Syntax und Semantik von Programmen 1

Modul 5 (v1.0)

Kanonikvorlesung: Foundations of Computing

Heiko Mantel

MAIS, TU Darmstadt, WS10/11

Motivation

Spezifikationssprache in diesem Modul

- IMP
  - eine einfache imperative Programmiersprache

Semantik der „Spezifikationssprache“

- operationelle Semantik
  - syntaxorientierte Auswertungssemantik

Fokus dieses Moduls: Was bedeutet ein Programm?

Übersicht: Modul 5

Syntax einer imperativen Programmiersprache

- Backus-Naur Form
- arithmetische und boolesche Ausdrücke und Kommandos
- abstrakte und konkrete Syntax
- syntaktische Gleichheit versus semantische Äquivalenz

Semantik der Programmiersprache

- Urteile und Kalküle
- Substitutionen und Herleitungen
- operationelle Semantik arithmetischer Ausdrücke
- operationelle Semantik boolescher Ausdrücke
- operationelle Semantik der Kommandos

Syntax (1)

IMP

- eine einfache imperative Programmiersprache
- sequentielle Sprache mit bedingter Verzweigung und Schleifen

Wertebereiche

- N die Zahlen (N steht für engl. numbers)
- T die Wahrheitswerte (T steht für engl. truth values)
- Loc die Programmvariablen (Loc steht für engl. locations)
- Aexp die arithmetischen Ausdrücke (Aexp steht für engl. arithmetic expressions)
- Bexp die booleschen Ausdrücke (Bexp steht für engl. boolean expressions)
- Com die Kommandos (Com steht für engl. Commands)

Syntax (2)

N

- Definition: $N = \{0\} \cup \{ n \mid n \in \mathbb{N} \} \cup \{ -n \mid n \in \mathbb{N} \}$
- Intuition: die negativen und positiven ganzen Zahlen

T

- Definition: $T = \{ true, false \}$
- Intuition: die Wahrheitswerte

Loc

- Der Wertebereich Loc bleibt unterspezifiziert.
- Intuition: die Programmvariablen

Metavariablen

- Platzhalter für Elemente aus einem Wertebereich

Konventionen

- m, n — Platzhalter für Elemente aus N
- t — Platzhalter für Elemente aus T
- X, Y — Platzhalter für Elemente aus Loc

Backus-Naur Form

Backus-Naur Form

- eine kompakte Schreibweise für Produktionsregeln
- kurz: BNF

BNF und Grammatiken

- BNF spezifiziert die Produktionsregeln einer Grammatik (Σ, V, X_0, P)
- Sind Σ, V und X_0 aus dem Kontext klar, dann verwendet man die BNF auch, um Grammatiken zu spezifizieren (siehe nächste Folie).

Folgende BNF

- **u ::= v1 | v2 | … | vn**

ist eine kompakte Schreibweise für folgende Menge von Regeln:

- **u → v1**
  u → v2
  …
  u → vn

Syntax (3)

Aexp

- Definition (durch Angabe einer BNF):
  - a ::= n | X | a+a | a-a | a*a
- Intuition: arithmetische Ausdrücke

Obige Definition ist eine kompakte Schreibweise für folgendes

„Aexp ist eine formale Sprache, die durch die Grammatik (Σ, V, X_0, P) spezifiziert wird, wobei

- $\Sigma = N \cup Loc \cup \{ +, -, * \}$, $V = \{a\}$, $X_0 = a$
- $P = \{ a \rightarrow n \mid n \in N \} \cup \{ a \rightarrow X \mid X \in Loc \}$
  $\cup \{ a \rightarrow a+a,$
  $a \rightarrow a-a,$
  $a \rightarrow a*a \}$

Beachte

Da die Metavariablen n und X in der BNF als Platzhalter für beliebige Elemente aus den Wertebereichen N und Loc agieren, ergeben sich Mengen von Produktionsregeln aus der BNF.

Syntax (4)

Bexp

- Definition (durch Angabe einer BNF):
  - b ::= true | false | a=a | a≤a | ¬b | b∧b | b∨b
- Intuition: boolesche Ausdrücke

Com

- Definition (durch Angabe einer BNF):
  - c ::= skip | X:=a | c;c | if b then c else c fi | while b do c od
- Intuition: Kommandos einer imperativen Programmiersprache
- Konvention:
  „Programm“ wird als Synonym für „Kommando“ verwendet.

Übung: Gib die obigen Definitionen in expandierter Form an.

Syntaktische Korrektheit

Ist ein Ausdruck ein syntaktisch korrektes Programm?

- Diese Frage kann durch die Angabe einer Ableitung mit den Produktionsregeln positiv beantwortet werden.

