Deklarative Semantik

Bisher:

- Prolog als Programmiersprache.
- Operationale „Semantik“:
  Wie wird ein Programm ausgeführt?
- Welche Antworten werden berechnet?

Jetzt:

- Prolog als logischer Spezifikations-Formalismus.
- Deklarative Semantik:
  Was ist die Bedeutung eines Programms?
- Welche Antworten sind korrekt?

Dann:

- Prolog-Interpreter als automatischer Beweiser.
- Ist der Beweiser korrekt?
  Sind die berechneten Antworten auch korrekte Antworten?
- Ist der Beweiser vollständig?
  Werden alle korrekten Antworten berechnet?

Logische Formeln (1)

Alphabet A:

- logische Zeichen A_L: $\leftarrow$, $\wedge$, (,), ...
- nichtlogische Zeichen A_N: `member`, `length`, ...
- Variablen A_V: `X`, `DiesIstEineVariable`, ...
- Entspricht den Token von Prolog
- Formeln sind spezielle Zeichenketten über diesem Alphabet.

Signatur Σ:

- Menge der Prädikate $P \subseteq A_N \times \mathbb{N}$

  z.B. `member/2`, `append/3`, ...
- Menge der Funktionssymbole $F \subseteq A_N \times \mathbb{N}$

  z.B. `./2`, `datum/3`, `anton/0`, ...
- In Prolog ist die Signatur implizit durch Verwendung der Symbole gegeben. Ein Symbol kann als Prädikat und Funktionssymbol verwendet werden, und auch mit unterschiedlichen Stelligkeiten.
- Die Signatur enthält mindestens die vordefinierten Symbole.

Terme T_Σ:

- Variablen X (wobei $X \in A_V$)
- Konstanten c (wobei $(c, 0) \in F$)
- $f(t_1, \ldots, t_n)$ (wobei $(f, n) \in F$ und $t_i \in T_\Sigma$)

Logische Formeln (2)

Atomare Formel λ:

- p (wobei $(p, 0) \in P$)
- $p(t_1, \ldots, t_n)$ (wobei $(p, n) \in P$, $t_i \in T_\Sigma$)
- Da in Prolog jedes Prädikat auch als Funktionssymbol verwendet kann, ergibt sich dort die Unterscheidung zwischen atomaren Formeln und Termen nur aus dem Kontext (oder man verzichtet ganz darauf).

Allgemeine Formel φ:

- λ (atomare Formel)
- $\neg\varphi_0$ (Negation, „φ_0 gilt nicht")
  In Prolog wird „`not`" oder „`\+`" statt $\neg$ geschrieben (aber: s.u.).
- $\varphi_1 \wedge \varphi_2$ (Konjunktion, „φ_1 und φ_2")
  In Prolog wird „`,`" statt $\wedge$ geschrieben.
- $\varphi_1 \vee \varphi_2$ (Disjunktion, „φ_1 oder φ_2")
  In Prolog wird „`;`" statt $\vee$ geschrieben.
- $\varphi_1 \leftarrow \varphi_2$ (Implikation, „φ_1 wenn φ_2")
  In Prolog wird „`:-`" statt $\leftarrow$ geschrieben.
- $\varphi_1 \leftrightarrow \varphi_2$ (Äquivalenz, „φ_1 gdw. φ_2")
- $\forall X\ \varphi_0$ (Allaussage, „für alle X gilt φ_0")
- $\exists X\ \varphi_0$ (Existenzaussage, „es gibt X mit φ_0")
- $\perp$ (falsch)

Logische Formeln (3)

Prolog:

- Nur eingeschränkte Formeln: Hornklauseln.
- Dafür auch „außerlogische" Bestandteile.

  Prädikate mit Seiteneffekten, der Cut, etc.
- „Pure Prolog": Hornklauseln ohne Zusätze.

  Gewissermaßen der Schnitt der beiden Sprachen.

Horn-Klausel φ:

- λ (Faktum)
- $\lambda_0 \leftarrow \lambda_1 \wedge \cdots \wedge \lambda_m$ (Regel)

  In Prolog wird `:-` statt $\leftarrow$ und `,` statt $\wedge$ geschrieben.
- $\perp \leftarrow \lambda_1 \wedge \cdots \wedge \lambda_m$ (Anfrage)

  In Prolog: „`:-` λ_1`, ..., `λ_m". „$\perp \leftarrow$" enstpricht einer Negation (s.u.).
- Die λ_i sind atomare Formeln. Die Hornklauselmenge wird aufgeteilt in das Programm Φ (Fakten und Regeln) und jeweils eine Anfrage.

