**Übungsblatt 1 / Tutorium 05**

Yannick Eschweiler 434446 & Marc Gehring 358302

**Aufgabe 3**

1. G2 = ({A, B, S2}, {., :-, p, q, r, s}, P2, S2) ist die Grammatik

N = {A, B, S2}

T = {., :-, p, q, r, s}

P2 = {S2 → A., S2 → A. S2, A → B, A → B :- B, B → p, B → q, B → r, B → s}

Semantik W(P) = {für x, y ∈ {p, q, r, s}: W(x.) = {x}, W(x :- y.) = Ø,

W(P‘ x.) = W(P‘) ∪ {x}, W(P‘ x :- y.) = {W(P‘) ∪ {x} falls y ∈ W(P‘), W(P‘) sonst}

W(P) ⊆ {p, q, r, s}

1. Da t ∉ T, kann P = (t. s :- t.) nicht in der Sprache erzeugt werden

=> Das Programm ist syntaktisch inkorrekt

1. S2 → A. S2 => S2 → A. A. S2 => S2 → A. A. A. S2 => S2 → A. A. A. A. => A. B :- B. A. B :- B. => B. B :- B. B. B :- B. => s. p :- r. r. q :- r.

=> Das Programm ist syntaktisch korrekt

Semantik:

W(P) = W(s. p :- r. r. q :- r.) = W(P‘ q :- r.) mit P’ = (s. p :- r. r.)

W(P‘) = W(P‘‘ r.) mit P‘‘ = s. p :- r.

W(P‘‘) = W(P‘‘‘ p :- r.) mit P‘‘‘ = s.

W(P’’’’) = {s} => r ∉ W(P’’’) => W(P’’) = W(P’’’) = {s}

=> W(P’) = W(P’’) ∪ {r} = {s, r} => r ∈ W(P’) => W(P) = W(P’) ∪ {q} = {s, r, q}

1. Das Programm P = (r) ist syntaktisch inkorrekt, da eine einzelne Terminale nicht ohne Punkt erzeugt werden kann
2. S2 → A. S2 => S2 → A. A. S2 => S2 → A. A. A. S2 => S2 → A. A. A. S2 => S2 → A. A. A. A. A. => A. A. B :- B. A. B :- B. => B. B. B :- B. B. B :- B. => p. q. q :- r. r. s :- q.

=> Das Programm ist syntaktisch korrekt

Semantik:

W(P) = W(p. q. q :- r. r. s :- q.) = W(P’ s :- q.) mit P’ = (p. q. q :- r. r.)

W(P’) = W(P’’ r.) mit P‘‘ = (p. q. q :- r.)

W(P‘‘) = W(P‘‘‘ q :- r.) mit P‘‘‘ = (p. q.)

W(P‘‘‘) = W(P‘‘‘‘ q.) = W(p.) ∪ {q} = {p, q}

=> r ∉ W(P’’’) => W(P‘‘) = W(P‘‘‘) = {p, q}

=> W(P‘) = W(P‘‘) ∪ {r} = {p, q, r}

=> q ∈ W(P’) => W(P) = {p, q, r, s}

1. Da eine Semantik für eine Sprache klar definiert sein muss, können zwei Begriffe mit gleicher Syntax nur die gleiche Semantik besitzen.
2. Die Aussage stimmt nicht, da man Beispiele finden kann, bei denen zwei Ausdrücke mit unterschiedlicher Syntax die gleiche Semantik aufweisen, zum Beispiel (in Java):

int x = 5; //hat die gleiche Semantik wie: int x = 5-4+2+2;

**Aufgabe 5**

1. L2 = {w ∈ {a, b}\*: auf jedes b folgen direkt mindestens drei a oder auf jedes a folgt direkt mindestens ein b}

G = ({A, B, S}, {a, b}, {S → AB, A → (ε|baaaA|aA), B → (ε|abB|bB)}, S)

1. G = ({S}, {a, b}, {S → (ε|{[{a}]baaa[{a}]}|{[{b}]ab[{b}]}|a|b)}, S)
2. Diagram, schematic

   Description automatically generated

**Aufgabe 7**

1. i) (0101110100)2 = (372)10

ii) (1010000101)2 = -K2(0101111011) = -(379)10

iii) (1101001011)2 = -K2(0010110101) = -(181)10

iv) (0111001100)2 = (460)2

1. (1010010010)2 = -K2(0101101110) = -(366)10
2. Die höchste in Java in einem Integer darstellbare Zahl ist 231-1 oder 2\_147\_483\_647. Addiert man die beiden Zahlen links ergeben sie eine Zahl die größer ist als die höchste darstellbare Zahl. Es entsteht ein Überlauf und die Zahl wird auf die kleinste darstellbare Zahl gesetzt, also -2\_147\_483\_647. Auf der rechten Seite wird die kleinste darstellbare Zahl mit -1 multipliziert. Da sie allerdings dann erneut größer ist als der darstellbare Zahlenbereich, führt der Überlauf wieder dazu, dass sie zur kleinsten darstellbaren Zahl wird.
3. In Java entsteht durch einen Überlauf bei der Addition von 2\_000\_000\_000 und 1\_000\_000\_000 die Zahl -1\_294\_967\_296. Diese addiert zu 1\_294\_967\_296 ergibt somit genau 0.

Aufgabe 9 Diagram

Description automatically generated

1. int: -1\_294\_967\_296

* Es handelt sich um zwei Integer die einfach addiert werden können, da die Syntax stimmt

1. int: 2\_000\_000\_120

* Der char ‘x’ wird in einen Integer überführt, der seinem Unicode-Wert entspricht. Hier: 120. Dies wird dann zu 2\_000\_000\_000 addiert.

1. String: “2000000000x”

* Der Integer 2\_000\_000\_000 wird in einen String umgewandelt, damit die Typen übereinstimmen. Danach werden sie String typisch addiert, d.h. einfach hintereinandergeschrieben.

1. Boolean: true

* Die Zahl 126 aus z wird mit 3 multipliziert und danach zu einem byte explizit gecastet. Das Ergebnis daraus ist der Wert, den der char ‘z‘ auch im Unicode hat. Damit ist die Aussage true. true oder false ergibt danach ebenfalls true.

1. int: -126

* Der long-Wert wird in einen Integer überführt und danach mit z multipliziert. Das Ergebnis ist durch das wegfallen der überschüssigen Bits, die long im vergleich zu int hat -126.

1. float: 0.0

* Die 256 wird in einen byte gecastet und erhält somit den Wert 0. Danach wird die 0 implizit in einen float umgewandelt und anschließend 0 mit 3(beide als float) multipliziert. Das Ergebnis ist dann 0.0 als float.

1. Error: Operation - erfordert zwei Integer, erhält aber String “xy“ und Integer z.
2. float: 122.0

* Der Unicode-Wert von ‘y‘ ist 121. Nach dem Casten von dem char ‘y‘ ist also die Aussage 121 != 121 false. Dementsprechend wird der else-Fall ausgegeben der hier ‘z‘ ist. Der char wird aufgrund des vorstehenden floats jedoch vor der Ausgabe in einen float gecastet, da die Ausgabewerte vom gleichen Typ sein müssen. Der Unicode-Wert von ‘z‘ ist 122 und wird anschließend als float ausgegeben.