Beispiel

- c $\rightarrow$ if b then c else c fi
  - $\rightarrow$ if a≤a then c else c fi
  - $\rightarrow$ if a≤a then c else c;c fi
  - $\rightarrow$ if a≤0 then c else c;c fi
  - $\rightarrow$ if a≤0 then v0:=a else c;c fi
  - $\rightarrow$ if a≤0 then v0:=a+a else c;c fi
  - $\rightarrow$ …
  - $\rightarrow$ if v0≤0 then v0:=v1+5 else skip;skip fi
- if v0≤0 then v0:=v1+5 else skip;skip fi
  ist also ein syntaktisch korrektes Programm.

Abstrakte und Konkrete Syntax (1)

Wie wird 5+3*4 gelesen?

- 5+3*4 ist ein syntaktisch korrekter Ausdruck der Sprache Aexp.

Es gibt mehrere Ableitungen für 5+3*4 , z.B.

- a → a*a → a+a*a → 5+a*a → 5+3*a → 5+3*4
- a → a*a → a*4 → a+a*4 → a+3*4 → 5+3*4
- a → a+a → a+a*a → 5+a*a → 5+3*a → 5+3*4
- …

Es gibt zwei mögliche Darstellungen von 5+3*4 als Baum:

- Dadurch lassen sich zwei Klassen von Ableitungen unterscheiden.
  1. Die Regel für * wird vor der für + angewandt (linker Baum).
  2. Die Regel für * wird nach der für + angewandt (rechter Baum).

Abstrakte und Konkrete Syntax (2)

Abstrakte Syntax

- Man betrachtet Texte als Terme.
- Die Baumstruktur der Terme klärt, wie der Text abzuleiten ist.
  - Baumstruktur gibt nur an, in welcher Klasse die Ableitung liegt.
  - Die Ableitungen in einer Klasse werden als äquivalent gesehen.
- Beispiele: die beiden Bäume auf der vorigen Folie

Konkrete Syntax

- Man betrachtet Text ohne Struktur.
- Man fügt Klammern ein oder definiert die Bindungsstärke der Operatoren, um zu klären, wie der Text abzuleiten ist.
- Beispiele: (5+3)*4 und 5+(3*4)

Abstrakte und Konkrete Syntax (3)

Wann verwendet man abstrakte bzw. konkrete Syntax?

Programmieren

- Man verwendet konkrete Syntax beim Programmieren.

Kompilierung eines Programms

- Der Compiler erhält ein Programm in konkreter Syntax.
- Vor der eigentlichen Kompilierung wird die Termstruktur des Programms rekonstruiert, d.h. es geschieht eine Übersetzung in abstrakte Syntax.

Interpretierung eines Programms

- Der Interpreter erhält ein Programm in konkreter Syntax.
- Vor der eigentlichen Interpretierung geschieht eine Übersetzung in abstrakte Syntax.

Theoretische Analysen von Programmen

- Man verwendet meist abstrakte Syntax zur Vereinfachung.

Syntaktische Gleichheit

Wann sind zwei Ausdrücke gleich?

Sind 5+3 und 3+5 gleich?

- Nein, die Ausdrücke sind syntaktisch unterschiedlich.

Sind 5+3 und (5+3) gleich?

- Ja, beide Ausdrücke werden auf die gleiche Weise abgeleitet.

Sind 5+3*4 und (5+3)*4 gleich?

- Nein, denn es gibt Ableitungen von 5+3*4, die keine Ableitungen des zweiten Ausdrucks sind. Ein Gegenbeispiel ist z.B. die Ableitung $a \rightarrow a+a \rightarrow a+a*a \rightarrow 5+a*a \rightarrow 5+3*a \rightarrow 5+3*4$.

Ob zwei Ausdrücke syntaktisch gleich sind, hängt nur davon ab, ob sie die gleichen Ableitungen haben.

Für syntaktische Gleichheit spielen nur die Ableitungen der Ausdrücke eine Rolle, nicht die Bedeutung der Ausdrücke.

Übersicht: Modul 5

Syntax einer imperativen Programmiersprache

- Backus-Naur Form
- arithmetische und boolesche Ausdrücke und Kommandos
- abstrakte und konkrete Syntax
- syntaktische Gleichheit versus semantische Äquivalenz

Semantik der Programmiersprache

- Urteile und Kalküle
- Substitutionen und Herleitungen
- operationelle Semantik arithmetischer Ausdrücke
- operationelle Semantik boolescher Ausdrücke
- operationelle Semantik der Kommandos

Zustände

Definition

Ein **Zustand** ist eine Funktion σ:**Loc**$\rightarrow$**N**. Die **Menge aller Zustände** wird mit Σ bezeichnet.

