

## Vorlesung Höhere Programmiersprachen, WS 2002/03 Teil 2: Formale Semantik

Einleitung

Operationale Semantik  
mit ASMs

Dr. Sabine Glesner

## Inhaltsübersicht HPS WS 2002/03

- Grundlagen (1,2,3)
- Konzepte imperativer Programmiersprachen (3,4)
- Deklarative Programmiersprachen (5,6)
- Objektorientierte Programmiersprachen (6,7)
- Wissenschaftliches Rechnen: Fortran (8,9)
- **Formale Semantik**
  - **Operationale Semantik mit ASMs** (8,9)
  - Operationale Semantik mit natürlicher Semantik und SOS (10)
  - Denotationelle Semantik (11)
  - Axiomatische Semantik (12)
- Skriptsprachen (13)
- Wirtschaftsanwendungen:
  - Cobol (14)
  - Abap/4 (15)

## Literatur zu Semantik

- H. Riis Nielson, F. Nielson: *Semantics with Applications: A Formal Introduction*. Published by John Wiley & Sons 1992, überarbeitete Version von 1999 unter <http://www.daimi.au.dk/~hrn>.
- Peter D. Mosses: *Fundamental Concepts and Formal Semantics of Programming Languages - An Introductory Course*, verfügbar unter <http://www.brics.dk/~pdm/dSprogSem-02/Notes-1.pdf>
- S. Glesner: *ASMs versus Natural Semantics: A Comparison with New Insights*. In: *Abstract State Machines - Advances in Theory and Applications, Proceedings of 10th International Workshop, ASM 2003 Taormina, Italy, March 2003*. Springer LNCS 2589, verfügbar auf Glesners Homepage.
- ASM Homepage: [www.eecs.umich.edu/qasm](http://www.eecs.umich.edu/qasm).

HPS WS 2002/03

Dr. Sabine Glesner

## Warum formale Semantik?

- exakte Definition, wie sich Programme verhalten (natürliche Sprache immer mehrdeutig)
- besseres Verständnis der Programmiersprache
- Grundlage für Programmanalysen
- Grundlage für korrekte Übersetzerimplementierung
- Definition für semantische Äquivalenz
- schnelle Prototyp-Entwicklung
- formale Beweise für Korrektheit (Testen zeigt, dass Fehler vorhanden sind; nicht, dass keine Fehler da sind)

HPS WS 2002/03

Dr. Sabine Glesner

## Übersicht Formale Semantik

- **Konzepte in Programmiersprachen**
  - Was muß überhaupt spezifiziert werden
- Überblick verschiedene Formalismen
- Operationale Semantik
  - Abstrakte Zustandsmaschinen (ASMs)
  - Strukturell operationale Semantik (SOS)
  - natürliche Semantik
- Denotationelle Semantik
- Axiomatische Semantik

HPS WS 2002/03

Dr. Sabine Glesner

## Konzepte in Programmiersprachen Übersicht

- Syntax versus Semantik
- Steuerfluß
- Datenfluß
- Programmierparadigmen

HPS WS 2002/03

Dr. Sabine Glesner

## Syntax versus Semantik

- Syntax von Programmiersprachen:
  - kontextfreie Syntax (überprüft in lexikalischer und syntaktischer Analyse)
  - context-sensitive syntax (überprüft in semantischer Analyse)
  - definiert abstrakten Syntaxbaum (abstract syntax tree, AST)
- Semantik:
  - definiert die Bedeutung von Programmen
  - basierend auf AST definiert
  - drei prinzipielle Methoden: operationale, denotationelle und axiomatische Semantik

HPS WS 2002/03

Dr. Sabine Glesner

## Kompositionalität

- Semantik basierend auf AST definiert
- Kompositionalität von Programmiersprachen:
  - Die Bedeutung eines Programms ergibt sich unmittelbar aus seinen Unterprogrammen, d.h. seinen direkten Unterbäumen.**
- nicht kompositional: gotos
- Aber: Ausnahmebehandlung ist kompositional.

HPS WS 2002/03

Dr. Sabine Glesner

## Steuerfluß

- Sequenzen von Anweisungen
- Auswahl aus Alternativen
- Ausnahmebehandlung
- Iteration
- Prozedurdefinition und -aufruf
- nebenläufige Prozesse
- Austauschen von Nachrichten
- Synchronisation

## Datenfluß

- Berechnung von Werten
- Verwendung berechneter Werte
- Speichermodifikationen
- Bindungen und ihre Gültigkeitsbereiche
- Methodenaufruf
- Funktionsaufruf

