Universität Karlsruhe (TH)
Lehrstuhl für Programmierparadigmen

Theorembeweiser und ihre Anwendungen SS 2009 http://pp.info.uni-karlsruhe.de/
Übungsleiter: Daniel Wasserrab wasserra@ipd.info.uni-karlsruhe.de

Übungsblatt 11 Besprechung: 07.07.2009

1 Rotation mal ungewöhnlich

Wir definieren eine Funktion auf Listen, welche das erste Element an die letzte Stelle schiebt:

```
fun rot :: "'a list ⇒ 'a list"
where "rot [] = []"
  | "rot [x] = [x]"
  | "rot (x#y#zs) = y # rot(x#zs)"
```

Die von Isabelle automatisch erstellte Rekursionsregel lautet `rot.induct:`
`⟦P []; ⋀x. P [x]; ⋀x y zs. P (x # zs) ⟹ P (x # y # zs)⟧ ⟹ P a0`
Damit beweisen Sie bitte folgende Aussagen mittels eines *verständlichen* Isar-Beweises:

```
lemma "length (rot xs) = length xs"
oops

lemma "xs ≠ [] ⟹ rot xs = tl xs @ [hd xs]"
oops
```

2 Reflexiv-transitive Hülle

Wir definieren die reflexiv-transitive Hülle einer binären Prädikats `r` mittels eines induktiven Prädikats:

```
inductive rtc :: "('a ⇒ 'a ⇒ bool) ⇒ 'a ⇒ 'a ⇒ bool" ("(_*)" [1000] 1000)
for r::"'a ⇒ 'a ⇒ bool"
where refl: "r* x x"
  | step: "⟦r x y; r* y z⟧ ⟹ r* x z"
```

Anstatt `rtc r` darf man also `r*` schreiben. Auch hier generiert Isabelle automatisch eine Induktionsregel `rtc.induct:` `⟦r* x1 x2; ⋀x. P x x; ⋀x y z. ⟦r x y; r* y z; P y z⟧ ⟹ P x z⟧ ⟹ P x1 x2`
Zeigen Sie jetzt, dass `r*` tatsächlich transitiv ist:

```
lemma "⟦r* x y; r* y z⟧ ⟹ r* x z"
oops
```

Außerdem beweisen Sie noch, dass `r*` idempotent (die reflexiv-transitive Hülle von `r*` gleich `r*`) ist. Dazu brauchen Sie folgende Aussage: `ext: (⋀x. ?f x = ?g x) ⟹ ?f = ?g`

```
lemma rtc_idemp: "(r*)* = r*"
proof(rule ext)+
oops
```

Alle Beweise sollen natürlich mittels Isar erstellt werden.