *integer***

Tabelle

Welcher Schritt? (Siehe «Tabellen» Slide)

	Regel
integer_set	Anfang jeder Tabelle
{ [*integer_list*] }	Ersetzen von *integer_set* durch RHS (1)
{ *integer_list* }	Option eingeschlossen (3)
{ *integer* { , *integer* } }	Ersetzen von *integer_list* durch RHS (1)
{ *integer* , *integer* , *integer* }	2 Wiederholungen (4)
{ 2 , *integer* , *integer* }	Lemma
{ 2 , -5 , *integer* }	
{ 2 , -5 , 18 }	

Bedeutung von Mengen

- **Wann sind zwei Mengen äquivalent?**
  - Mehrfach Nennungen sind nicht wichtig
  - {1, 2, 3, 3, 2, 2, 2 } äquivalent zu {1, 2, 3}
  - Reihenfolge nicht wichtig
  - { 1, 2, 3 } äquivalent zu { 3, 2, 1 }
- **Kanonische** (in Übereinstimmung mit Regel) **Darstellung: geordnet, von kleinster [links] nach grösster Zahl [rechts]**
  - Die kanonische Darstellung kann *nicht* durch EBNF Regeln erzwungen werden

EBNF Beschreibungen

- **Erstellen Sie eine EBNF Beschreibung so dass Zahlen nicht mit einer Null anfangen (also 007 ist illegal, 7 ist legal).**

zero ⇐ **0**

nonzero ⇐ **1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**

digit ⇐ ***zero*** **|** ***nonzero***

integer ⇐ **[+ | -]** ***nonzero*** **{** ***digit*** **}**

zero ⇐ **0**

nonzero ⇐ **1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**

digit ⇐ ***zero*** **|** ***nonzero***

integer ⇐ **[+ | -]** ***nonzero*** **{** ***digit*** **}**

… aber jetzt ist 0 kein gültiges Symbol

Wie können wir die Beschreibung verbessern?

EBNF Beschreibung *canonic_int*

zero ⟸ **0**

nonzero ⟸ **1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**

digit ⟸ ***zero*** **|** ***nonzero***

canonic_ int ⟸ **(** **[+ | -]** ***nonzero*** **{** ***digit*** **})** **|** ***zero***

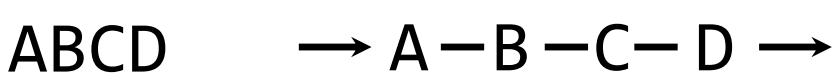
Lässt nur 0, nicht aber +0 oder -0 zu (auf vielfachen Wunsch)

1.4 Graphische Darstellung von EBNF Regeln

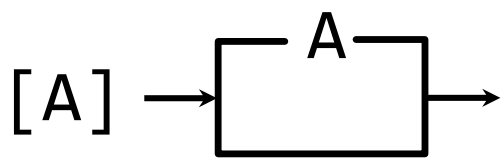
- **Syntax Graph: graphische Darstellung**
  - Kanten (gerichtet) mit Zeichen
- **Pfad durch den Graphen entspricht legalem Symbol**
  - Links anfangen, dann durch Graphen
- **Macht es leicht(er) zu erkennen, welche Zeichen in einem Symbol (in welcher Reihenfolge) auftreten müssen**

Graphische Darstellung von EBNF Regeln

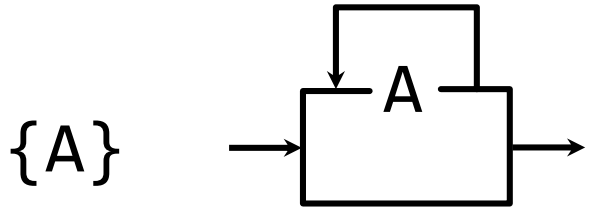
- ***Aufreihung:*** **A B C D als Graph:**



