Ternáris kondicionális programozási nyelve

Példaként arra, hogy a logika hogy kapcsolódik a programozáshoz, tekintsük az if then else parancsot tartalmazó legszűkebb még értelmes programozási nyelvet. Két

Termek

Az induktív típusok BNF-formában a következők:

$$\text{Term} ::= \texttt{tt} \mid \texttt{ff} \mid \texttt{if\_then\_else}\ \text{Term}\ \text{Term}\ \text{Term}$$

Típusolási szabályok

A típusolási szabályok az alábbiak:

$$\frac{\texttt{T-True}}{\vdash \texttt{tt} : \text{bool}} \qquad \frac{\texttt{T-False}}{\vdash \texttt{ff} : \text{bool}}$$

$$\frac{\begin{array}{c}\texttt{T-If}\\ \hline \vdash p : \text{bool} \quad \vdash q : \text{bool} \quad \vdash r : \text{bool}\end{array}}{\vdash \texttt{if\_then\_else}\ p\ q\ r : \text{bool}}$$

Példák a típusolás használatára:

$$\vdash \texttt{tt} : \text{bool}$$

$$\vdash \texttt{if\_then\_else}\ \texttt{tt}\ (\texttt{if\_then\_else}\ \texttt{tt}\ \texttt{ff}\ \texttt{tt})\ \texttt{ff} : \text{bool}$$

Példalként, ebben a nyelvben a vagy definiálható:

$$\texttt{OR}_1(x, y) = \texttt{if\_then\_else}\ x\ \texttt{tt}\ y$$

Egyenlőség és beta-redukció

Most az operacionális szemantikát definiáljuk (egy adott módon a sok közül). Ez azt jelenti, hogy megadjuk, a programfutás során a nyelv elemei hogyan működnek. Most csak egy olyan ekvivalencia reláció lesz, ami azonosítja a futást követő értéket az a futásra kijelölt értékkel.

$$\frac{\texttt{E-Refl}}{\vdash t \equiv t} \qquad \frac{\begin{array}{c}\texttt{E-Symm}\\ \hline \vdash t \equiv s\end{array}}{\vdash s \equiv t} \qquad \frac{\begin{array}{c}\texttt{E-Trans}\\ \hline \vdash t \equiv s \quad \vdash s \equiv u\end{array}}{\vdash t \equiv u}$$

$$\frac{\begin{array}{c}\texttt{E-If}\\ \hline \vdash p_1 \equiv p_2 \quad \vdash q_1 \equiv q_2 \quad \vdash r_1 \equiv r_2\end{array}}{\vdash \texttt{if\_then\_else}\ p_1\ q_1\ r_1 \equiv \texttt{if\_then\_else}\ p_2\ q_2\ r_2}$$

$$\frac{\texttt{E-beta-True}}{\vdash \texttt{if\_then\_else}\ \texttt{tt}\ p\ q \equiv p} \qquad \frac{\texttt{E-beta-False}}{\vdash \texttt{if\_then\_else}\ \texttt{ff}\ p\ q \equiv q}$$

A `beta_reduct` függvény rekurzív definíciója, az alábbi lesz és lényegében az előző definícióban a két utolsú úgy nevezett béta redexet, vargabetűt vágja le.

$$\texttt{beta\_reduct}(t) = \begin{cases} \texttt{beta\_reduct}(q) & \text{ha } t = \texttt{if\_then\_else tt } q \ r \\ \texttt{beta\_reduct}(r) & \text{ha } t = \texttt{if\_then\_else ff } p \ q \\ \begin{array}{c}\texttt{if\_then\_else} \\ (\texttt{beta\_reduct}(p)) \\ (\texttt{beta\_reduct}(q)) \\ (\texttt{beta\_reduct}(r))\end{array} & \text{egyébként} \end{cases}$$

Példa:

$$\texttt{beta\_reduct}(\texttt{if\_then\_else tt } q \ r) \equiv \texttt{beta\_reduct}(q)$$

Mélység és teljes beta-redukció

A mélység (`depth`) függvény rekurzív definíciója:

$$\texttt{depth}(t) = \begin{cases} 0 & \text{ha } t = \texttt{tt} \text{ vagy } t = \texttt{ff} \\ 1 + \max(\texttt{depth}(p), \max(\texttt{depth}(q), \texttt{depth}(r))) & \text{ha } t = \texttt{if\_then\_else } p \ q \ r \end{cases}$$

A `beta_reduct_full` függvény, amely a teljes beta-redukciót végzi el:

$$\texttt{beta\_reduct\_aux}(n, t) = \begin{cases} \texttt{beta\_reduct}(t) & \text{ha } n = 0 \\ \texttt{beta\_reduct}(\texttt{beta\_reduct\_aux}(m, t)) & \text{ha } n = S(m) \end{cases}$$

$$\texttt{beta\_reduct\_full}(t) = \texttt{beta\_reduct\_aux}(\texttt{depth}(t), t)$$

Gyenge normalizációs tétel

Tétel: Minden $t \in \text{Term}$ esetén a `beta_reduct_full`(t) kimenete `tt` vagy `ff`.

Denotációs szemantika

Denotációs szemantikának nevezzük, amikor megmondjuk, mivé fordítható át akár matematikai nyelvre, akár természetes nyelvre a programnyelv kifejezése.

A `denote_sem` függvény a következőképpen definiálható, amely a Term-ek értékét adja meg a denotációs szemantikában:

$$\texttt{denote\_sem}(t) = \begin{cases} \begin{array}{l}\ \texttt{if } (\texttt{denote\_sem}(p)) \\ \texttt{then } (\texttt{denote\_sem}(q)) \\ \texttt{else } (\texttt{denote\_sem}(r))\end{array} & \text{ha } t = \texttt{if\_then\_else } p \ q \ r \\ \texttt{true} & \text{ha } t = \texttt{tt} \\ \texttt{false} & \text{ha } t = \texttt{ff} \end{cases}$$

Példa a denotációs szemantika használatára:

$$\texttt{denote\_sem}(\texttt{if\_then\_else}\,\texttt{tt}\,\texttt{ff}\,\texttt{tt}) = \texttt{denote\_sem}(\texttt{beta\_reduct\_full}(\texttt{if\_then\_else}\,\texttt{tt}\,\texttt{ff}\,\texttt{tt}))$$