Quantifizierung von Variablen:

- Alle Variablen sind implizit allquantifiziert.
- Also: `p(X) :- q(X,Y)` bedeutet eigentlich $\forall X, Y\ p(X) \leftarrow q(X, Y)$.
- Dies ist wiederum äquivalent zu $\forall X, Y\ p(X) \vee \neg q(X, Y)$ und damit zu $\forall X\ p(X) \leftarrow \exists Y\ q(X, Y)$.

Logische Interpretationen (1)

Allgemeines:

- Eine Interpretation ordnet den Symbolen der Signatur eine Bedeutung zu.
- Nur bezüglich einer Interpretation kann man sagen, ob eine Formel wahr oder falsch ist.
- Der Programmierer denkt sich eine „intendierte Interpretation".
- Die Kunst besteht dann darin, nur solche Formeln aufzuschreiben, die in der intendierten Interpretation wahr sind, und andererseits
- die intendierte Interpretation vollständig zu beschreiben, d.h. daß im wesentlichen nur die intendierte Interpretation alle Formeln erfüllt.

Interpretation I:

- D: Individuenbereich (nichtleere Menge)
- $I[\![p, n]\!]$: Relation $\subseteq D^n$

  z.B. $I[\![\mathrm{vater}, 2]\!] = \{(\mathrm{anton}, \mathrm{christoph}), (\mathrm{berta}, \mathrm{christoph}), \ldots\}$.

  z.B. $I[\![<, 2]\!] = \{(0, 1), (0, 2), \ldots, (1, 2), (1, 3), \ldots\}$.
- $I[\![f, n]\!]$: Funktion $D^n \rightarrow D$

  z.B. $I[\![7, 0]\!] =$ „die Zahl 7".

  z.B. $I[\![\,.\,, 2]\!](1, [2, 3]) = [1, 2, 3]$. ($I[\![\,.\,, 2]\!]$ ist Funktion f, und $f(1, [2, 3]) = [1, 2, 3]$.)

  z.B. $I[\![+, 2]\!](1, 3) = 4$ (Nicht in Prolog! s.u.)

Logische Interpretationen (2)

Einschränkungen in Prolog:

- Vordefinierte Prädikate haben feste Bedeutung.
  Z.B. $<$, `is`, ...
- Die Funktionssymbole werden „frei" interpretiert (als Term-/Record-Konstruktoren).

Clark's Gleichheitstheorie:

- $c \neq d$.
  Für alle Paare verschiedener Konstanten c, d.
- $f(X_1, \ldots, X_n) \neq g(Y_1, \ldots, Y_m)$.
  Für alle Paare verschiedener Funktionssymbole f/n, g/m.
- $f(X_1, \ldots, X_n) \neq c$.
- $(X_1 \neq Y_1) \vee \cdots \vee (X_n \neq Y_n)$
  $\quad \rightarrow \; f(X_1, \ldots, X_n) \neq f(Y_1, \ldots, Y_n)$.
- $t \neq X$.
  Für Variablen X und Terme t, die X echt enthalten, z.B. $f(f(X)) \neq X$.
- Daneben gibt es noch die folgenden Axiome, die erzwingen, daß $=$ eine mit den Funktionen/Prädikaten verträgliche Äquivalenzrelation ist:
  $X = X$.
  $(X_1 = Y_1) \wedge \cdots \wedge (X_n = Y_n) \rightarrow f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n)$.
  $(X_1 = Y_1) \wedge \cdots \wedge (X_n = Y_n) \wedge p(X_1, \ldots, X_n) \rightarrow p(Y_1, \ldots, Y_n)$.
  (Auch für $p \equiv =$, oder Symmetrie und Transitivität einzeln fordern.)

Logische Interpretationen (3)

Beispiel:

- $I[\![+, 2]\!](4, 5) = 9$ verletzt Gleichheitstheorie.
- In Prolog: $I[\![+, 2]\!](4, 5) =$ „der Term $+(4, 5)$“.

  Anstelle von Termen (Zeichenketten) kann man sich auch die entsprechenden Baumstrukturen / verzeigerte Records vorstellen.

Herbrand-Interpretation:

- Noch spezieller als Clark's Gleichheitstheorie.
- Betrache die Signatur Σ der im Programm vorkommenden Symbole.