Intuition (Zustand)

Ein Zustand ordnet jeder Programmvariablen aus Loc einen Wert aus N zu, d.h. eine ganze Zahl.

Intuition (Auswertung)

Der Wert eines arithmetischen oder booleschen Ausdrucks in einem Zustand wird durch eine Auswertungssemantik (oder operationelle Semantik) definiert.

Die Veränderung von Zuständen durch Ausführung von Programmen wird ebenfalls durch eine Auswertungssemantik definiert.

Eine Auswertungssemantik für arithmetische und boolesche Ausdrücke und für Programme wird im Folgenden eingeführt.

Substitutionen

Definition

Eine **Substitution** ist eine Funktion, die eine endliche Menge von Metavariablen als Definitionsbereich hat und jedem Element aus dieser Menge einen Ausdruck zuordnet.

Eine **Grundsubstitution** ist eine Substitution, deren Bildbereich nur Ausdrücke ohne Metavariablen enthält.

Wir verwenden die Notation $[X_1 \mapsto t_1, \ldots, X_n \mapsto t_n]$ für die Substitution mit Definitionsbereich $\{X_1,\ldots,X_n\}$, die X_1 den Ausdruck t_1, … und X_n den Ausdruck t_n zuordnet.

Definition

Die **Anwendung einer Substitution η auf einen Ausdruck α** (geschrieben $\alpha\eta$) resultiert in dem Ausdruck β, den man erhält, indem man in α jedes freie Auftreten jeder Metavariablen X aus dem Definitionsbereich von η durch $\eta(X)$ ersetzt.

Beispiel (Substitution von Metavariablen in Ausdrücken)

(((2+**X**)*(Y-**X**))***X**)**[X↦(1+Z)]** = (((2+(**1+Z**))*(Y-(**1+Z**)))*(**1+Z**))

Auswertungssemantik (1)

Definition

Ein **Urteil** ist ein Schema für Ausdrücke, also ein Ausdruck, der Metavariablen als atomare Ausdrücke enthalten kann. Ein Urteil soll eine gegebene Intuition über einen Sachverhalt formalisieren.

Definition

Ein Ausdruck ξ ist eine **Instanz eines Urteils** ζ, wenn ξ und ζ gleich sind oder man ξ durch Ersetzen einer oder mehrerer Metavariablen in ζ durch geeignete Ausdrücke konstruieren kann.

