Berthold Hoffmann <hof>
Christian Maeder <maeder>
Klaus Lüttich <luettich>
Pascal Schmidt <pharao90>
Felix Beckwermert <foetus>
Dennis Walter <dw>
Franck Ngueuleu <nngf>

6. Übungsblatt

Ausgabe: 26.1.2004

Bearbeitungszeit: 9 Tage, bis 5.2.2004 mittags

Dieses Übungsblatt soll in PROLOG bearbeitet werden.

13 *Geordnete Bäume* — *6 Punkte*

Geordnete Bäume sollen entweder aus einem Blatt `leaf` bestehen, oder sie sind von der Form `node(`l`,`x`,`r`)`, wobei x eine Zahl, und l und r geordnete Bäume mit Einträgen kleiner bzw. größer als x sind. (Da Prolog ungetypt ist, brauchen und können Sie geordnete Bäume nicht als Typ definieren, sollten es aber bei den Fragen beachten.)

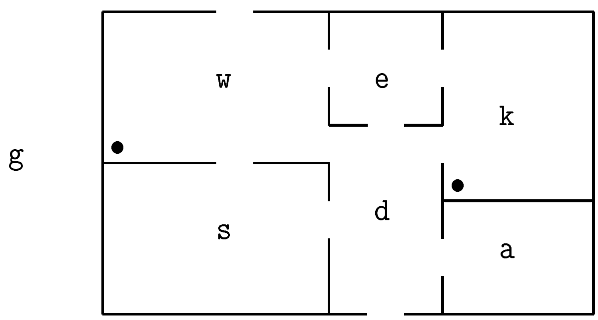
Definieren Sie zwei Prädikate auf geordneten Bäumen:

1. Das Prädikat `insert(`x`,`t`,`t'`)` soll genau dann *yes* antworten, wenn t' der geordnete Baum ist, der durch geordnetes Einfügen von x in den geordneten Baum t entsteht. Das Prädikat soll eindeutig (*determiniert*) sein, d.h. für jede Zahl und jeden geordneten Baum t nur *einen Baum* t' liefern. (Sollte t wider Erwarten nicht geordnet sein, ist die Antwort des Prädikats *undefiniert*, d.h. dann sind beliebige Antworten zulässig.)

2. Das Prädikat `element(`x`,`t`)` soll genau dann *yes* antworten, wenn eine Zahl x in einem geordneten Baum t enthalten ist.

14 *Pfade in Graphen* — *9 Punkte*

Definieren Sie für dem unten abgebildeten Wohnungsgrundriss einen Graphen, der die Türen zwischen den mit Buchstaben bezeichneten Flächen $F = \{\mathtt{a}, \mathtt{d}, \mathtt{e}, \mathtt{g}, \mathtt{k}, \mathtt{s}, \mathtt{w}\}$ als eine Menge von Fakten `tuer(`n_1`,`n_2`)` darstellt, und kennzeichnen sie einige Räume mit Telefon (z.B. • auf dem Plan) mit einem Fakt `telefon(`n`)`. Im Graphen sollten die `tuer`-Verbindungen ruhig *gerichtet* sein; für den Fakt `tuer(g,d)` sollte nicht auch noch die Gegenrichtung `tuer(d,g)` angegeben werden.



Schreiben Sie ein Prädikat `telefonsuche(`z`, `p`)`, das eine Liste p von den Räumen liefert, durch die man gehen muss, um ein Telefon zu finden. Dabei darf man natürlich in *beide* Richtung durch eine Tür gehen! Das Prädikat ist *nicht determiniert*, d.h. es kann im Allgemeinen mehrere Antworten p geben. Allerdings soll `telefonsuche` nur Wege *ohne Wiederholungen* finden! Dafür ist es freigestellt, den Weg in der Form `[`*`start`*`, .. , `*`ziel`*`]` oder aber umgekehrt anzugeben (auch wenn das nur einen Aufruf des Prädikats `reverse` kosten würde).

Benutzung von Prolog

Auf allen Unix-Rechnern in Ebene 0 ist ein Prolog-System von SWI installiert. Es kann mit `pl` aufgerufen werden.

Daraufhin erscheint ein "Prompt" `?-` als Eingabeaufforderung.

Als Erstes werden Sie vermutlich eine Programm-Datei ⟨*file*⟩`.pl` laden wollen. Das können sie tun, indem Sie `[`⟨*file*⟩`].` $\boxed{\text{enter}}$ eingeben.

Wenn dabei keine Fehler gemeldet werden, erscheint wieder der Prompt `?-`, und Sie können die in ⟨*file*⟩`.pl` definierten Prädikate benutzen. Dazu formulieren Sie *Fragen* wie z.B. `telefonsuche(g,P)` und schicken sie mit `.` $\boxed{\text{enter}}$ ab. Fragen können aber auch mehrere Prädikataufrufe enthalten, die mit Konjunktion (in Prolog als `,` geschrieben) verbunden sind.

Die Antwort auf die Frage lautet entweder *yes* oder *no*. Bei einer positiven Antwort wird eine Belegung der in der Frage genannten Variablen angegeben. Für unser Beispiel könnte die lauten: `P= [k,d,g]`. daraufhin kann man sich mit Eingabe von `;` oder `n` (jedenfalls *ohne* $\boxed{\text{enter}}$!) weitere Antworten auf die Frage aufzählen lassen, z.B. `P= [w,d,g]` und `P= [w,e,d,g]`, bis irgendwann die Antwort *no* gegeben wird, wenn keine weiteren Antworten gefunden werden können.

Für die Überprüfung eines Weges auf Wiederholungen kann das Prädikat `not/1` nützlich sein. (Hinter dem Schrägstrich `/` wird die *Stelligkeit* eines Prädikats angegeben.) Eine Frage `not(`$p(\cdots)$`)` *negiert* das Prädikat $p(\cdots)$. Also stellt `not(X=Y)` sicher, dass `X` und `Y` unterschiedliche Werte haben.

Durch `Ctrl-d` wird das Prolog-System beendet.