  Falls das Programm keine Konstanten enthält, wird eine hinzugefügt.
- D ist die Menge der variablenfreien Terme über Σ
- Funktionssymbole werden als Termkonstruktoren interpretiert: $I[\![f, n]\!](d_1, \ldots, d_n)$= „$f(d_1, \ldots, d_n)$“.
- Also: Eine Herbrand-Interpretation ist durch die Werte der Prädikate vollständig definiert.

Variablenbelegung α:

- Abbildung α: $A_V \rightarrow D$

  (ordnet jeder Variablen einen Wert zu, z.B. $\alpha(\mathrm{X}) = \mathrm{anton}$)

Modell-Beziehung (1)

Auswertung von Termen:

- $(I, \alpha)[\![c]\!] := I[\![c, 0]\!]$
- $(I, \alpha)[\![X]\!] := \alpha(X)$
- $(I, \alpha)[\![f(t_1, \ldots, t_n)]\!] := I[\![f, n]\!](d_1, \ldots, d_n)$,
  wobei $d_i := (I, \alpha)[\![t_i]\!]$.

Gültigkeit von Formeln in (I, α):

- $(I, \alpha) \models p(t_1, \ldots, t_n) \;\; :\Longleftrightarrow$
  $(d_1, \ldots, d_n) \in I[\![p, n]\!]$, wobei $d_i := (I, \alpha)[\![t_i]\!]$.
- $(I, \alpha) \models \neg\varphi_0 \;\; :\Longleftrightarrow$
  es gilt nicht, daß $(I, \alpha) \models \varphi_0$.
- $(I, \alpha) \models \varphi_1 \wedge \varphi_2 \;\; :\Longleftrightarrow$
  sowohl $(I, \alpha) \models \varphi_1$, als auch $(I, \alpha) \models \varphi_2$.
- $(I, \alpha) \models \varphi_1 \vee \varphi_2 \;\; :\Longleftrightarrow$
  $(I, \alpha) \models \varphi_1$ oder $(I, \alpha) \models \varphi_2$ (oder beides).
- $(I, \alpha) \models \varphi_1 \leftarrow \varphi_2 \;\; :\Longleftrightarrow$
  wenn $(I, \alpha) \models \varphi_2$, so $(I, \alpha) \models \varphi_1$.
- $(I, \alpha) \models \varphi_1 \leftrightarrow \varphi_2 \;\; :\Longleftrightarrow$
  $(I, \alpha) \models \varphi_1 \iff (I, \alpha) \models \varphi_2$.

Modell-Beziehung (2)

Gültigkeit von Formeln in (I, α), Forts.:

- $(I, \alpha) \models \forall X\ \varphi_0\ :\Longleftrightarrow$

  für alle $d \in D$ gilt: $(I, \alpha_{X:=d}) \models \varphi_0$.

  Dabei ist $\alpha_{X:=d}(X) = d$ und $\alpha_{X:=d}(Y) = \alpha(Y)$ für $Y \neq X$.

- $(I, \alpha) \models \exists X\ \varphi_0\ :\Longleftrightarrow$

  für mindestens ein $d \in D$ gilt: $(I, \alpha_{X:=d}) \models \varphi_0$.

- $(I, \alpha) \not\models \bot$. D.h.: Es gilt nicht, daß $(I, \alpha) \models \bot$.

Modell:

- $I \models \varphi\ :\Longleftrightarrow$ für alle α gilt: $(I, \alpha) \models \varphi$.

  Die Variablen sind also implizit all-quantifiziert.

- $I \models \Phi\ :\Longleftrightarrow$ für alle $\varphi \in \Phi$ gilt: $I \models \varphi$.

  Man sagt dann auch: I ist ein Modell von Φ.

Beispiel:

- Gegebenes Programm:

```
p(a).
q(X) :- p(X).
q(b).
```

- Intendiertes Modell (Herbrand-Interpretation):

  $D := \{a, b\}$.

  $I[\![p, 1]\!] := \{(a)\}$.

  $I[\![q, 1]\!] := \{(a),\ (b)\}$.

Kausale Modelle

Problem:

- In einem Modell kann „zuviel“ gelten.
- Auch folgende Herbrand-Interpretation ist Modell des letzten Beispiels:
  $J[\![p, 1]\!] := \{(a),\ (b)\}$.
  $J[\![q, 1]\!] := \{(a),\ (b)\}$.
- Allgemein: Eine Interpretation, in der alle Prädikate für beliebige Argumente wahr sind, erfüllt jedes logische Programm (Fakten, Regeln).

