

Praktikum Spezifikation und Verifikation

Prädikatenlogik

Beweisen Sie die folgenden Formeln der Prädikatenlogik mit den Regeln des natürlichen Schließens wie auf Blatt 4.

Zusätzlich können Sie die folgenden Regeln verwenden:

exI: $P\ x \implies \exists x. P\ x$

exE: $[\exists x. P\ x; \bigwedge x. P\ x \implies Q] \implies Q$

allI: $(\bigwedge x. P\ x) \implies \forall x. P\ x$

allE: $[\forall x. P\ x; P\ x \implies R] \implies R$

lemma " $\exists x. (P\ x \longrightarrow (\forall x. P\ x))$ "

⋮

lemma " $(\exists x. \forall y. P\ x\ y) \longrightarrow (\forall y. \exists x. P\ x\ y)$ "

⋮

lemma " $(\forall x. P\ x \longrightarrow Q) = ((\exists x. P\ x) \longrightarrow Q)$ "

⋮

Das folgende Lemma (Cantor's Theorem) besagt, dass keine Funktion vom Typ $'a \Rightarrow 'a\ set$ surjektiv ist. Um es etwas einfacher zu machen, stellen wir $'a\ set$ als $'a \Rightarrow bool$ dar. Sie dürfen für diese Aufgabe *case_tac* und das Lemma *fun_cong*: $f = g \implies f\ x = g\ x$ zusammen mit *drule_tac* benutzen

lemma " $\forall f :: 'a \Rightarrow 'a \Rightarrow bool. \neg(\forall y. \exists x. f\ x = y)$ "

⋮

Merge-Sort

Implementieren Sie den Algorithmus Merge-Sort: Eine Liste wird sortiert, indem sie in der Mitte geteilt wird, die Teillisten sortiert werden, und danach zur geordneten Liste gemerged werden.

Definieren Sie dazu mit *recdef* zwei Funktion

```
consts merge :: "nat list × nat list ⇒ nat list"  
consts msort :: "nat list ⇒ nat list"
```

und zeigen Sie

```
theorem "sorted (msort xs)"  
  :
```

```
theorem "count (msort xs) x = count xs x"  
  :
```

▷ Hinweise

- Informationen zu *recdef* finden Sie im Tutorial, Abschnitt 3.5.
- Um die Liste in zwei ungefähr gleich große Hälften zu teilen, können Sie die Funktionen *x div 2*, *take n xs* und *drop n xs* benutzen. Die Funktion *take* gibt die ersten *n* Elemente von *xs* zurück, die Funktion *drop* liefert die Liste *xs* bis auf die ersten *n* Elemente.
- Sie können die folgenden Lemmas benutzen:
linorder_not_le: $(\neg x \leq y) = (y < x)$
order_less_le: $(x < y) = (x \leq y \wedge x \neq y)$
div_if: $0 < n \implies m \text{ div } n = (\text{if } m < n \text{ then } 0 \text{ else } \text{Suc } ((m - n) \text{ div } n))$
min_def: $\text{min } a \ b \equiv \text{if } a \leq b \text{ then } a \text{ else } b$

▷ Abgabe 5. Juni 2002