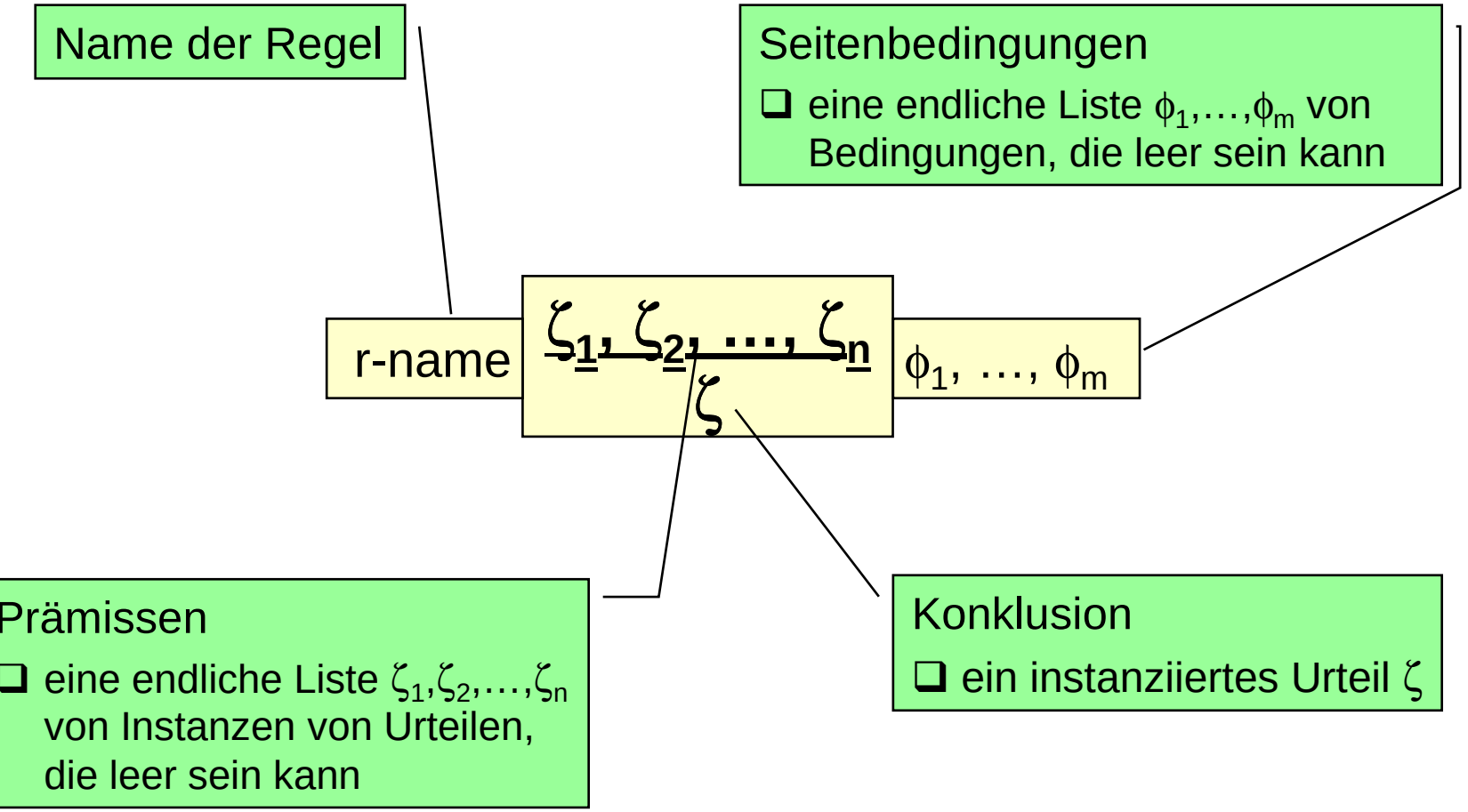
Definition

ξ ist eine **Grundinstanz** , wenn ξ keine Metavariablen enthält.

Konvention: Nachfolgend erlauben wir Metavariablen als atomare Ausdrücke in Ausdrücken aus Aexp, Bexp und Com.

Auswertungssemantik (2)

Notation für Kalkülregeln



Auswertungssemantik (3)

Definition

Ein **Kalkül** ist eine Menge von Kalkülregeln.

Herleitbarkeit von Urteilen

Durch Angabe eines Kalküls wird definiert, welche Instanzen eines Urteils **herleitbar** sind.

Intuition

Ist eine Grundinstanz des Urteils herleitbar, dann sollte der durch diese ausgedrückte Sachverhalt auch zutreffen. (Angemessenheit)

Semantik für Aexp (1)

Informelle Beschreibung der Auswertung eines Ausdrucks

Die Auswertung eines arithmetischen Ausdrucks z.B. der Form **(a1+a2)** in einem Zustand σ lässt sich wie folgt beschreiben:

- Werte den Ausdruck **a1** aus, um eine Zahl **n1** zu erhalten.
- Werte den Ausdruck **a2** aus, um eine Zahl **n2** zu erhalten.
- Der Wert des Ausdrucks **(a1+a2)** ist dann das Resultat der Addition **n1+n2**.

Unterscheidung zwischen Syntax und Semantik

Das Symbol **+** tritt in obiger Beschreibung in zwei Rollen auf:

- in **(a1+a2)** als Symbol in arithmetischen Ausdrücken (Syntax)

und

- in **n1+n2** als Bezeichner der Addition (Semantik).

Beachte

Wird dasselbe Symbol sowohl in der Syntax als auch in Semantik verwendet (wie z.B. + in obigem Beispiel), so sollte man beim Lesen und Schreiben von Ausdrücken darauf achten, dass die Unterscheidung zwischen Syntax und Semantik eingehalten wird.

Semantik für Aexp (2)

Urteil

Wir führen das Urteil $\langle a,\sigma\rangle \Downarrow n$ ein, um auszudrücken, dass

- ein arithmetischer Ausdruck $a \in Aexp$
- in einem Zustand σ
- zu einem Wert $n \in N$ auswertet.

Der Kalkül enthält die Regeln rN, rLoc, r+, r- und r*, die auf den folgenden Folien definiert werden.

