

Abgeleitete big step Regeln

$$\text{(ANDALSO-TRUE)} \quad \frac{e_1 \Downarrow \text{true} \quad e_2 \Downarrow v}{e_1 \text{ andalso } e_2 \Downarrow v}$$

Herleitung:

$$\begin{array}{l} (1) \quad e_1 \text{ andalso } e_2 \Downarrow v \\ \uparrow \text{ist syntaktischer Zucker für (2)} \\ (2) \quad \text{if } e_1 \text{ then } e_2 \text{ else } \text{false} \Downarrow v \\ \uparrow \text{mit Regel (COND-TRUE) aus (3) (4)} \\ \begin{array}{l} (3) \quad e_1 \Downarrow \text{true} \\ \quad \quad \text{1. Prämisse} \\ (4) \quad e_2 \Downarrow v \\ \quad \quad \text{2. Prämisse} \end{array} \end{array}$$

$$\text{(ANDALSO-FALSE)} \quad \frac{e_1 \Downarrow \text{false}}{e_1 \text{ andalso } e_2 \Downarrow \text{false}}$$

Herleitung:

$$\begin{array}{l} (1) \quad e_1 \text{ andalso } e_2 \Downarrow \text{false} \\ \uparrow \text{ist syntaktischer Zucker für (2)} \\ (2) \quad \text{if } e_1 \text{ then } e_2 \text{ else } \text{false} \Downarrow \text{false} \\ \uparrow \text{mit Regel (COND-FALSE) aus (3) (4)} \\ \begin{array}{l} (3) \quad e_1 \Downarrow \text{false} \\ \quad \quad \text{Prämisse} \\ (4) \quad \text{false} \Downarrow \text{false} \\ \quad \quad \text{mit Regel (VAL)} \end{array} \end{array}$$

$$\text{(ORELSE-TRUE)} \quad \frac{e_1 \Downarrow \text{true}}{e_1 \text{ orelse } e_2 \Downarrow \text{true}}$$

Herleitung:

$$\begin{array}{l} (1) \quad e_1 \text{ orelse } e_2 \Downarrow \text{true} \\ \uparrow \text{ist syntaktischer Zucker für (2)} \\ (2) \quad \text{if } e_1 \text{ then } \text{true} \text{ else } e_2 \Downarrow \text{true} \\ \uparrow \text{mit Regel (COND-TRUE) aus (3) (4)} \\ \begin{array}{l} (3) \quad e_1 \Downarrow \text{true} \\ \quad \quad \text{Prämisse} \\ (4) \quad \text{true} \Downarrow \text{true} \\ \quad \quad \text{mit Regel (VAL)} \end{array} \end{array}$$

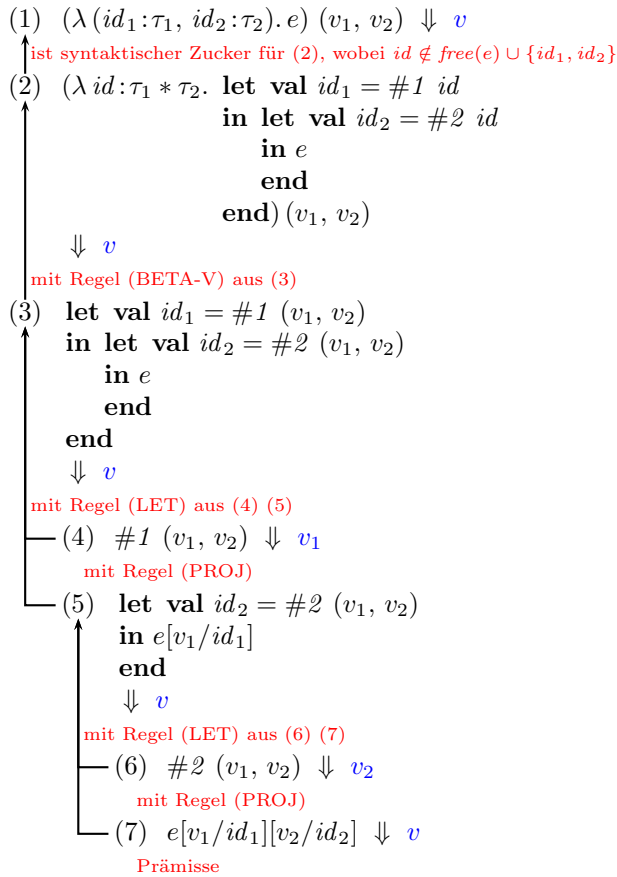
$$\text{(ORELSE-FALSE)} \quad \frac{e_1 \Downarrow \text{false} \quad e_2 \Downarrow v}{e_1 \text{ orelse } e_2 \Downarrow v}$$