## Programmierparadigmen

- Verschiedene Programmierparadigmen betonen verschiedene Steuer- und Datenflußkonzepte
- Imperative Programmiersprachen:
  - sequenzielle Anweisungen, Iteration, Prozedurdefinition und -aufruf, Speichereffekte
- Funktionale Programmiersprachen
  - Berechnung von Werten, Gültigkeitsbereiche von Bindungen, Funktionsdefinition und -aufruf
- Nebenläufige Programmiersprachen
  - nebenläufige Prozesse, Auswahl unter Alternativen, Verschicken von Nachrichten
- Objektorientierte Programmiersprachen
  - wie imperativ mit speziellen Bindungsregeln

## Übersicht Formale Semantik

- Konzepte in Programmiersprachen
  - Was muß überhaupt spezifiziert werden
- **Überblick verschiedene Formalismen**
- Operationale Semantik
  - Abstrakte Zustandsmaschinen (ASMs)
  - Strukturell operationale Semantik (SOS)
  - natürliche Semantik
- Denotationelle Semantik
- Axiomatische Semantik

## Formale Semantik: Überblick

- Operationale Semantik:
  - definiert Programmausführung als Zustandsübergangssystem
- Denotationelle Semantik:
  - definiert Ergebnis der Programmausführung
- Axiomatische Semantik:
  - definiert Prädikate, die an bestimmten Programmpunkten gelten

## Überblick: Operationale Semantik

- definiert, **wie** Programme auszuführen sind
- z.B. Folge von Anweisungen:
  - wird ausgeführt, indem eine Anweisung nach der anderen berechnet wird
  - von links nach rechts
- Zuweisungen ändern den Zustand
- ist eine Abstraktion von der tatsächlichen Programmausführung
  - keine Register, Adressen von Variablen, etc.

## Überblick: Denotationelle Semantik

- Bedeutung von Programmen beschrieben durch mathematische Objekte
- **nur die Effekte** von Berechnungen, nicht der Berechnungsweg, wird definiert
- modelliert über mathematische Funktionen
- z.B. Effekt einer Zuweisung:
  - modelliert mit einer Funktion, die einen gegebenen Zustand in einen neuen Zustand überführt
  - neuer Zustand ist identisch mit dem alten bis auf den Wert der Variablen, die bei der Zuweisung modifiziert wurde

## Überblick: Axiomatische Semantik

- **Zuweisungen** spezifizieren den Effekt der Ausführung von Programmkonstrukten
- Diese Zuweisungen können auch nur Teile des vollständigen Effekts modellieren
  - bestimmte Aspekte evtl. unberücksichtigt
- Verwendung bei Nachweis **partieller Korrektheit**
- axiomatische Semantik stellt ein **logisches System** zur Verfügung

## Übersicht Formale Semantik

- Konzepte in Programmiersprachen
  - Was muß überhaupt spezifiziert werden
- Überblick verschiedene Formalismen
- **Operationale Semantik**
  - Abstrakte Zustandsmaschinen (ASMs)
  - Strukturell operationale Semantik (SOS)
  - natürliche Semantik
- Denotationelle Semantik
- Axiomatische Semantik

HPS WS 2002/03

Dr. Sabine Glesner

## Operationale Semantik

- Idee: Zustände gehen in neue Zustände über
- Abstrakte Zustandsmaschinen (ASMs)
  - Zustand ist Interpretation einer Logik der ersten Stufe
- Strukturell operationale Semantik (SOS)
  - auch small-step semantics genannt
  - (Progr, Zustand) transformiert in (Progr', Zustand') bzw. (Endzustand')
- Natürliche Semantik
  - auch big-step semantics genannt
  - beschreibt vollständige Zustandsübergänge eines Programms in Abhängigkeit der Zustandsübergänge von Teilprogrammen

HPS WS 2002/03

Dr. Sabine Glesner

## Operationale Semantik

- Unterschiede zwischen verschiedenen Formalismen:
  - Wie wird das Programm während der spezifizierten Ausführung behandelt?
  - konstant: ASMs und natürliche Semantik
  - veränderbar: SOS (manchmal auch bei ASMs)
    - Continuation:  
Bezeichnung für noch abzuarbeitendes Programm

HPS WS 2002/03

Dr. Sabine Glesner

## Übersicht Formale Semantik

- Konzepte in Programmiersprachen
  - Was muß überhaupt spezifiziert werden
- Überblick verschiedene Formalismen
- **Operationale Semantik**
  - **Abstrakte Zustandsmaschinen (ASMs)**
  - Strukturell operationale Semantik (SOS)
  - natürliche Semantik
- Denotationelle Semantik
- Axiomatische Semantik

HPS WS 2002/03

Dr. Sabine Glesner

## Abstrakte Zustandsmaschinen (ASMs)

- entwickelt von Yuri Gurevich, Ende der 80er
- alter Name: EVA (evolving algebra)
- Idee:
  - Jeder Zustand ist Interpretation einer Logik
  - Zustandsübergänge ändern die Interpretation eines Teils der Funktionssymbole
- Vorteile: genauig, verständlich, ausführbar, allgemein, skalierbar

HPS WS 2002/03

Dr. Sabine Glesner

## ASM These

Jeder Algorithmus kann als ASM formuliert werden und zwar auf seinem natürlichen Abstraktionsniveau.

vorläufiger Beweis dieser These: große Sammlung von Fallstudien, vgl. ASM-Homepage (nicht nur Programmiersprachen!)