Kausales Modell:

Für jedes $(d_1, \ldots, d_n) \in I[\![p, n]\!]$ gibt es

eine Regel $p(t_1, \ldots, t_n) \leftarrow \lambda_1 \wedge \cdots \wedge \lambda_m$

und eine Variablenbelegung α, so daß

- $(I, \alpha)[\![t_i]\!] = d_i$ für $i = 1, \ldots, n$, und
- $(I, \alpha) \models \lambda_j$ für $j = 1, \ldots, m$.

Satz:

- Sei Φ ein logisches Programm. Der Schnitt aller Herbrandmodelle von Φ ist ein Modell von Φ.

  Dieser Schnitt wird das minimale Herbrandmodell genannt.
- Das minimale Herbrandmodell ist kausal.

  Jedes logische Programm hat also ein kausales Modell.

Clark's Vervollständigung

Idee:

- Programmierer schreibt nur positives Wissen auf.

  Aber dies ist vollständig. Was nicht ableitbar ist, gilt auch nicht.

- Wenn er $\leftarrow$ schreibt, meint er eigentlich $\leftrightarrow$.

Vervollständigung $comp(\Phi)$:

- Normalisiere Regelköpfe zu $p(X_1, \ldots, X_n)$.

  Z.B. gegeben: $p(a)$ und $p(f(Y)) \leftarrow q(Y)$.

  Überführe dies in: $p(X) \leftarrow X = a$ und $p(X) \leftarrow X = f(Y) \wedge q(Y)$.

- Binde Variablen, die nur im Rumpf auftreten, durch $\exists$.

  Im Beispiel wird die zweite Regel zu: $p(X) \leftarrow \exists Y\ (X = f(Y) \wedge q(Y))$.

- Verknüpfe Regelrümpfe über ein Prädikat mit $\vee$.

  Jetzt also $p(X) \leftarrow X = a \vee \exists Y\ (X = f(Y) \wedge q(Y))$.

  Beachte: Dies ist noch äquivalent zu den ursprünglichen Regeln!

- Mache $\leftarrow$ zu $\leftrightarrow$.

  Also Ergebnis: $p(X) \leftrightarrow X = a \vee \exists Y\ (X = f(Y) \wedge q(Y))$.

Satz:

I ist kausales Modell eines logischen Programms Φ gdw. $I \models comp(\Phi)$.

Korrekte Antworten

Logische Folgerung $\Phi \vdash \psi$:

- Jedes Modell I von Φ ist auch Modell von ψ.

  Alternativ: Für alle I gilt: $I \models \Phi \implies I \models \psi$.

Lemma:

- $\Phi \vdash \psi$ gdw. $\Phi \cup \{\exists\,(\neg\psi)\}$ hat kein Modell.

  $\exists\,(\neg\psi)$: alle vorkommenden Variablen werden $\exists$-quantifiziert.

- Prinzip des Widerspruchsbeweises.

  Daher also die Schreibweise `:- ` λ_1`, ...,`λ_n für Anfragen.

Korrekte Antwort:

- Substitution θ für Variablen der Anfrage ψ, so daß $comp(\Phi) \vdash \psi\theta$.

  D.h. jedes kausale Modell von Φ ist Modell von $\psi\theta$.

- Bisher Anfragen nur Konjunktionen von Atomen. Dann ist dies äquivalent zu $\Phi \vdash \psi\theta$.

- Die identische Substitution wird als „yes“ ausgedruckt. D.h. die Antwort „yes“ ist korrekt gdw. $comp(\Phi) \vdash \psi$.

Antwort „no“:

- ist korrekt gdw. $comp(\Phi) \vdash \forall\,(\neg\psi)$.

Fixpunkt-Semantik

Vorbemerkungen:

- Sei feste Grundinterpretation gegeben.

  Eine Grundinterpretation legt den Individuenbereich und die Interpretation der Funktionssymbole fest (aber nicht die der Prädikate).

- Identifiziere I mit Menge elementarer Fakten:

  $\{p(d_1, \ldots, d_n) \mid (d_1, \ldots, d_n) \in I[\![p, n]\!]\}$.

Direkte Konsequenzen:

$p(d_1, \ldots, d_n)$ direkte Konsequenz aus Φ, I gdw.

- es gibt $p(t_1, \ldots, t_n) \leftarrow \lambda_1, \ldots, \lambda_m$ und α mit
- $(I, \alpha)[\![t_i]\!] = d_i \quad (i = 1, \ldots, n)$,
- $(I, \alpha) \models \lambda_j \quad (j = 1, \ldots, m)$.

$T_\Phi(I) := \{p(d_1, \ldots, d_n) \mid \text{direkte Konsequenz}\}$.