Kalkülregeln

r+

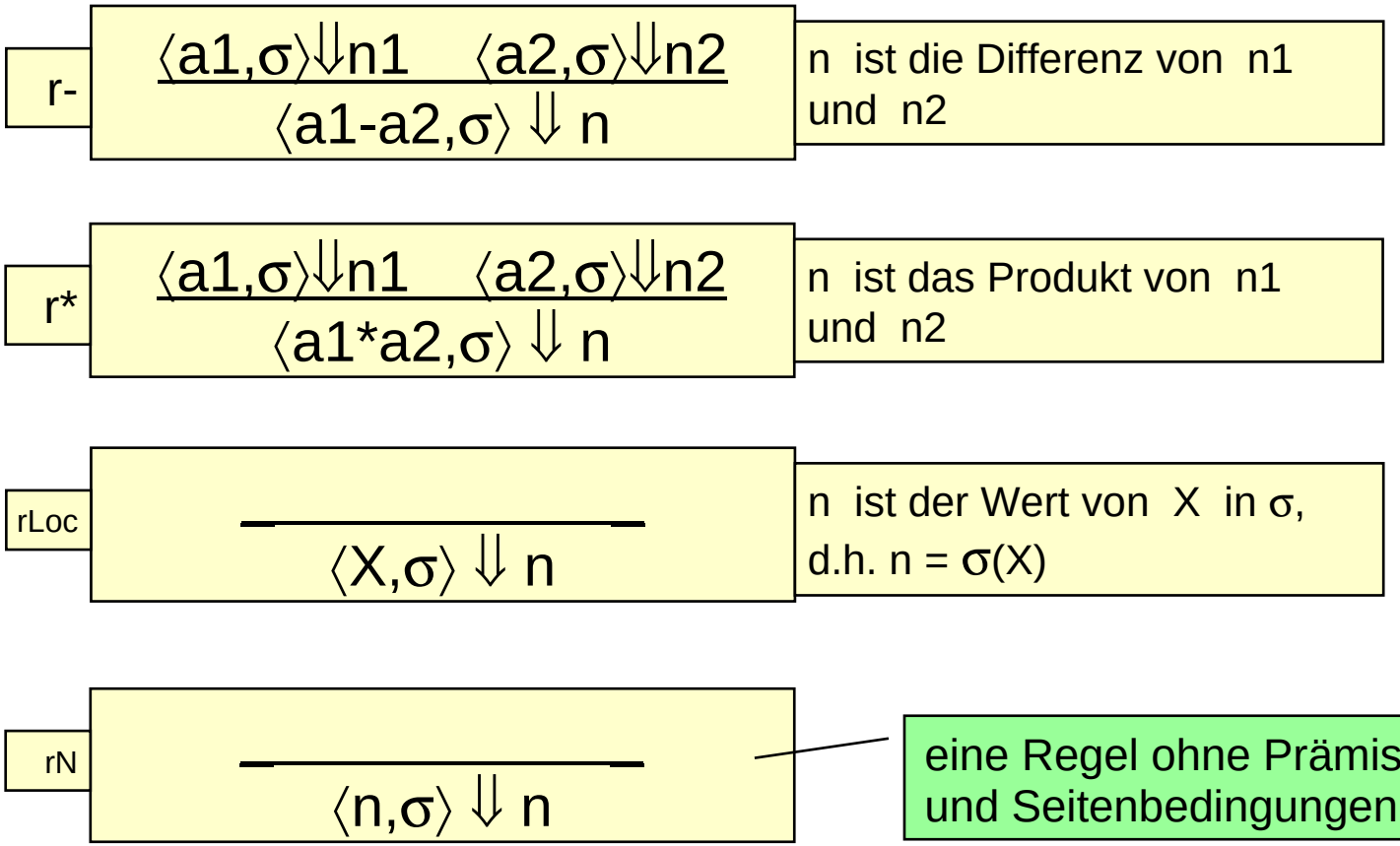
$$\frac{\langle a1,\sigma\rangle \Downarrow n1 \quad \langle a2,\sigma\rangle \Downarrow n2}{\langle a1+a2,\sigma\rangle \Downarrow n}$$

n ist die Summe von n1 und n2

Alle Prämissen und die Konklusion sind Instanzen des Urteils (Syntax), die Seitenbedingung beschränkt Werte (Semantik).

Semantik für Aexp (3)

Kalkülregeln (Fortsetzung)



Instanziierung von Kalkülregeln

Definition

Eine Regel

$$\frac{\xi_1, \xi_2, \ldots, \xi_n}{\xi}$$

ist die **Instanz (bzw. Grundinstanz) einer Kalkülregel**

$$\text{r-name}\ \frac{\zeta_1, \zeta_2, \ldots, \zeta_n}{\zeta}\ \phi_1,\ldots,\phi_m$$

wenn es eine Substitution (bzw. Grundsubstitution) η gibt, so dass $\xi = \zeta\eta$, $\xi_1 = \zeta_1\eta$, … und $\xi_n = \zeta_n\eta$ und die Instanzen $\phi_1\eta$,.. und $\phi_m\eta$ der Seitenbedingungen erfüllt sind.

