Grundlagen der Programmierung

Vorlesung 4 vom 04.11.2004

Sebastian Iwanowski
FH Wedel

Grundlagen der Programmierung

1. Einführung
   - Grundlegende Eigenschaften von Algorithmen und Programmen
2. Logik
   - ➔ Aussagenlogik
   - Prädikatenlogik
3. Programmentwicklung und –verifikation
   - Grundlagen der Programmverifikation
   - Verbundanweisungen
   - Verzweigungen
   - Schleifen
   - Modularisierung
   - Rekursion
4. Entwurf und Analyse von Algorithmen
   - Klassifikation von Algorithmen
   - Programmierung von Algorithmen
   - Bewertung von Algorithmen

Normalformen

Warum wollen wir Formeln in KNF bringen ?

- **Übersichtlichere Auswertung beim Erfüllbarkeitstest:**

  Eine Formel in KNF ist erfüllbar.
  ⇔ Eine Belegung enthält für jede Klausel wenigstens einen wahren Literal.

  Mögliche Belegungsstrategie:

  - Gehe die Klauseln nacheinander durch:
  - Belege genau einen noch nicht festgelegten Literal mit **w**

    (dadurch werden gleiche Literale oder deren Negationen in anderen Klauseln festgelegt)
  - Wenn es keine Möglichkeit mehr gibt, springe zurück zur vorigen Klausel und nimm eine andere Belegung

Vorsicht vor Illusionen:

Im schlechtesten Fall bringt das keinen Zeitgewinn verglichen mit purem Ausprobieren !

Normalformen

Warum wollen wir Formeln in KNF bringen ?

- **Kompakte Darstellbarkeit im Computer:**

  Stelle Klauseln als Mengen von Literalen dar:

  {p,¬q, r} entspricht (p ∨ ¬q ∨ r) **als Klausel**

  Stelle Formeln als Mengen von Klauseln dar:

  {{p,¬q, r} , {¬p,¬q,¬r} , {p, q}} entspricht (p ∨ ¬q ∨ r) ∧ (¬p ∨ ¬q ∨ ¬r) ∧ (p ∨ q)

  Das funktioniert sogar für die Spezialfälle:

  {{p,¬q, r}} entspricht (p ∨ ¬q ∨ r) **als Formel**

  {{p},{¬q},{r}} entspricht (p ∧ ¬q ∧ r) **als Formel**

  {} entspricht ⊤ **als Formel**

  {{}} entspricht ⊥ **als Formel**

Warnung:

Was dem Computer glasklar ist, kann für den Menschen höchst verwirrend sein:

- Das Trennzeichen (,) in **inneren** Klammern (Klauseln) entspricht einer Disjunktion (∨)
- Das Trennzeichen (,) in **äußeren** Klammern (Formeln) entspricht einer Konjunktion (∧)

Normalformen

Satz: **Jede aussagenlogische Formel lässt sich durch endlich viele äquivalente Umformungen in KNF bringen.**

Algorithmus zum Umwandeln einer Formel in die KNF:

1. Eliminiere alle Operatoren der Form ↔ und → mit den Ersetzungsregeln durch ∧ und ∨ !

2. Ziehe alle Negationszeichen vor Klammern in die Klammern hinein mit den deMorganschen Regeln !

3. Wende die Distributivgesetze so lange an, bis auf oberster Ebene nur noch Konjunktionen und darunter Disjunktionen sind !

Das funktioniert immer !

Andere Normalformen

Definition:

Eine aussagenlogische Formel ist in **disjunktiver Normalform (DNF)**, wenn sie als Disjunktion von Konjunktionen aus Aussagen oder Negationen von Aussagen dargestellt ist.

Jetzt fällt das Mitdenken schon leichter:

Was ist in DNF ein **Literal** ?

Was ist in DNF eine **Klausel** ?

Was ist in DNF eine **Formel** ?

Beispiele ?

Warum wollen wir Formeln in DNF bringen ?

wollen wir nicht: Eine Normalform reicht uns aus !

