Formale Grundlagen der Informatik 3

Prof. Dr. Christoph Walther / Visar Januzaj, Nathan Wasser
Technische Universität Darmstadt — Wintersemester 2011/12

Übung 2

Version 1 vom 11.11.2011

Aufgabe 2.1 (Kalküle)

In dieser Aufgabe betrachten Sie verschiedene Kalküle, die für zwei Zeichenketten $\alpha, \beta \in \Sigma^*$ feststellen, ob α eine Teilkette von β ist.

(a) Geben Sie einen geeigneten Kalkül (nach dem Schema aus Folie 5 von Kapitel 4) an, der auf der Zerlegung von Zeichenketten in Teilketten basiert. Verwenden Sie als Sprache des Kalküls $\Sigma^* \times \Sigma^*$, geben Sie Regeln sowie einen Herleitungsbegriff an und erläutern Sie kurz die Semantik des Kalküls.

(b) Entwerfen Sie nun einen Kalkül, der die Zeichenketten Zeichen für Zeichen abarbeitet. Verwenden Sie dazu die Sprache $\Sigma^* \times \Sigma^* \times \{\mathsf{G}, \mathsf{T}\}$. Nutzen Sie dabei G und T um zu markieren, ob noch die gesamte Zeichenkette α oder nur ein Teil davon gefunden werden soll.

(c) Ihr Kalkül aus Teilaufgabe (b) ist nicht-deterministisch, da ein Tripel aus $\Sigma^* \times \Sigma^* \times \{\mathsf{G}, \mathsf{T}\}$ nicht genügend Information für ein Backtracking enthält, das notwendig ist, falls ein Präfix von α vor α in β auftritt, z.B. für $\alpha = abc$ und $\beta = ababc$ mit $a, b, c \in \Sigma$ und $a \neq c$.

Geben Sie einen deterministischen Kalkül an, der die Zeichenketten Zeichen für Zeichen abarbeitet. Verwenden Sie dazu die Sprache $\Sigma^* \times \Sigma^* \times \Sigma^*$.

(d) Überprüfen Sie mit dem letzten Kalkül, welche der folgenden Zeichenketten Teilketten von *abbacaab* sind. Geben Sie alle für den Nachweis notwendigen Herleitungen an.

- *caac*
- *bbac*
- *abca*

Hinweis: Bearbeiten Sie die folgenden Beweisaufgaben mit ✓`eriFun` in der gegebenen Reihenfolge, damit die zu übenden Effekte sichtbar werden. In den Aufgaben wird Bezug auf in Übung 1 zu implementierende Prozeduren genommen. Diese finden Sie in der beiliegenden Datei `Uebung2.vf`.

Aufgabe 2.2 (Addition)

(a) Beweisen Sie die Lemmata „`0 + x = x`“ und „`x + 0 = x`“. Vergleichen Sie die Beweise. Überlegen Sie, worauf die Unterschiede in den Beweisen zurückzuführen sind.

(b) Lassen Sie mit „Generate Lemma“ alle für `+` angebotenen Lemmata erzeugen. Beweisen oder widerlegen Sie jedes dieser Lemmata.

(c) Geben Sie zu allen widerlegten Lemmata ein Gegenbeispiel in Form einer Substitution an.

(d) Vergleichen Sie die Beweise von „`+` is left-cancelable“ und „`+` is right-cancelable“. Überlegen Sie, worauf die Unterschiede in den Beweisen zurückzuführen sind.

Aufgabe 2.3 (Use Lemma)

Ziel dieser Aufgabe ist es, die interaktive Verwendung eines Lemmas mit „Use Lemma“ zu erlernen. Außerdem soll sie Ihnen helfen, Argumentationslücken der eigenen Beweisidee zu erkennen und diese durch den Beweis zusätzlicher Hilfslemmata zu schließen. In Aufgabe 2.4 sowie in den praktischen Übungen können Sie diese Fähigkeiten einsetzen, um schwierigere Beweisprobleme zu lösen.

(a) Beweisen Sie das Lemma „`k` $\neq$ `∅` $\rightarrow$ `but_last(k)` $\neq$ `k`“. Der Beweis sollte automatisch gelingen. Zur Vorbereitung der Aufgabe 2.4, in der eine ähnliche Aussage (Ungleichheit zweier Listen) nicht automatisch bewiesen wird, führen wir den Beweis nun ein weiteres Mal, und zwar mit einer Interaktion.

(b) Schneiden Sie den Beweis dazu am Wurzelknoten ab. Wir versuchen zu zeigen, dass „`but_last(k)`“ und „`k`“ ungleich sind, indem wir nachweisen, dass ihre Längen bereits unterschiedlich sind. Formulieren Sie den Kern dieser Beweisidee als Lemma „`length(k)` $\neq$ `length(l)` $\rightarrow$ `k` $\neq$ `l`“ und beweisen Sie dieses.

(c) Wenden Sie auf den Wurzelknoten des Beweisbaums zu „`k` $\neq$ `∅` $\rightarrow$ `but_last(k)` $\neq$ `k`“ das Lemma „`length(k)` $\neq$ `length(l)` $\rightarrow$ `k` $\neq$ `l`“ mittels der HPL-Regel „Use Lemma“ (aus dem Menü „Proof“, im Untermenü „Proof Rules“) und einer geeigneten Substitution an.

(d) Leider bringt die eben beschriebene Interaktion den Beweis nicht voran. Entweder war die Interaktion ungeeignet, oder in unserer Argumentationsidee (Ungleichheit zweier Listen aus ungleichen Längen folgern) war eine so große Lücke, dass ✓`eriFun` sie nicht automatisch schließen konnte. In diesem Fall hat unsere Argumentation eine zu große Lücke: Es fehlt eine Aussage über die Länge von „`but_last(k)`“. Ergänzen und beweisen Sie ein entsprechendes Lemma.

(e) Schneiden Sie die interaktive „Use Lemma“-Anwendung ab und versuchen Sie erneut, „Use Lemma“ anzuwenden (alternativ können Sie auch nur die an „Use Lemma“ anschließende „Simplification“ abschneiden und erneut ausführen lassen). Das Ergebnis offenbart eine weitere Argumentationslücke: Wir haben stillschweigend vorausgesetzt, dass eine natürliche Zahl „`n`$\neq$`0`“ verschieden von „`pred(n)`“ ist. Formulieren Sie dies als Lemma und vollenden Sie den Beweis.

Aufgabe 2.4 (Use Lemma)

(a) Lassen Sie mit „Generate Lemma“ (aus dem Menü „Program“) alle für „append“ angebotenen Lemmata erzeugen. Beweisen oder widerlegen Sie jedes dieser Lemmata.

Hinweis: Der Beweis von „append is right-cancelable“ ist der einzige Beweis in dieser Aufgabe, der etwas Arbeit erfordert (ca. 2 Hilfslemmata und 2 Interaktionen mit „Use Lemma“). Starten Sie zunächst einen automatischen Beweisversuch mit „Verify“ (aus dem Menü „Program“) und vervollständigen Sie den Beweis, indem Sie diejenigen Ungleichheiten von Listen, die ✓`eriFun` nicht automatisch nachweisen konnte, interaktiv wie in Aufgabe 2.3 beweisen.

(b) Geben Sie zu allen widerlegten Lemmata ein Gegenbeispiel in Form einer Substitution an.

(c) Geben Sie zu den beiden interaktiven Anwendungen von „Use Lemma“ die verwendeten Substitutionen an.

(d) Überlegen Sie sich, weshalb für „append is right-cancelable“ nicht so ein einfacher Beweis wie für „`+` is right-cancelable“ möglich ist.