

Herleitung der Regel (ABSTR-n) für $n = 2$

Es ist zu zeigen, dass aus der Gültigkeit der Prämisse

$$\Gamma[\tau_1/id_1][\tau_2/id_2] \triangleright e :: \tau' \quad (\text{prem})$$

stets die Gültigkeit der Konklusion

$$\Gamma \triangleright \lambda(id_1, id_2) : \tau_1 * \tau_2. e :: \tau_1 * \tau_2 \rightarrow \tau' \quad (\text{conc})$$

folgt, wobei

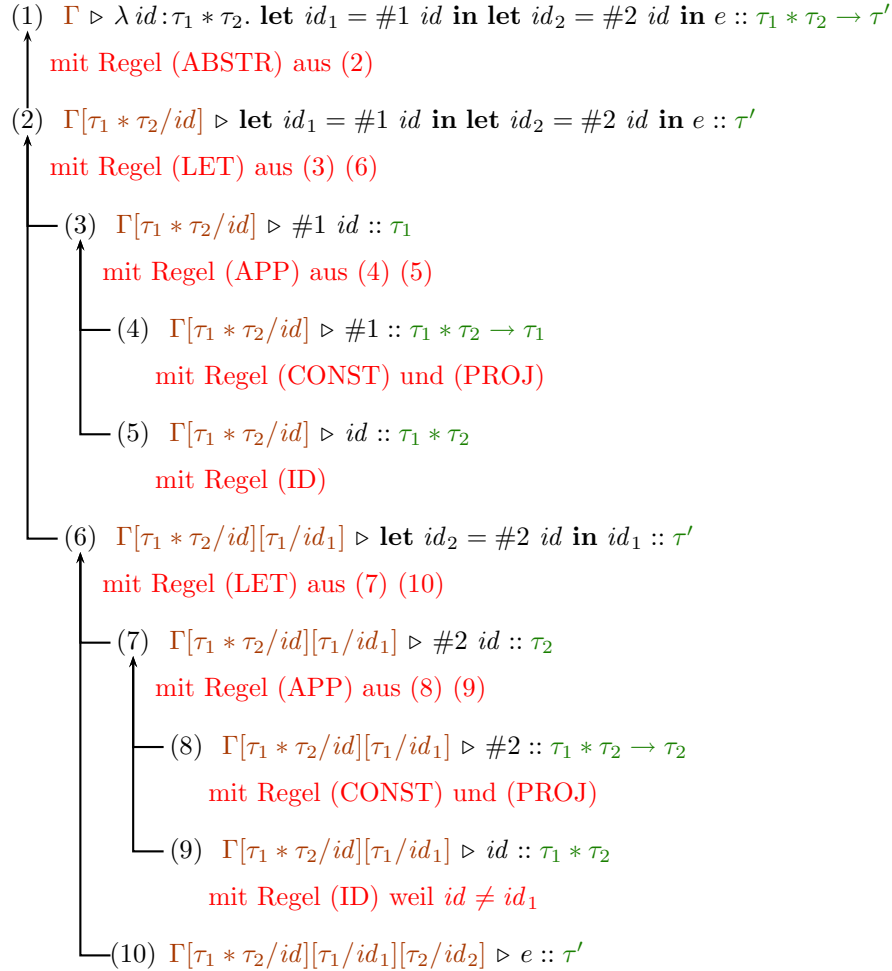
$$\lambda(id_1, id_2) : \tau_1 * \tau_2. e$$

als Abkürzung für die Kernsyntax

$$\lambda id : \tau_1 * \tau_2. \text{let } id_1 = \#1 id \text{ in let } id_2 = \#2 id \text{ in } e$$

zu lesen ist.

Da Gültigkeit nichts anderes ist als Ableitbarkeit mit den *ursprünglichen* Typregeln, versuchen wir (conc) mit diesen Regeln auf (prem) zurückzuführen.



(10) stimmt zwar nicht ganz mit (prem) überein, aber weil id nicht in e vorkommt, ist (10) genau dann gültig, wenn (prem) gültig ist. Diesen Zusammenhang werden wir später noch als Lemma formulieren.

Übrigens haben wir die Voraussetzungen $id \neq id_2$ und $id_1 \neq id_2$ bei der Herleitung nicht gebraucht. Die erste ist tatsächlich unnötig, man könnte auch $id = id_2$ anstelle eines neuen Namens id wählen, und die zweite ist eine Frage des (guten) Programmiersprachen-Designs: Es macht wenig Sinn, zwei Parameter gleichen Namens zu haben, weil der erste sofort durch den zweiten "überschrieben" wird. Also deutet es eher auf einen Programmierfehler hin, wenn so etwas in einem Programm auftaucht. Deshalb verbietet man es besser.