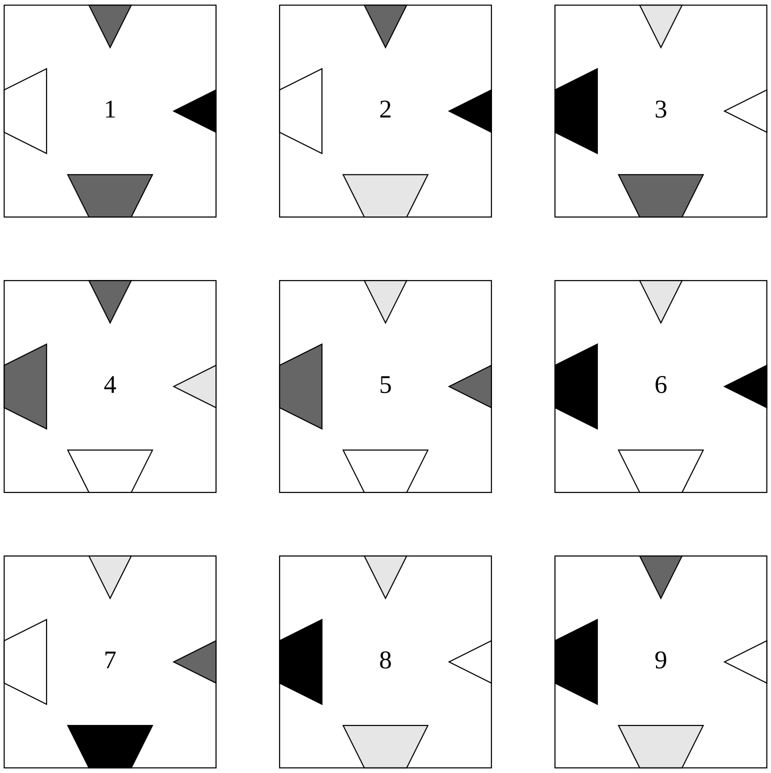
Michael Arndt
Fritz Hamm
Peter Schroeder-Heister

Sommersemester 2003
Übungsblatt 9
2.7.2003

Übungen zur Vorlesung *Logikprogrammierung*

Aufgabe 1 Puzzle (4 Punkte)

Schreiben Sie ein Prolog-Programm, welches das folgende Puzzle löst. Eine Lösung sei dabei eine Liste von neun Paaren `(N,O)`, wobei `N` die Nummer der Karte und `O` die Orientierung dieser Karte (z.B. in vielfachen von 90 Grad) angibt. Die Liste soll als Anordung von jeweils drei Karten in drei Reihern gelesen werden.



Die Spezifikation der Karten sei gegeben als:

```
card(1,rumpf/weiss,rumpf/dunkel,spitze/schwarz,spitze/dunkel),
card(2,rumpf/weiss,rumpf/hell,spitze/schwarz,spitze/dunkel),
card(3,rumpf/schwarz,rumpf/dunkel,spitze/weiss,spitze/hell),
card(4,rumpf/dunkel,rumpf/weiss,spitze/hell,spitze/dunkel),
card(5,rumpf/dunkel,rumpf/weiss,spitze/dunkel,spitze/hell),
card(6,rumpf/schwarz,rumpf/weiss,spitze/schwarz,spitze/hell),
card(7,rumpf/weiss,rumpf/schwarz,spitze/dunkel,spitze/hell),
card(8,rumpf/schwarz,rumpf/hell,spitze/weiss,spitze/hell),
card(9,rumpf/schwarz,rumpf/hell,spitze/weiss,spitze/dunkel).
```

Aufgabe 2 Parsing (2 Punkte)

Eine *aussagenlogische Formel* ist wie folgt definiert:

- Jedes atomare Aussagenzeichen A, B, C, D, E, F ist eine Formel.
- Wenn ϕ eine Formel ist, dann auch $\neg\phi$.
- Wenn ϕ, ψ Formeln sind, dann auch $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$.

Definieren Sie ein Prädikat `parse(TokenList,ParseTerm)`, welches eine Liste von Eingabetokens in einen funktionalen Term übersetzt, der die Struktur der durch TokenList ausgedrückten aussagenlogischen Formel wiedergibt, sofern TokenList einer aussagenlogischen Formel entspricht, und ansonsten `No` ausgibt. Den Tokens `aT`,..., `fT` werden dabei Terme `a`,..., `f` zugeordnet, und den Tokens `negT`, `konT`, `disT`, `impT`, `gdwT` die Funktionszeichen `neg`, `kon`, `dis`, `imp`, `gdw`. Den Tokens `klaT` und `klzT` werden natürlich keine Terme oder Funktionszeichen zugewiesen; sie gehen implizit in die Struktur des funktionalen Terms ein.

Beispiele:

```
?- parse([aT, impT, klaT, aT, disT, bT, klzT],T).
T = imp(a, dis(a, b))
Yes
?- parse([klzT, aT, impT, aT, klaT, disT, bT],T).
No
```