Programmiersprachen-Semantik: Spezifikation von C, Java, SDL (Specification and Description Language), ...

HPS WS 2002/03

Dr. Sabine Glesner

## Grundlegende Begriffe: Prädikatenlogik

- Prädikatenlogik:
- Syntax:
  - definiert Terme und Formeln über einer Variablenmenge, einer Menge von Funktionssymbolen und einer Menge von Prädikatssymbolen
  - All- und Existenzquantor
- Semantik:
  - definiert Wahrheitswert von Formeln bzgl. möglicher Interpretationen
- Allgemeingültig: wahr in allen Interpretationen

HPS WS 2002/03

Dr. Sabine Glesner

## Grundlegende Begriffe: Semantik

- Struktur (ist eine Algebra):
  - definiert ein nicht-leeres Universum  $U$
  - ordnet Funktionssymbolen Funktionen auf  $U$  zu
  - ordnet Prädikatssymbolen Prädikate auf  $U$  zu
- Variablenbelegung: Substitution:
  - ordnet Variablen Werte aus  $U$  zu
- Definiere damit induktiv über Aufbau von Formeln deren Wahrheitswert
  - Allquantor: muß für alle Belegungen der allquantifizierten Variable gelten
  - Existenzquantor: muß für mind. eine Belegung der existenzquantifizierten Variable gelten
- Interpretation = Struktur + Variablenbelegung

HPS WS 2002/03

Dr. Sabine Glesner

## Definition ASM

- ASM hat vier Komponenten  $(\Sigma \cup \Delta, \mathcal{A}, \text{Init}, \text{Trans})$ 
  - Signatur  $\Sigma \cup \Delta$ : zwei disjunkte Mengen
    - $\Sigma$ : Menge der statischen Funktionssymbole
    - $\Delta$ : Menge der dynamischen Funktionssymbole
  - Statische Algebra  $\mathcal{A}$ :
    - interpretiert Funktionssymbole in  $\Sigma$
  - Initiale Zustände *Init*:
    - Gleichungen über  $\mathcal{A}$ , definieren initiale Zustände in  $\mathcal{A}$
  - Übergangsregeln *Trans*:
    - definieren Zustandsübergänge, indem Interpretation dynamischer Funktionssymbole in  $\Delta$  definiert oder modifiziert wird

HPS WS 2002/03

Dr. Sabine Glesner

## Definition ASM (Fortsetzung)

- Zustände einer ASM:
  - die  $(\Sigma \cup \Delta)$ -Algebren, deren Einschränkung auf  $\Sigma$  die statische Algebra  $\mathcal{A}$  ist

HPS WS 2002/03

Dr. Sabine Glesner

## Definition ASM (Fortsetzung)

- Übergangsregeln in *Trans*:  
if *Cond* then *Update*<sub>1</sub> ... *Update*<sub>n</sub> fi
- $\text{Update}_i: f(t_1, \dots, t_m) := t_0, f \in \Delta$
- Sei  $q$  ein Zustand,  $t_i$  Terme über  $\Sigma \cup \Delta$  mit Interpretationen  $x_i$  in  $q$
- $\text{Update}_i$  definiert neue Interpretation von  $f$ :
  - $f(x_1, \dots, x_m) := x_0$  falls  $q \models t_i = x_i$
  - $f(x_1, \dots, x_m) := f_i(x_1, \dots, x_m)$  sonst
- Updates werden parallel ausgeführt

HPS WS 2002/03

Dr. Sabine Glesner

## Beispiel: Speichermodellierung

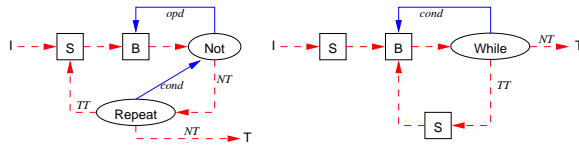
- Funktionssymbol  $S$  spezifiziert Speicherzustand:
  - Argumente sind Terme, die für Variablen stehen
  - Ergebnisterm steht für Belegung der Variable bzw. Speicherstelle
- Zuweisung  $x := v$  an Variable  $x$  ändert den Zustand des Speichers
- Sei  $I(S)$  alte Interpretation
- Neue Interpretation:
  - identisch mit  $I(S)$  bis auf  $I(S(x)) = I(v)$