Beispiel

- Der Zustand $\sigma 0$ ordne allen Programmvariablen den Wert 0 zu. Beide folgenden Regeln sind dann Instanzen der Kalkülregel r+ !

$$\text{r+}\ \frac{\langle 2,\sigma 0\rangle \Downarrow 2 \quad \langle X,\sigma 0\rangle \Downarrow 0}{\langle 2+X,\sigma 0\rangle \Downarrow 2}\ \text{2=2+0 gilt}$$

$$\text{r+}\ \frac{\langle 2,\sigma 0\rangle \Downarrow 3 \quad \langle X,\sigma 0\rangle \Downarrow 1}{\langle 2+X,\sigma 0\rangle \Downarrow 4}\ \text{4=3+1 gilt}$$

Herleitbarkeit von Urteilen

Definition

Die Instanz ξ eines Urteils ist in einem Kalkül **herleitbar** genau dann wenn eine der folgenden beiden Bedingungen erfüllt ist:

- Es gibt eine Kalkülregel der Form

$$\text{r-name}\ \frac{}{\zeta}\ \phi_1,\ldots,\phi_m$$

und eine Substitution η, so dass
  - $\zeta\eta = \xi$ und
  - die Instanzen $\phi_1\eta$,.. und $\phi_m\eta$ der Seitenbedingungen erfüllt sind.

- Es gibt eine Kalkülregel der Form

$$\text{r-name}\ \frac{\zeta_1, \zeta_2, \ldots, \zeta_n}{\zeta}\ \phi_1,\ldots,\phi_m$$

und eine Substitution η, so dass
  - $\zeta\eta = \xi$;
  - die Instanzen $\phi_1\eta$,.. und $\phi_m\eta$ der Seitenbedingungen erfüllt sind und
  - jede der Instanzen $\zeta_1\eta$, … und $\zeta_n\eta$ der Prämissen herleitbar ist.

Semantik für Aexp (4)

Beispiel (textuelle Herleitung)

Der Zustand σ ordne der Programmvariablen x den Wert 5 zu.
Dann ist $\langle 1+x,\sigma\rangle \Downarrow 6$ herleitbar, **weil**

- $\langle 1,\sigma\rangle \Downarrow 1$ durch eine Anwendung der Regel rN hergeleitet werden kann.
- $\langle x,\sigma\rangle \Downarrow 5$ durch eine Anwendung der Regel rLoc hergeleitet werden kann.
  - Die Instanz $\sigma(x)=5$ der Nebenbedingung von rLoc ist gültig.
- $\langle 1+x,\sigma\rangle \Downarrow 6$ aus $\langle 1,\sigma\rangle \Downarrow 1$ und $\langle x,\sigma\rangle \Downarrow 5$ durch eine Anwendung der Regel r+ hergeleitet werden kann.
  - Die Instanz 1+5=6 der Nebenbedingung von r+ ist gültig..

Eine Herleitung muss detailliert angegeben werden.
Insbesondere sollten

- die Namen der verwendeten Regeln angegeben werden,
- alle Prämissen einer Regel hergeleitet werden und
- die Gültigkeit aller Nebenbedingungen gezeigt werden.

Semantik für Aexp (5)

Beispiel (graphische Herleitung)

Statt der textuellen Beschreibung einer Herleitung kann man auch folgende graphische Notation verwenden:

$$\dfrac{\dfrac{}{\langle 1,\sigma\rangle \Downarrow 1}\,r_N \qquad \dfrac{}{\langle x,\sigma\rangle \Downarrow 5}\,r_{Loc}\ \sigma(x)=5 \text{ gilt}}{\langle 1+x,\sigma\rangle \Downarrow 6}\,r_+\ 1+5=6 \text{ gilt}$$

Eine graphische Herleitung muss auch detailliert angegeben werden.

Insbesondere sollten (genau wie bei einer textuellen Herleitung)

- die Namen der verwendeten Regeln angegeben werden,
- alle Prämissen einer Regel hergeleitet werden und
- die Gültigkeit aller Nebenbedingungen gezeigt werden.

Semantik für Bexp (1)

Urteil

Wir führen das Urteil $\langle b,\sigma\rangle \Downarrow t$ ein, um auszudrücken, dass

- ein boolescher Ausdruck $b \in Bexp$
- in einem Zustand σ
- zu einem Wert $t \in T$ auswertet.

Der Kalkül enthält die Regeln rtrue, rfalse, r=t, r=f, r≤t, r≤f, r¬t, r¬f, r∧t, r∧f1, r∧f2, r∨t1, r∨t2 und r∨f, die auf den folgenden Folien definiert werden.

