Übung zur Vorlesung

Automatisierte Logik und Programmierung II

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Sommersemester 2006

Blatt 3 — Abgabetermin: —

Aufgabe 3.1 (Aufbau einer formalen Theorie)

Graphen und Bäume sind wichtige Datenstrukturen für eine große Menge von Anwendungsproblemen. Formalisieren Sie eine Theorie endlicher Graphen und Bäume derart, daß sich die folgenden Probleme darin beschreiben lassen.

- Das Cliquen-Problem: Gegeben ein Graph $G = (V, E)$ der Größe n und eine Zahl $k \leq |V|$. Gibt es in G eine Clique der Mindestgröße k?
- Das Independent Set: Gegeben ein Graph $G = (V, E)$ der Größe n und eine Zahl $k \leq |V|$. Gibt es in G eine unabhängige Knotenmenge der Mindestgröße k?
- Das Vertex Cover Problem: Gegeben ein Graph $G = (V, E)$ der Größe n und eine Zahl $k \leq |V|$. Gibt es in G eine Knotenüberdeckung der Maximalgöße k?
- Das Travelling Salesman Problem: Gegeben ein vollständiger gewichteter Graph (G, g). Gibt es in G einen Zyklus dessen Gesamtgewicht unter B liegt?
- Das MWST Problem. Gegeben ein gewichteter Graph (G, g). Bestimme einen minimal spannenden Baum von G.

3.1–a Formalisieren Sie zunächst die wichtigsten Begriffe der Theorie endlicher Graphen und Bäume in der Typentheorie (Beispiele sind im Anhang genannt). Identifizieren Sie dabei Konzepte aus der Theorie endlicher Mengen und Listen, die für eine Formalisierung erforderlich wären.

3.1–b Formulieren Sie Lemmata, welche Zusammenhänge zwischen verschiedenen Begriffen beschreiben, z.B. den Zusammenhang zwischen Cliquen und unabhängige Knotenmengen.

3.1–c Formalisieren Sie die obengenannten Probleme als "Spezifikationstheoreme", mit denen im Sinne des Prinzips "Beweise als Programme" gezeigt würde, daß für jede Problemstellung eine Lösung konstruierbar ist. Wie müssen dabei Entscheidungsprobleme spezifiziert werden?

Diese Aufgabe ließe sich leicht zu einem Projekt/Studienarbeit erweitern

Aufgabe 3.2 (Anwendungsbezogene Erweiterung formaler Theorien)

Das n-Dame Problem: Gegeben ein Schachbrett mit nxn Feldern und n Dame Figuren. Eine Dame kann alle Figuren schlagen, die sich auf derselben waagerechten oder senkrechten Linie befinden sowie alle Figuren, die sich auf der von ihr ausgehenden Diagonale befinden. Gesucht sind *alle* möglichen Plazierungen der n Damen auf dem Schachbrett, so daß keine Dame eine andere schlagen kann.

Formalisieren Sie das n-Dame Problem als Spezifikationstheorem. Stellen Sie eventuell notwendige Definitionen für neue Begriffe auf und beschreiben Sie die Gesetze dieser neuen Konzepte.