HPS WS 2002/03

Dr. Sabine Glesner

## Grundlage für ASM-Spezifikation

Grundlage: Abstrakter Syntaxbaum (AST) mit seinen Continuations (statisch berechenbar)



HPS WS 2002/03

Dr. Sabine Glesner

## ASM-Semantik-Spezifikation

- AST mit Continuations
- Knoten im AST haben Sorten:
  - festgelegt durch abstrakte Syntax
  - z.B. While, Assignment, Proc-Call, etc.
- Zustand der ASM enthält  $CT$  (current task)
  - zeigt auf aktuellen Knoten im AST
- Transitionsregeln spezifisch für Knotensorten

HPS WS 2002/03

Dr. Sabine Glesner

## Beispiel While-Schleife

```

if  $CT \in \text{While}$ 
then
  if  $\text{value}(\text{cond}) = \text{true}$  then
     $CT := CT.TT$ 
  else  $CT := CT.FT$ 
fi
fi
    
```

$CT.TT$  steht für "true task",  $CT.FT$  steht für "false task"

HPS WS 2002/03

Dr. Sabine Glesner

## Beispiel Zuweisung

```

if  $CT \in \text{Assign}$  then
   $S(CT.lhs) := CT.rhs;$ 
   $CT := CT.NT$ 
fi
    
```

$lhs$  steht für left-hand side,  $rhs$  steht für right-hand side, bezeichnet linke und rechte Seite der Zuweisung.

Statische Semantik hat geprüft, dass  $lhs$  Speicherstelle bezeichnet.

HPS WS 2002/03

Dr. Sabine Glesner

## Wert von Variablen

```
if  $CT \in \text{Var}$   
  then  $\text{value}(CT) := S(CT.Id)$ ;  
fi
```

## Bindungen und Gültigkeitsbereiche

- Modellierung mit dynamischer Funktion:
  - $\text{Cur\_Bindings} : () \rightarrow \text{Bindings}$   
funktioniert für ungeschachtelte Gültigkeitsbereiche
- Geschachtelte Gültigkeitsbereiche:
  - $\text{Stacklevel} : () \rightarrow \text{Naturals}$
  - aktualisiert beim Betreten und Verlassen von Blöcken
  - $\text{Cur\_Bindings} : \text{Naturals} \rightarrow \text{Bindings}$   
für geschachtelte Gültigkeitsbereiche

## Zusammenfassung ASMs - Teil 1

- geeignet für die Spezifikation aller Arten von Programmiersprachen
  - auch für kompletten Sprachumfang
  - relativ leicht verständlich
- Grundlage AST
  - konstant während spezifizierter Ausführung
  - enthält statisch berechenbare Continuations
- Zeiger (CT, current task) auf aktuell auszuführenden Knoten im AST,
- CT ist Teil des Zustands
- Zustand ist Interpretation (Struktur) einer Logik erster Stufe

## Zusammenfassung ASMs - Teil 2

- Zustände interpretieren statische und dynamische Funktionssymbole
- Zustandsübergänge legen (neue) Funktionswerte für bestimmte Argumente fest
- Berechnete Werte werden Knoten im AST zugeordnet
- Bindungen und Gültigkeitsbereiche werden explizit über Stacks modelliert
- Änderungen im Speicherzustand lassen sich direkt darstellen
- Verteilte ASMs modellieren nebenläufige Prozesse

## Beispiele für ASM-Spezifikationen I

- Folge von Zuweisungen

if  $CT \in \text{Assign}$  then

$S(CT.lhs) := CT.rhs;$

$CT := CT.NT$

fi

Nach Abarbeitung der Zuweisung 1:  
 $CT := CT.NT$



HPS WS 2002/03

Dr. Sabine Glesner

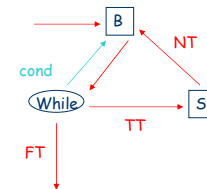
## Beispiele für ASM-Spezifikationen II

- While-Schleife

```

if  $CT \in \text{While}$ 
then
  if  $\text{value}(\text{cond}) = \text{true}$  then
     $CT := CT.TT$ 
  else  $CT := CT.FT$ 
fi

```



Steuerfluß

Datenfluß



Anweisungssequenzen



Knoten im AST

HPS WS 2002/03

Dr. Sabine Glesner

## Semantik eines kleinen Programms

- Semantik des Programms:

```

x := 2;
while x < 3 do
  x := x+1;
od;

```

- Zustand: Liste von Variablenbelegung
- initialer Zustand: leere Liste []
- vorgeführt an der Tafel

HPS WS 2002/03

Dr. Sabine Glesner