Kalkülregeln

$$\text{rtrue}\quad \frac{}{\langle true,\sigma\rangle \Downarrow true}$$

$$\text{rfalse}\quad \frac{}{\langle false,\sigma\rangle \Downarrow false}$$

Semantik für Bexp (2)

Kalkülregeln (Fortsetzung 1. Teil)

r=t

$$\frac{\langle a1,\sigma\rangle \Downarrow n1 \quad \langle a2,\sigma\rangle \Downarrow n2}{\langle a1=a2,\sigma\rangle \Downarrow \text{true}}$$

n1 und n2 sind gleich

r=f

$$\frac{\langle a1,\sigma\rangle \Downarrow n1 \quad \langle a2,\sigma\rangle \Downarrow n2}{\langle a1=a2,\sigma\rangle \Downarrow \text{false}}$$

n1 und n2 sind nicht gleich

r≤t

$$\frac{\langle a1,\sigma\rangle \Downarrow n1 \quad \langle a2,\sigma\rangle \Downarrow n2}{\langle a1\leq a2,\sigma\rangle \Downarrow \text{true}}$$

n1 ist kleiner oder gleich n2

r≤f

$$\frac{\langle a1,\sigma\rangle \Downarrow n1 \quad \langle a2,\sigma\rangle \Downarrow n2}{\langle a1\leq a2,\sigma\rangle \Downarrow \text{false}}$$

n1 ist größer als n2

Semantik für Bexp (3)

Kalkülregeln (Fortsetzung 2. Teil)

$$r\neg t\ \frac{\langle b,\sigma\rangle \Downarrow \text{false}}{\langle \neg b,\sigma\rangle \Downarrow \text{true}} \qquad r\neg f\ \frac{\langle b,\sigma\rangle \Downarrow \text{true}}{\langle \neg b,\sigma\rangle \Downarrow \text{false}}$$

$$r\wedge t\ \frac{\langle b1,\sigma\rangle \Downarrow \text{true} \quad \langle b2,\sigma\rangle \Downarrow \text{true}}{\langle b1\wedge b2,\sigma\rangle \Downarrow \text{true}}$$

$$r\wedge f1\ \frac{\langle b1,\sigma\rangle \Downarrow \text{false}}{\langle b1\wedge b2,\sigma\rangle \Downarrow \text{false}} \qquad r\wedge f2\ \frac{\langle b2,\sigma\rangle \Downarrow \text{false}}{\langle b1\wedge b2,\sigma\rangle \Downarrow \text{false}}$$

$$r\vee t1\ \frac{\langle b1,\sigma\rangle \Downarrow \text{true}}{\langle b1\vee b2,\sigma\rangle \Downarrow \text{true}} \qquad r\vee t2\ \frac{\langle b2,\sigma\rangle \Downarrow \text{true}}{\langle b1\vee b2,\sigma\rangle \Downarrow \text{true}}$$

$$r\vee f\ \frac{\langle b1,\sigma\rangle \Downarrow \text{false} \quad \langle b2,\sigma\rangle \Downarrow \text{false}}{\langle b1\vee b2,\sigma\rangle \Downarrow \text{false}}$$

Semantik für Com (1)

Urteil

Wir führen das Urteil $\langle c,\sigma\rangle \to \sigma'$ ein, um auszudrücken, dass

- ein Kommando $c \in$ Com
- in einem Zustand σ
- zu einem Zustand σ' auswertet.

Der Kalkül enthält die Regeln rsk, r:=, r;, rift, riff, rwht und rwhf, die auf den folgenden Folien definiert werden.

Kalkülregeln

rsk

$$\frac{}{\langle skip,\sigma\rangle \to \sigma}$$

r:=

$$\frac{\langle a,\sigma\rangle \Downarrow n}{\langle X:=a,\sigma\rangle \to \sigma'}$$

σ' ist der Zustand, der X den Wert n und jeder anderen Programmvariablen y den Wert $\sigma(y)$ zuordnet

Semantik für Com (2)

Kalkülregeln (Fortsetzung)

r;
$$\frac{\langle c1,\sigma\rangle \to \sigma'' \quad \langle c2,\sigma''\rangle \to \sigma'}{\langle c1;c2,\sigma\rangle \to \sigma'}$$

rift
$$\frac{\langle b,\sigma\rangle \Downarrow \text{true} \qquad \langle c1,\sigma\rangle \to \sigma'}{\langle \text{if b then c1else c2 fi},\sigma\rangle \to \sigma'}$$

riff
$$\frac{\langle b,\sigma\rangle \Downarrow \text{false} \qquad \langle c2,\sigma\rangle \to \sigma'}{\langle \text{if b then c1else c2 fi},\sigma\rangle \to \sigma'}$$

rwht
$$\frac{\langle b,\sigma\rangle \Downarrow \text{true} \quad \langle c1,\sigma\rangle \to \sigma'' \quad \langle \text{while b do c1 od},\sigma''\rangle \to \sigma'}{\langle \text{while b do c1 od},\sigma\rangle \to \sigma'}$$

rwhf
$$\frac{\langle b,\sigma\rangle \Downarrow \text{false}}{\langle \text{while b do c1 od},\sigma\rangle \to \sigma}$$

Semantische Äquivalenz (1)

Wann sind zwei Ausdrücke semantisch äquivalent?