Beispiele graphentheoretischer Definitionen

- Ein (ungerichteter) *Graph* ist ein Paar $G = (V, E)$, wobei V endliche Menge und $E \subseteq \{ \{v, v'\} \mid v, v' \in V \wedge v \neq v' \}$.
  Ein Graph ist darstellbar als Liste $v_1, ..., v_n, \{v_{i_1}, v'_{i_1}\}, ..., \{v_{i_m}, v'_{i_m}\}$.
- Ein *gerichteter Graph* ist ein Paar $G = (V, E)$, wobei V endliche Menge und $E \subseteq V \times V$.
- Ein *gewichteter Graph* ist ein Graph $G = (V, E)$ mit einer Gewichtungsfunktion $g : E \to \mathbb{N}$.
- Ein Graph $H = (V_H, E_H)$ ist genau dann *Subgraph* des Graphen $G = (V, E)$ ($H \sqsubseteq G$), wenn alle Ecken und Kanten von H auch Ecken bzw. Kanten in G sind:
  $$(V_H, E_H) \sqsubseteq (V, E) :\Leftrightarrow V_H \subseteq V \wedge E_H \subseteq E$$
- $H = (V_H, E_H)$ ist *isomorph* zu $G = (V, E)$ (kurz: $H \cong G$), wenn die Graphen durch Umbenennung (bijektive Abbildung $h : V_2 \to V$) ineinander überführt werden können:
  $(V_H, E_H) \cong (V, E) :\Leftrightarrow \exists h : V \to V_H.(h \text{ bijektiv } \wedge\ E_H = \{\{h(u), h(v)\} \mid \{u, v\} \in E\})$
- Die *Größe* $|G|$ eines Graphen $G = (V, E)$ ist die Anzahl $|E|$ seiner Kanten.
- Der *Komplementärgraph* des Graphen $G = (V, E)$ ist der Graph $G^c = (V, E^c)$ mit $E^c = \{ \{v, v'\} \mid v, v' \in V \} - E$.
- Eine *Clique* der Größe k im Graphen $G = (V, E)$ ist eine vollständig verbundene Knotenmenge $V' \subseteq V$ mit $|V| = k$.
  Dabei heißt V' *vollständig verbunden*, wenn gilt: $\forall v, v' \in V'.\, v \neq v' \Rightarrow \{v, v'\} \in E$
- Eine *unabhängige Knotenmenge* der Größe k im Graphen $G = (V, E)$ ist eine Knotenmenge $V' \subseteq V$ mit $|V| = k$ mit der Eigenschaft $\forall v, v' \in V'.\, v \neq v' \Rightarrow \{v, v'\} \notin E$
- Eine *Knotenüberdeckung* (Vertex cover) des Graphen $G = (V, E)$ ist eine Knotenmenge $V' \subseteq V$ mit der Eigenschaft $\forall \{v, v'\} \in E.\, v \in V' \vee v' \in V'$
- Ein *Zyklus* (*Kreis*) in einem Graphen $G = (V, E)$ ist eine Menge $V_z = \{v_1, .., v_n\} \subseteq V$ mit der Eigenschaft $\forall i < n.\, \{v_i, v_{i+1}\} \in E \wedge \{v_n, v_1\} \in E)$
- Ein *Hamiltonscher Kreis* im Graphen $G = (V, E)$ ist ein Kreis, der nur aus Kanten aus E besteht und jeden Knoten genau einmal berührt.
- Ein Graph $G = (V, E)$ heißt zusammenhängend, wenn jeder Knoten in V von jedem anderen Knoten über Kanten aus E erreichbar ist.
- Ein *Baum* ist ein zyklenfreier zusammenhängender Graph.
- Ein *spannender Baum* in einem Graphen $G = (V, E)$ ist ein Subgraph $G_B = (V, E_B)$ von G, der ein Baum ist.
- Ein *minimal spannender Baum* in einem gewichteten Graphen (G, g) ist ein spannender Baum von G mit minimalem Gesamtgewicht.

Detailliertere Formulierungen mancher Konzepte findet man z.B. in

- S. O. Krumke, H. Noltemeier: *Graphentheoretische Konzepte und Algorithmen*, Teubner 2005.
- C. Meinel, M. Mundhenk: *Mathematische Grundlagen der Informatik*, Teubner 2002.
- K. Denecke: *Algebra und Diskrete Mathematik für Informatiker*, Teubner 2003.

Lösung 3.1 *Hierzu gibt es vorerst keine Musterlösung. Das folgende sind ein paar Gedanken zur Formalisierung*

Es macht wenig Sinne, Graphen axiomatisch zu beschreiben, da die übliche mathematische Definition eine direkte Beschreibung der Struktur von Graphen gibt. Diese läßt sich unmittelbar umsetzen.

Eine erste Idee wäre, Graphen als abhängige Datentypen zu formalisieren. Dies würde zu folgendem Ansatz führen

`Graph` $\equiv$ `V:FINSET` $\times$ `SET(V`$\times$`V)`

Hierbei ist jedoch zu bedenken, daß das `V` in `SET(V`$\times$`V)` eigentlich als endliche Aufzählung von Elementen gedacht ist, von der Verwendung her aber als Datentyp eingesetzt wird. Für diese Problematik gibt es zwei Lösungen.

1. `FINSET` wird als die Teilmenge der endlichen Datentypen in $\mathbb{U}_1$ modelliert. In diesem Fall können wir rechts in der Tat die normalen Typkonstruktoren einsetzen.

   Die Modellierung von `FINSET` ist aber nicht ganz einfach (Menge der Typen in $\mathbb{U}_1$, von denen es eine injektive Abbildung in einen der Typen $\{1..n\}$ gibt) und außerdem ist es nicht leicht, auf die Elemente von `V` zuzugreifen.

2. `FINSET` wird als `Set(`$\mathbb{N}$`)` modelliert. In diesem Fall haben wir Zugriff auf die Elemente von `V`. Wir müssen aber nun die "Liste" V zu einem Typ anheben ($\bar{V}$:= `{X:Set(`$\mathbb{N}$`)|X=V}`, um `E` definieren zu können.

Beide Lösungen stellen sich im Endeffekt als zu kompliziert heraus.