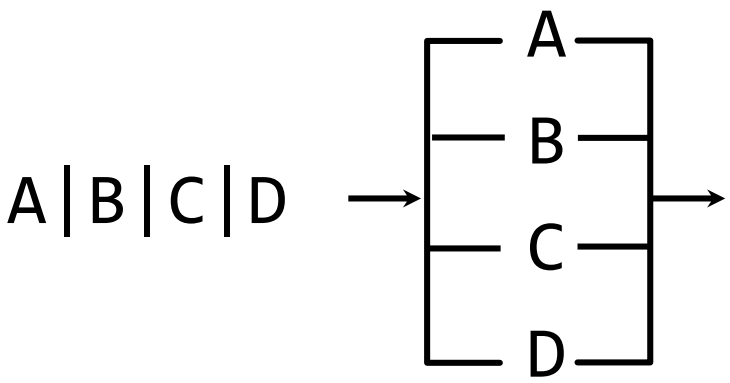
- ***Option:*** **[A] als Graph:**



- ***Wiederholung:*** **{ A } als Graph:**



- ***Auswahl:*** **A | B | C | D als Graph**



Pfad durch Graph: legales Symbol

- **Aufreihung: durch jedes Element in der Reihe**
- **Auswahl: ein Element in der Leiter**
- **Option: entweder obere Kante (mit Element) oder untere (ohne)**
- **Wiederholung: wie Auswahl**
  - Einzige Form die einen Pfeil von rechts nach links hat

Pfade durch Graphen

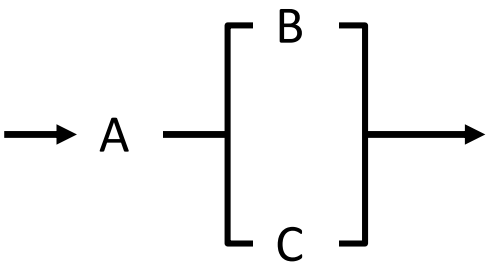
- **EBNF Beschreibung: Menge von EBNF Regeln**
- **Graph: Menge von Graphen**
  - Ein Graph für jede Regel
- **Pfad(e) durch Graphen für Regeln**

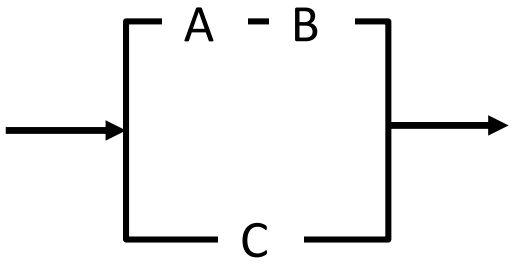
Wofür wir (und) brauchen

- **A B | C**

Wofür wir (und) gebrauchen können

- **A B | C**

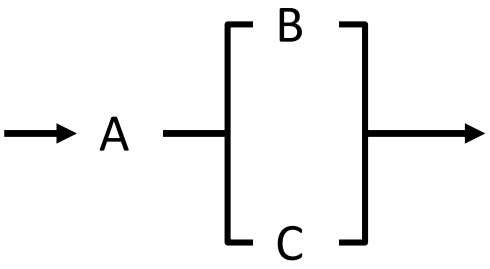


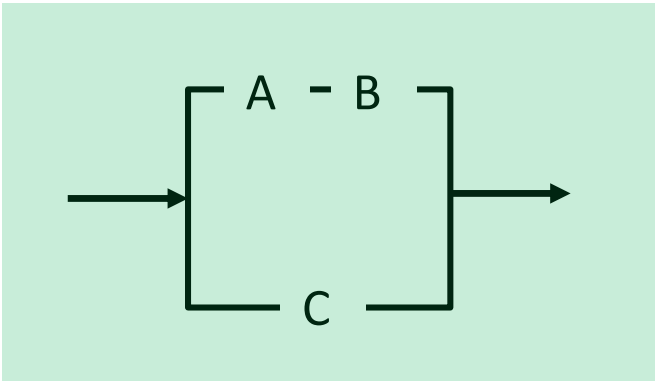


- **Welcher Graph?**

Wofür wir (und) gebrauchen können

- **A B | C**





- **Um Unklarheit zu vermeiden verwenden wir (und)**
  - A (B | C)
  - (A B) | C --- wenn es keine Klammern gibt

    --- Aufreihung «bindet stärker» als Auswahl