Zusammenfassung: Aussagenlogische Formeln

Formeln haben <u>logische Eigenschaften</u> (Semantik):

- **Erfüllbarkeit**
- **Widerspruch** — äquivalent zu $\bot$ — komplementär zueinander
- **Tautologie** — äquivalent zu $\top$ — komplementär zueinander
- ***Falsifizierbarkeit***

Formeln haben <u>Darstellungseigenschaften</u> (Syntax):

- **Normalformen (KNF oder DNF)**
- **Mengendarstellung der KNF**

Die eben genannten Darstellungseigenschaften sind für <u>jede</u> Formel durch Äquivalenz erzielbar.

Es gibt auch Darstellungseigenschaften, die nur für bestimmte Formeln möglich sind.

<u>Bsp.:</u> KNF mit Klauseln aus genau drei Literalen

Grundlagen der Programmierung

1. Einführung
   - Grundlegende Eigenschaften von Algorithmen und Programmen
2. Logik
   - Aussagenlogik
   - ➔ Prädikatenlogik
3. Programmentwicklung und –verifikation
   - Grundlagen der Programmverifikation
   - Verbundanweisungen
   - Verzweigungen
   - Schleifen
   - Modularisierung
   - Rekursion
4. Entwurf und Analyse von Algorithmen
   - Klassifikation von Algorithmen
   - Programmierung von Algorithmen
   - Bewertung von Algorithmen

Grenzen der Aussagenlogik

Gegeben

- Alle, die die Vorlesung besuchen, bestehen die Klausur (A)
- Susi besucht die Vorlesung (B1)
- Bernd besucht die Vorlesung (B2)
- Linda besucht nicht die Vorlesung (B3)
- Alex hat die Klausur nicht bestanden (B4)

Gewünschte Folgerungen

- Susi besteht die Klausur (C1)
- Bernd besteht die Klausur (C2)
- Linda ? Alex ?

Aussagenlogische Formeln:

- $A \wedge B1 \rightarrow C1$
- $A \wedge B2 \rightarrow C2$
- ...

Gewünschte Formel:

- besuchtVorlesung (x) → bestehtKlausur (x)

In der Aussagenlogik:

- **keine Variablen**
- **keine variablenabhängigen Aussagen**

Grenzen der Aussagenlogik

Gegeben

- Die Klausurnote eines Studenten, der alle Übungen gelöst hat, ist 1 oder 2 (A)
- Susi hat alle Übungen gelöst (B1)
- Bernd hat alle Übungen gelöst (B2)
- Linda hat nicht alle Übungen gelöst (B3)
- Alex hat eine 5 in der Klausur (B4)

Gewünschte Formel:

- hatAlleÜbungenGelöst (x) → (Klausurnote (x) ≤ 2)

In der Aussagenlogik:

- **keine Variablen**
- **keine variablenabhängigen Aussagen**
- **keine Funktionen über Variablen**

Grenzen der Aussagenlogik

Gegeben

- Eine Vorlesung, die nur von Studenten oder nur von Studentinnen besucht wird, ist langweilig (A)
- Susi ist eine Studentin (B1)
- Bernd ist ein Student (B2)
- Susi besucht GdP (B3)
- GdP ist nicht langweilig (B5)

Gewünschte Formel:

- ((besuchtVorlesung (v, s) → weiblich (s)) v (besuchtVorlesung (v, s) → weiblich (s))) → langweilig (v)

- Wie drückt man aus, dass eine Vorlesung nur von Studenten oder nur von Studentinnen besucht wird ?
- Wie drückt man aus, dass es für jede Vorlesung überhaupt Studenten oder Studentinnen geben sollte, die sie besuchen ?

In der Aussagenlogik:

- **keine Variablen**
- **keine variablenabhängigen Aussagen**
- **keine Funktionen über Variablen**
- **keine Operatoren „für alle“ oder „es gibt“**

Von der Aussagenlogik zur Prädikatenlogik

In der Aussagenlogik:

- keine Variablen
- keine variablenabhängigen Aussagen
- keine Funktionen
- keine Operatoren „für alle“ oder „es gibt“

In der Prädikatenlogik:

- Variable
- Prädikate
- Funktionen
- Quantoren

Prädikatenlogik

- **Variable**

  In eine Variable dürfen beliebige Elemente eingesetzt werden.