Satz:

- I ist Modell von Φ gdw. $T_\Phi(I) \subset I$.
- I ist kausales Modell von Φ gdw. $T_\Phi(I) = I$.
- I ist minimales Modell von Φ gdw.

  I ist kleinster Fixpunkt von T_Φ.

Nachtrag zu Clark's Gleichheitstheorie

Semantische Formulierung:

- Die Funktionen $I[\![f,n]\!]$ sind injektiv.
- Die Wertebereiche der $I[\![f,n]\!]$ sind disjunkt.

  Konstanten werden hier als 0-stellige Funktionen gesehen.
- Es gibt keine Zyklen der Form:

  $d_i := I[\![f_i, n_i]\!](\ldots, d_{i-1}, \ldots), \quad (i = 1, \ldots, m)$

  mit $d_m = d_0$.

Lemma:

- Sind $p(t_1, \ldots, t_n)$ und $p(u_1, \ldots, u_n)$ unifizierbar mit allg. Unifikator $\{X_1/v_1, \ldots, X_m/v_m\}$, so gilt:

  $$\begin{aligned} CET \models \forall(\; & (t_1 = u_1 \wedge \cdots \wedge t_n = u_n) \\ & \leftrightarrow \; (X_1 = v_1 \wedge \cdots \wedge X_m = v_m)). \end{aligned}$$
- Andernfalls (nicht unifizierbar):

  $CET \models \forall((t_1 = u_1 \wedge \cdots \wedge t_n = u_n) \leftrightarrow \bot)$.

  D.h. $CET \models \neg\exists(t_1 = u_1 \wedge \cdots \wedge t_n = u_n)$.

Warum nicht allgemeine Modelle:

- Programmierer will häufig Aufzählungstypen/Record-Konstruktoren, wäre aufwendig explizit zu spezifizieren.
- Enge Beziehung zur Unifikation.

Fixpunkt-Semantik (2)

Bemerkung (für Mathematiker):

- Die Interpretationen über einer Grundinterpretation bilden bezüglich „$\subseteq$“ einen vollständigen Verband, eben den Teilmengenverband.
- Die Abbildung T_Φ ist monoton, d.h. $I_1 \subseteq I_2 \implies T_\Phi(I_1) \subseteq T_\Phi(I_2)$.
- T_Φ ist sogar stetig, d.h. man kann die Bildung der kleinsten oberen Schranke mit der Funktionsanwendung vertauschen (für Mengen, die obere Schranken für jede endliche Teilmenge enthalten).
- Satz der Verbandstheorie: Unter diesen Voraussetzungen kann man den kleinsten Fixpunkt durch ω-fache (unendlich häufige) Iteration von T_Φ beginnend mit dem kleinsten Element $\emptyset$ bestimmen.

„Berechnung“ des kleinsten Fixpunktes:

- $T_\Phi^0 := \emptyset$.
- $T_\Phi^{i+1} := T_\Phi(T_\Phi^i)$.
- $T_\Phi^\omega := \cup_{i=0}^\infty T_\Phi^i$.

  Die Vereinigung ist nicht eigentlich nötig, weil $T_\Phi^i \subseteq T_\Phi^{i+1}$.

Satz:

- T_Φ^ω ist gerade das minimale Modell von Φ.

  bezüglich der gegebenen Grundinterpretation.
- Sollten sich schon nach endlich häufiger Iteration keine neuen Fakten mehr ableiten lassen, d.h. $T_\Phi^{n+1} = T_\Phi^n$, so ist $T_\Phi^\omega = T_\Phi^n$.

Aufgaben

Aufgabe:

- Berechnen Sie das kleinste Herbrandmodell durch iterierte Fixpunktbildung:
  $p(a, b).$
  $p(b, c).$
  $q(X, Y) :- p(X, Y).$
  $q(X, Z) :- p(X, Y), q(Y, Z).$
- Ist Gleichheitstheorie erfüllt?
  $D := \mathbb{N}$
  $I[\![a, 0]\!] := 0, \quad I[\![b, 0]\!] := 1, \quad I[\![c, 0]\!] := 2$
- Ist folgende Interpretation ein Modell?
  $I[\![p, 2]\!] := \{(0, 1),\ (1, 2)\}.$
  $I[\![q, 2]\!] := \{(i, i + 1) \mid i \in \mathbb{N}\}.$
- Ist sie kausal?

Aufgabe:

- Bestimmen Sie T_Φ:
  $a\_list([\,]).$
  $a\_list([a|X]) :- a\_list(X).$
- Wie sieht das kleinste Herbrandmodell aus?