Grundlagen der Programmierung (Vorlesung 9)

Ralf Möller, FH-Wedel

- Vorige Vorlesung
  - Semantik der Prädikatenlogik erster Stufe
  - Entscheidungsprobleme (Teil 1)
- Inhalt dieser Vorlesung
  - Entscheidungsprobleme (Teil 2)
  - Spezifikation der Aufgabe von Algorithmen
- Lernziele
  - Grundlagen der systematischen Programmentwicklung

Folgerung und Äquivalenz

Eine Formel G heißt eine *Folgerung* der Formeln $F_1, \ldots, F_k$ falls für jede Struktur, die sowohl zu $F_1, \ldots, F_k$ als auch zu G passend ist, gilt:

> Wenn $\mathcal{A}$ Modell von $\{F_1, \ldots, F_k\}$ ist, dann ist $\mathcal{A}$ auch Modell von G.

Wir schreiben $F_1, \ldots, F_k \models G$, falls G eine Folgerung von $F_1, \ldots, F_k$ ist.

Zwei Formeln F unf G heißen (*semantisch*) *äquivalent*, falls für alle Strukturen $\mathcal{A}$, die sowohl für F als auch für G passend sind, gilt $\mathcal{A}(F) = \mathcal{A}(G)$. Hierfür schreiben wir $F \equiv G$.

Aufgabe

1. $\exists y \forall x P(x,y)$

2. $\forall x \exists y P(x,y)$

	J	N
1. $\models$ 2.		
2. $\models$ 1.		

Äquivalenzen

Satz:

Seien F und G beliebige Formeln:

1. $\neg\forall xF \equiv \exists x\neg F$

   $\neg\exists xF \equiv \forall x\neg F$

2. Falls x in G nicht frei vorkommt, gilt:

   $(\forall xF \wedge G) \equiv \forall x(F \wedge G)$

   $(\forall xF \vee G) \equiv \forall x(F \vee G)$

   $(\exists xF \wedge G) \equiv \exists x(F \wedge G)$

   $(\exists xF \vee G) \equiv \exists x(F \vee G)$

3. $(\forall xF \wedge \forall xG) \equiv \forall x(F \wedge G)$

   $(\exists xF \vee \exists xG) \equiv \exists x(F \vee G)$

4. $\forall x\forall yF \equiv \forall y\forall xF$

   $\exists x\exists yF \equiv \exists y\exists xF$

Aufgabe

	J	N
$\forall x \forall y F \equiv \forall y \forall x F$		
$\forall x \exists y F \equiv \exists x \forall y F$		
$\exists x \exists y F \equiv \exists y \exists x F$		
$\forall x F \vee \forall x G \equiv \forall x (F \vee G)$		
$\forall x F \wedge \forall x G \equiv \forall x (F \wedge G)$		
$\exists x F \vee \exists x G \equiv \exists x (F \vee G)$		
$\exists x F \wedge \exists x G \equiv \exists x (F \wedge G)$		

Beispiel

- Es seien Animal, Vegetarian, Sheep, Cow, und Mad_cow einstellige Prädikatensymbole. Die Symbole EATS und PART_OF seien zweistellige Prädikatensymbole, und es seien x, y, z Variablen. Wir betrachten folgende Menge T von Formeln:
- T := {□x (Sheep(x) □
  (Animal(x) □ Vegetarian(x))) ,
  □x (Cow(x) □
  (Animal(x) □ Vegetarian(x))) ,
  □x (Vegetarian(x) □
  (□□y (EATS(x,y) □ Animal(y)) □
  □y (EATS(x,y) □
  (□□z (PART_OF(y,z) □ Animal(z))))) }

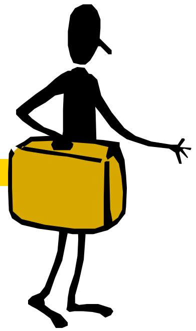
Beispiel (2)

- Annahme: Formeln seien Grundlage für die Spezifikation von Programmen für ein großes Softwareprojekt für das Landwirtschaftsministerium dar.
- Frage: Welche Kühe (eines bestimmten Bestandes, der als Parameter eingeht) sind beim Verzehr gesundheitsgefährdend?
- Hierzu wird folgende Formel verwendet:
  f := □x (Mad_cow(x) □
  (Cow(x) □ □y (EATS (x, y) □
  □z PART_OF(y, z) □ Sheep(z))))

Beispiel (3)

- Zu erstellen: Programm, das die Menge M aller x berechnen soll, für die Mad_cow(x) gilt
- Nach eingehender Beratung mit Ihren Mitarbeitern kommen Sie zu dem Schluß, daß Sie das Projekt, obwohl es lukrativ sein mag, aus Gewissensgründen nicht annehmen werden, da die Menge M immer leer sein wird.
- Sie zeigen, daß die Formel $\exists x$ Mad_cow(x) unerfüllbar bzgl. $T \cup \{f\}$ ist

Zusammenfassung, Kernpunkte



- Prädikatenlogik
- Folgerbarkeit und Äquivalenz
- Äquivalente Transformation von Formeln
- Grundlagen der logischen Modellierung

Was kommt beim nächsten Mal?

- Logik und die systematische Entwicklung von Programmen
- Spezifikation und Programmverifikation