Dokazovanje ekvivalenc med programi

Dostikrat si želimo dokazati, da nek program deluje tako, kot želimo. Običajno je to precej zahtevna naloga, saj so programski jeziki nepredvidljivi. Na srečo so v OCaml-u čisti izrazi enakovredni svojim definicijam, zato lahko ene zamenjamo z drugimi in obratno. Da bomo imeli kaj, s čimer bomo lahko delali, si najprej definirajmo sledeče tri funkcije:

$$\texttt{[] @ ys} = \texttt{ys} \tag{1}$$
$$\texttt{(x::xs) @ ys} = \texttt{x::(xs @ ys)} \tag{2}$$

$$\texttt{obrni []} = \texttt{[]} \tag{3}$$
$$\texttt{obrni (x::xs)} = \texttt{obrni xs @ [x]} \tag{4}$$

$$\texttt{dolzina []} = \texttt{0} \tag{5}$$
$$\texttt{dolzina (x::xs)} = \texttt{1 + dolzina xs} \tag{6}$$

Trd. 1: `obrni [x]` = `[x]`

Trd. 2: `dolzina (xs @ ys)` = `dolzina xs + dolzina ys`

Trd. 3: `xs @ []` = `xs`

Trd. 4: `xs @ (ys @ zs)` = `(xs @ ys) @ zs`

Trd. 5: `obrni (xs @ ys)` = `obrni ys @ obrni xs`

Trd. 6: `dolzina (obrni xs)` = `dolzina xs`