Karlsruher Institut für Technologie

Institut für Programmstrukturen und Datenorganisation
Lehrstuhl Programmierparadigmen
Am Fasanengarten 5
76131 Karlsruhe
http://pp.ipd.kit.edu/

Theorembeweiserpraktikum – SS 2018

http://pp.ipd.kit.edu/lehre/SS2018/tba

Blatt 1: Deduktion

Abgabe: 23. April 2018, 12:00 Uhr
Besprechung: 24. April 2018

1 Natürliches Schließen

In dieser Aufgabe geht es um den Kalkül des natürlichen Schließens, mit dessen Hilfe einige Lemmas der Aussagen-Logik bewiesen werden sollen (nächste Seite).
Für die Beweise gelten die folgenden Spielregeln:

- Es dürfen nur die Befehle **proof**, **assume**, **have**, **show**, **next**, **qed**, und **from** verwendet werden, sowie darauf aufbauende Abkürzungen wie **then**, **with**, **..** und **.**.
- Der Befehl **proof** darf nur als **proof** (`rule` *regel*) (oder als **proof** -) verwendet werden, wobei die Regel eine der folgenden ist: (Anzeigen der Lemmas mittel **thm** *lemma-Name*)

  impI: $(A \Longrightarrow B) \Longrightarrow A \longrightarrow B$, *impE*: $A \longrightarrow B \Longrightarrow A \Longrightarrow (B \Longrightarrow C) \Longrightarrow C$,
  conjI: $A \Longrightarrow B \Longrightarrow A \wedge B$, *conjE*: $A \wedge B \Longrightarrow (A \Longrightarrow B \Longrightarrow C) \Longrightarrow C$,
  disjI1: $A \Longrightarrow A \vee B$, *disjE*:
  disjI2: $B \Longrightarrow A \vee B$, $A \vee B \Longrightarrow (A \Longrightarrow C) \Longrightarrow (B \Longrightarrow C) \Longrightarrow C$,
  notI: $(A \Longrightarrow False) \Longrightarrow \neg A$, *notE*: $\neg A \Longrightarrow A \Longrightarrow B$,
  iffI: $(A \Longrightarrow B) \Longrightarrow (B \Longrightarrow A) \Longrightarrow A \longleftrightarrow B$,
  iffE: $A \longleftrightarrow B \Longrightarrow (A \longrightarrow B \Longrightarrow B \longrightarrow A \Longrightarrow C) \Longrightarrow C$,
  ccontr: $(\neg A \Longrightarrow False) \Longrightarrow A$
  classical: $(\neg A \Longrightarrow A) \Longrightarrow A$

  Alle diese Regeln, außer den letzten beiden, sind als Standard-Regeln vorgegeben, das heißt der Befehl **proof** (`rule`) (oder kurz **proof**) wählt die passende Regel aus, auch ohne dass man sie explizit angibt. Lassen Sie nur Namen von Regeln weg, die sie zuvor zumindest einmal explizit verwendet haben.

Beispiel

```
lemma imp_uncurry: "(P ⟶ (Q ⟶ R)) ⟶ P ∧ Q ⟶ R"
proof (rule impI)
  assume PQR: "P ⟶ (Q ⟶ R)"
  show "P ∧ Q ⟶ R"
  proof — Das (rule impI) kann wegelassen werden
    assume "P ∧ Q"
    then have "P" by (rule conjE)
    with PQR
    have "Q ⟶ R" by (rule impE)
```

```
    from ‹P ∧ Q›
    have "Q".. — Hier steht eigentlich by (rule conjE)
    with ‹Q ⟶ R›
    show R..
  qed
qed

lemma I: "A ⟶ A"
⟨solution⟩

lemma "A ∧ B ⟶ B ∧ A"
⟨solution⟩

lemma "A ∧ B ⟶ A ∨ B"
⟨solution⟩

lemma "((A ∨ B) ∨ C) ⟶ A ∨ (B ∨ C)"
⟨solution⟩

lemma K: "A ⟶ B ⟶ A"
⟨solution⟩

lemma "A ∨ A ⟷ A ∧ A"
⟨solution⟩

lemma S: "(A ⟶ B ⟶ C) ⟶ (A ⟶ B) ⟶ A ⟶ C"
⟨solution⟩

lemma "(A ⟶ B) ⟶ (B ⟶ C) ⟶ A ⟶ C"
⟨solution⟩

lemma "¬ ¬ A ⟶ A"
⟨solution⟩

lemma "A ⟶ ¬ ¬ A"
⟨solution⟩

lemma "(¬ A ⟶ B) ⟶ (¬ B ⟶ A)"
⟨solution⟩

lemma "((A ⟶ B) ⟶ A) ⟶ A"
⟨solution⟩

lemma "A ∨ ¬ A"
⟨solution⟩

lemma deMorgan1: "¬ (A ∨ B) ⟷ ¬ A ∧ ¬ B"
⟨solution⟩

lemma deMorgan2: "¬ (A ∧ B) ⟷ ¬ A ∨ ¬ B"
⟨solution⟩
```

Anmerkung: Ist Ihnen bei den Beweisen der De Morgan-Regeln etwas aufgefallen?