Die Auswertungssemantik induziert auf natürliche Weise einen semantischen Äquivalenzbegriff

- Beachte den Unterschied zu syntaktischer Gleichheit!

Definition

Zwei **boolesche Ausdrücke $b1,b2 \in Bexp$, die keine Metavariablen enthalten, sind zueinander semantisch äquivalent** wenn für alle Zustände σ die folgenden beiden Bedingungen gelten:

- $\langle b1,\sigma\rangle \Downarrow$ true ist herleitbar genau dann wenn $\langle b2,\sigma\rangle \Downarrow$ true herleitbar ist,
- $\langle b1,\sigma\rangle \Downarrow$ false ist herleitbar genau dann wenn $\langle b2,\sigma\rangle \Downarrow$ false herleitbar ist.

Wann sind zwei boolesche Ausdrücke, die Metavariablen als atomare Ausdrücke enthalten, zueinander äquivalent?

Semantische Äquivalenz (2)

Definition

Zwei **boolesche Ausdrücke b1,b2∈ Bexp** (die Metavariablen enthalten dürfen) **sind zueinander semantisch äquivalent** wenn für alle Grundsubstitutionen η, deren Definitionsbereich alle Metavariablen in b1 und b2 einschließt, und für alle Zustände σ die folgenden beiden Bedingungen gelten:

- $\langle b1\eta,\sigma\rangle \Downarrow$ true ist herleitbar genau dann wenn $\langle b2\eta,\sigma\rangle \Downarrow$ true herleitbar ist,
- $\langle b1\eta,\sigma\rangle \Downarrow$ false ist herleitbar genau dann wenn $\langle b2\eta,\sigma\rangle \Downarrow$ false herleitbar ist.

Beispiel

$(X=5 \wedge Y=X)$ und $(X=Y \wedge Y=3+2)$ sind semantisch äquivalent.

Übung: Zeige formal, dass die Äquivalenz wirklich gilt.

Semantische Äquivalenz (3)

Definition

Zwei **arithmetische Ausdrücke $a1,a2 \in Aexp$ sind zueinander semantisch äquivalent** wenn für alle Grundsubstitutionen η, deren Definitionsbereich alle Metavariablen in a1 und a2 einschließt, und für alle Zustände σ und für alle ganzen Zahlen n die folgende Bedingung gilt:

- $\langle a1\eta,\sigma\rangle \Downarrow n$ ist herleitbar genau dann wenn $\langle a2\eta,\sigma\rangle \Downarrow n$ herleitbar ist.

Definition

Zwei **Kommandos $c1,c2 \in Com$ sind zueinander semantisch äquivalent** wenn für alle Grundsubstitutionen η, deren Definitionsbereich alle Metavariablen in c1 und c2 einschließt, und für alle Zustände σ,σ' die folgende Bedingung gilt:

- $\langle c1\eta,\sigma\rangle \rightarrow \sigma'$ ist herleitbar genau dann wenn $\langle c2\eta,\sigma\rangle \rightarrow \sigma'$ herleitbar ist.

Übersicht: Modul 5

Syntax einer imperativen Programmiersprache

- Backus-Naur Form
- arithmetische und boolesche Ausdrücke und Kommandos
- abstrakte und konkrete Syntax
- syntaktische Gleichheit versus semantische Äquivalenz

Semantik der Programmiersprache

- Urteile und Kalküle
- Substitutionen und Herleitungen
- operationelle Semantik arithmetischer Ausdrücke
- operationelle Semantik boolescher Ausdrücke
- operationelle Semantik der Kommandos

Rückblick

Einige wesentliche Lernziele dieses Moduls

- Wie kann ich die Syntax einer Programmiersprache formal modellieren?
- Wie kann ich die Bedeutung von Programmen in einer gegebenen Programmiersprache formal modellieren?
  - klare Unterscheidung zwischen Syntax und Semantik!!!
- Was ist ein Urteil?
- Was ist eine Auswertungssemantik?
- Wie definiere ich einen natürlichen Äquivalenzbegriff basierend auf einer Auswertungssemantik?
- Was ist der Unterschied zwischen syntaktischer Gleichheit und semantischer Äquivalenz?

Selbsttest: Können Sie Syntax und Semantik Ihrer Lieblingsprogrammiersprache bereits formal modellieren?

❒ Tip: Beginnen Sie mit Teilsprachen, die Sie dann erweitern.

Literatur

Glynn Winskel

The Formal Semantics of Programming Languages; Kapitel 2
The MIT Press, 1993.