Herleitung:

$$\begin{array}{l} (1) \quad e_1 \text{ orelse } e_2 \Downarrow v \\ \uparrow \text{ist syntaktischer Zucker für (2)} \\ (2) \quad \text{if } e_1 \text{ then } \text{true} \text{ else } e_2 \Downarrow v \\ \uparrow \text{mit Regel (COND-FALSE) aus (3) (4)} \\ \begin{array}{l} (3) \quad e_1 \Downarrow \text{false} \\ \quad \quad \text{1. Prämisse} \\ (4) \quad e_2 \Downarrow v \\ \quad \quad \text{2. Prämisse} \end{array} \end{array}$$

$$\text{(BETA-V-n)} \quad \frac{e[v_1/id_1] \dots [v_n/id_n] \Downarrow v}{(\lambda(id_1 : \tau_1, \dots, id_n : \tau_n). e)(v_1, \dots, v_n) \Downarrow v}$$

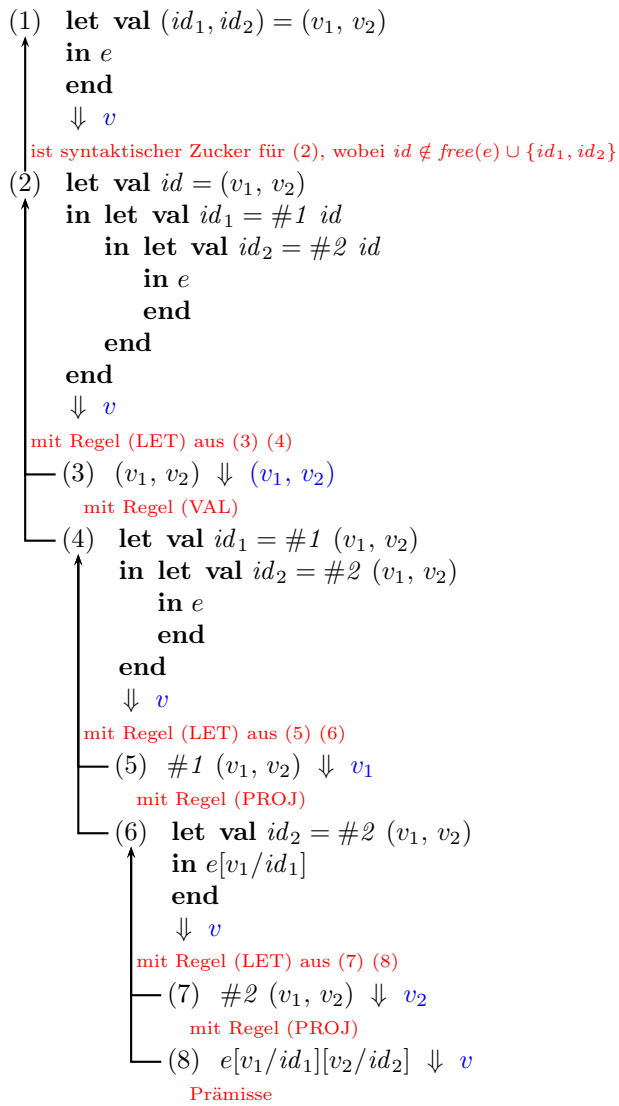
Herleitung für $n = 2$:



Man beachte: Beim Übergang von (2) zu (3) ist es wichtig, dass id ein neuer Name ist. Wegen $id \notin free(e)$ muss in e nichts ersetzt werden, und wegen $id \notin \{id_1, id_2\}$ sind die beiden Vorkommen von id im **let**-Ausdruck *frei*, d.h. sie werden durch (v_1, v_2) ersetzt.

$$(LET-n) \quad \frac{e_1 \Downarrow (v_1, \dots, v_n) \quad e_2[v_1/id_1] \dots [v_n/id_n] \Downarrow v}{\text{let val } (id_1, \dots, id_n) = e_1 \text{ in } e_2 \text{ end} \Downarrow v}$$

Herleitung für $n = 2$:



Auch hier ist es wichtig, dass id ein neuer Name ist, nämlich beim Übergang von (2) zu (4). Die Argumentation ist die gleiche wie bei (BETA-V-n).