Fortgeschrittene Konzepte der Programmierung UE 04

Steffen Anhäuser, Jan-Erik Menzel, Melina Morch

Aufgabe 4.1

Die erweiterte Evaluationsregel:

$$\frac{t_x \to t'_x}{\{l_i{=}v_i^{i\in 1..x-1}, l_i{=}t_x, l_k{=}t_k^{k\in x+1..n}\}.l_j \to \{l_i{=}v_i^{i\in 1..x-1}, l_i{=}t'_x, l_k{=}t_k^{k\in x+1..n}\}.l_j} \text{(E-ProjRcd)}$$

Aufgabe 4.2

Im Tutorium haben wir gezeigt, dass das Progress-Theorem für das getypte Lambda Kalkül mit Booleschen Werten und Listen nicht erhalten bleibt. Bleibt das Preservation-Theorem erhalten? Beweisen Sie Ihre Antwort.

Wir gehen davon aus, dass man Preservation mit einer Induktion über die Typregeln für Listen und Booleans beweisen kann. Leider haben wir keinen IA gefunden und können demnach nur spekulieren.

Aufgabe 4.3

Betrachten Sie die Evaluations- und Typregeln für Listen auf der nächsten Seite. Diese enthalten sehr viele Annotationen für Typen, die die Terme etwas schwerer lesbar machen und nicht unbedingt benötigt werden, weil die Typen stattdessen auch aus dem Kontext hergeleitet werden können.
Wäre es möglich alle Typannotationen aus den Regeln zu entfernen?

Bei der Typregel T-Nil ist die Typannotation notwendig. Salopp gesagt besitzt die leere Liste keinen Inhalt aus dem der Typ der Liste abgeleitet werden kann. Deswegen müssen wir den Typen angeben damit über den Listentypkonstrukter ein Typ für die Liste erzeugt werden kann. Tut man dies nicht, können die Ableitungsbäume nicht-deterministisch werden, da wir den Typen

z.B. erraten müssten. Für die danach folgenden Typregeln gilt das gleiche, da diese ein Listentypelement t_1 enthalten, welches ja auch nil seien kann. Dort würden die gleichen Probleme wie bei T-Nil auftreten, falls man die Typen nicht mit angibt. Bei den Evaluationsregeln gibt es unserer Meinung nach keine Probleme.