Einfacher ist es, zunächst die Struktur von Graphen (Eine Menge von Zahlen und eine von Zahlenpaaren) zu definieren und dann den Zusammenhang zwischen Knoten und Kanten als Prädikat zu formulieren

```
GraphStrukt ≡ Set(ℕ) × Set(ℕ×ℕ)
isgraph(G) = let (V,E)=G in ∀e∈E. e.1∈V ∧ e.2∈V ∧ e.1≠e.2
Graph ≡ {G:GraphStrukt | isgraph(G)}
```

Analog für ungerichtete Graphen

```
binset(T) ≡ e,e:T×T// (e.1=e'.2 ∧ e.2=e'.1)
GraphStrukt-undirected ≡ Set(ℕ) × Set(binset(ℕ))
Graph ≡ {G:GraphStrukt | isgraph(G)}
```

Man beachte, daß durch die Verwendung des Quotiententyps in `binset` nach wie vor Paar-Operationen verwendet werden können.

Auf dieser Basis können nun die weiteren Konzepte formalisiert werden.

Lösung 3.2 *Ziel dieser Aufgabe ist es, die Formalisierung von Problemspezifikationen einzuüben und dabei eventuell fehlende Begriffe zu formalisieren, sofern dies für die Problemstellung sinnvoller ist als die Verwendung der entsprechenden komplexeren Ausdrücke.*

Die Bedingung dafür, daß die n Damen einander nicht schlagen können ist, daß in jeder waagerechten, senkrechten, aufwärts- und abwärts-diagonalen Reihe höchstens eine Dame steht. Da es nur n waagerechte und senkrechte Reihen gibt, muß jede Dame in genau einer dieser Reihen stehen. Damit genügt es, anstelle einer Repräsentation des gesamten Schachbretts die waagerechten Positionen der Damen in jeder senkrechten Reihe darzustellen, also eine Folge L von n Zahlen zwischen 1 und n zu verwalten. Da in jeder waagerechten Zeilee nur eine Dame stehen kann, darf diese Folge keine doppelten Vorkommen enthalten, muß also eine Permutation der Zahlen $\{1..n\}$ sein.

Zusätzlich müssen die aufwärts- und abwärts-diagonalen Reihen sicher sein: steht in Reihe `i` an Position L`[i]` eine Dame, so darf die Position L`[j]` der Dame in Reihe `j` nicht genau L`[i]+(j-i)` oder L`[i]-(j-i)` sein. Dies beschreiben wir durch zwei neue Einzelbegriffe, die wir insgesamt mit dem Begriff `safe(`L`)` zusammenfassen.

`perm(`L`,`S`)` $\equiv$ `nodups(`L`)` $\wedge$ `range(`L`)=`S

`free_up_diagonal(`L`)` $\equiv$ $\forall$`i`$\in$`domain(`L`).`$\forall$`j`$\in$`domain(`L`).i`$\neq$`j` $\Rightarrow$ L`[i]+(j-i)`$\neq L$`[j]`

`free_down_diagonal(`L`)` $\equiv$ $\forall$`i`$\in$`domain(`L`).`$\forall$`j`$\in$`domain(`L`).i`$\neq$`j` $\Rightarrow$ L`[i]-(j-i)`$\neq L$`[j]`

`safe(`L`)` $\equiv$ `free_up_diagonal(`L`)` $\wedge$ `free_down_diagonal(`L`)`

Die Spezifikation, die eine Erweiterung des 8-Dame Problems auf beliebig große Schachbretter ist, lautet nun

$\forall n$`:`$\mathbb{N}$`.` $\exists$`NQ:Set(Seq(`$\mathbb{N}$`)).` `NQ = {nq | perm(nq,{1..`n`}) `$\wedge$` safe(nq)}`

In der allgemeinen Formulierung formaler Spezifikationen würde das wie folgt aussehen.

`FUNCTION` *queens*`(`n`:`$\mathbb{N}$`):Set(Seq(`$\mathbb{Z}$`)) WHERE true`
`  RETURNS {nq | perm(nq,{1..`n`}) `$\wedge$` safe(nq)}`

Im Verlauf der Übung wurden die beiden Bedingungen `free_up_diagonal` *und* `free_down_diagonal` *in einem einzigen Prädikat vereinigt:*
`safe(`L`)` $\equiv$ $\forall$`i`$\in$`domain(`L`).`$\forall$`j`$\in$`domain(`L`).i`$\neq$`j` $\Rightarrow$ `|`L`[i]-`L`[j]| `$\neq$` |i-j|`

Eine Modellierung mit Arrays wäre ebenfalls möglich, würde aber zu einer deutlich aufwendigeren Spezifikation und Lösung führen.