  Könnten nicht die Literale in der Aussagenlogik als Variable aufgefasst werden ?

  Was ist bei Literalen in der Aussagenlogik anders ?

- **Prädikate**

  Ein Prädikat ist eine Aussage, die von anderen Werten abhängt.

  Die Anzahl der Werte, von denen ein Prädikat abhängt, ist für jedes Prädikat eine beliebige, aber feste Zahl.

  Ein Prädikat, das von k Werten abhängt, heißt k-stellig.
  Kurzform: $P(x_1, x_2, \ldots, x_k)$

Prädikatenlogik

- **Beispiele zu Prädikaten:**

1. Zwei Personen sind miteinander verheiratet

2. Eine Person liebt eine andere

3. Eine Person hasst eine andere

4. Eine Person besucht die Vorlesung GdP

5. Eine Person besucht eine beliebige Vorlesung

6. Ein Ort liegt zwischen zwei anderen

Prädikatenlogik

- **Funktionen**

  Eine Funktion ist eine Zuordnung, die einer Menge von Werten einen neuen Wert eindeutig zuordnet.
  Kurzform: $f(x_1, x_2, ..., x_n) = y$

  Eine Funktion, die von k Werten abhängt, heißt k-stellig.

  Bsp.: Drücke folgenden Sachverhalt mit prädikatenlogischen Hilfsmitteln aus:

  $(2 < x < 4) \wedge (0 < y < 6) \wedge (x + y > 7) \wedge (x \cdot y < 10)$

Prädikatenlogik

- **Quantoren**

  Der Existenzquantor ∃x (. . .) drückt aus, dass es einen Wert für x gibt, der den dahinter stehenden Ausdruck zu einer wahren Aussage macht.

  Der Allquantor ∀x (. . .) drückt aus, dass jeder Wert für x den dahinter stehenden Ausdruck zu einer wahren Aussage macht.

  Die Definitionsbereiche für die Variablen dürfen eingeschränkt werden:

  Für den Existenzquantor ist das eine Verschärfung,
  für den Allquantor eine Abschwächung der Aussage.

Bsp.:

$$\exists\, x \in \mathbb{R}\ \exists\, y \in \mathbb{R}\ ((2 < x < 4) \wedge (0 < y < 6) \wedge (x + y > 7) \wedge (x \cdot y < 10))$$

$$\forall\, x \in \mathbb{R}\ (x \cdot x \geq 0)$$

Prädikatenlogische Formeln

- **Eine prädikatenlogische Formel 1. Stufe ist eine Verknüpfung von endlich vielen Variablen, Funktionen und Prädikaten mit aussagenlogischen Operatoren oder Quantoren, die sich nur auf Variable beziehen.**

Bsp.: $\forall x\,(\,R(y, z) \wedge \exists y\,(\neg P(y, x) \vee R(y, z))\,)$

Grüne Vorkommen von y und z sind **frei**.
Rote Vorkommen von x, y und z sind **gebunden**.

Prädikatenlogische Formeln

- **Eine Belegung einer Formel ist eine Zuweisung von *Werten aus festgelegten Definitionsbereichen an die freien Variablen* derart, dass dieselben Variablen immer denselben Wert erhalten.**

- **Eine Formel heißt erfüllbar, wenn es eine Belegung gibt derart, dass die Formel wahr ist.**

  - Das Erfüllbarkeitsproblem ist in der Prädikatenlogik nicht entscheidbar, d.h. kein Algorithmus kann jemals in der Lage sein, von jeder Formel zu entscheiden, ob sie erfüllbar ist oder nicht.

Das allgemeine Problem ist unlösbar !

Beim nächsten Mal:

Prädikatenlogik, 2. Teil