CPSC 539B
Homework 1

4.2 About Numbers

Definition. *is-zero? = (λ (nat-fold n true (λ _ (λ _ false))))*.

Claim 4.2.1. *(eval (is-zero? z) true) is a valid judgement.*

Proof.

$$\dfrac{\dfrac{\dfrac{}{(\to \texttt{ (is-zero? z) (nat-fold z true (}\lambda\texttt{ \_ (}\lambda\texttt{ \_ false))))})}\text{(app)}}{(\to^{*} \texttt{ (is-zero? z) (nat-fold z true (}\lambda\texttt{ \_ (}\lambda\texttt{ \_ false))))})}\text{(step)} \quad \dfrac{\dfrac{}{(\to \texttt{ (nat-fold z true (}\lambda\texttt{ \_ (}\lambda\texttt{ \_ false))) true)})}\text{(nat-fold-zero)}}{(\to^{*} \texttt{ (nat-fold z true (}\lambda\texttt{ \_ (}\lambda\texttt{ \_ false))) true)})}\text{(step)}}{\dfrac{(\to^{*} \texttt{ (is-zero? z) true)})}{\texttt{(eval (is-zero? z) true)}}\text{(eval)}}\text{(trans)}$$

□

For the following equational proofs, we omit instantiations of the (trans) rule, and chain together → reduction steps directly. A lot of other steps are omitted as well, but I know that you know that I know what I'm talking about.

Claim 4.2.2. *(eval (is-zero? (s z)) false) is a valid judgement.*

Proof.

$$\begin{aligned}
\texttt{(is-zero? (s z))} &= \texttt{((}\lambda\texttt{ n (nat-fold n true (}\lambda\texttt{ _ (}\lambda\texttt{ _ false)))) (s z))} \\
&\xrightarrow{\text{(app)}} \texttt{(nat-fold (s z) true (}\lambda\texttt{ _ (}\lambda\texttt{ _ false)))} \\
&\xrightarrow{\text{(nat-fold-succ)}} \texttt{(((}\lambda\texttt{ _ (}\lambda\texttt{ _ false)) (s z)) (nat-fold z true (}\lambda\texttt{ _ (}\lambda\texttt{ _ false))))} \\
&\xrightarrow{\text{(app) via (app-compat)}} \texttt{((}\lambda\texttt{ _ false) (nat-fold z true (}\lambda\texttt{ _ (}\lambda\texttt{ _ false))))} \\
&\xrightarrow{\text{(app)}} \texttt{false} \\
&\therefore \texttt{ (eval (is-zero? (s z)) false)}
\end{aligned}$$

□

Claim 4.2.3. *(eval (is-zero? (s (s z))) false) is a valid judgement.*

Proof.

$$
\begin{aligned}
\texttt{(is-zero? (s (s z)))} &= \texttt{((λ n (nat-fold n true (λ \_ (λ \_ false)))) (s (s z)))} \\
&\xrightarrow{\text{(app)}} \texttt{(nat-fold (s (s z)) true (λ \_ (λ \_ false)))} \\
&\xrightarrow{\text{(nat-fold-succ)}} \texttt{(((λ \_ (λ \_ false)) (s (s z))) (nat-fold (s z) true (λ \_ (λ \_ false))))} \\
&\xrightarrow{\text{(app) via (app-compat)}} \texttt{((λ \_ false) (nat-fold (s z) true (λ \_ (λ \_ false))))} \\
&\xrightarrow{\text{(app)}} \texttt{false} \\
&\therefore \texttt{(eval (is-zero? (s (s z))) false)}
\end{aligned}
$$

□

Definition. + = *(λ n (λ m (nat-fold n m (λ _ (λ m (s m))))))*

Claim 4.2.4. *(eval ((+ z) z) z) is a valid judgement.*

Proof.

$$
\dfrac{\dfrac{\dfrac{A \qquad B}{(\to^* \; \texttt{((+ z) z) (nat-fold z z (λ \_ (λ m (s m)))))}}\,\text{(trans)} \qquad C}{(\to^* \; \texttt{((+ z) z) z)}}\,\text{(trans)}}{\texttt{(eval ((+ z) z) z)}}\,\text{(eval)}
$$

where A is the derivation tree

$$
\dfrac{\dfrac{}{(\to \; \texttt{((+ z) z) ((λ m (nat-fold z m (λ \_ (λ m (s m))))) z))}}\,\text{(app)}}{(\to^* \; \texttt{((+ z) z) ((λ m (nat-fold z m (λ \_ (λ m (s m))))) z))}}\,\text{(step)}
$$

B is the derivation tree

$$
\dfrac{\dfrac{}{(\to \; \texttt{((λ m (nat-fold z m (λ \_ (λ m m)))) z) (nat-fold z z (λ \_ (λ m (s m)))))}}\,\text{(app)}}{(\to^* \; \texttt{((λ m (nat-fold z m (λ \_ (λ m m)))) z) (nat-fold z z (λ \_ (λ m (s m)))))}}\,\text{(step)}
$$

and C is the derivation tree

$$
\dfrac{\dfrac{}{(\to \; \texttt{(nat-fold z z (λ \_ (λ m (s m)))) z)}}\,\text{(nat-fold-zero)}}{(\to^* \; \texttt{(nat-fold z z (λ \_ (λ m (s m)))) z)}}\,\text{(step)}
$$

□

Claim 4.2.5. *(eval ((+ z) (s z)) (s z)) is a valid judgement.*

Proof.

$$
\begin{aligned}
\texttt{((+ z) (s z))} &= \texttt{(((λ n (λ m (nat-fold n m (λ \_ (λ m (s m)))))) z) (s z))} \\
&\xrightarrow{\text{(app) via (app-compat), (app)}} \texttt{(nat-fold z (s z) (λ \_ (λ m (s m))))} \\
&\xrightarrow{\text{(nat-fold-zero)}} \texttt{(s z)} \\
&\therefore \texttt{(eval ((+ z) (s z)) (s z))}
\end{aligned}
$$

□

Claim 4.2.6. *(eval ((+ (s z)) (s z)) (s (s z))) is a valid judgement.*

Proof.

$$
\begin{aligned}
\texttt{((+ (s z)) (s z))} &= \texttt{(((λ n (λ m (nat-fold n m (λ \_ (λ m (s m)))))) (s z)) (s z))} \\
&\xrightarrow{\text{(app) via (app-compat), (app)}} \texttt{(nat-fold (s z) (s z) (λ \_ (λ m (s m))))} \\
&\xrightarrow{\text{(nat-fold-succ)}} \texttt{(((λ \_ (λ m (s m))) (s z)) (nat-fold z (s z) (λ \_ (λ m (s m)))))} \\
&\xrightarrow{\text{(app) via (app-compat), (app)}} \texttt{(s (nat-fold z (s z) (λ \_ (λ m (s m)))))} \\
&\xrightarrow{\text{(nat-fold-zero) via (succ-compat)}} \texttt{(s (s z))} \\
&\therefore \texttt{(eval ((+ (s z)) (s z)) (s (s z)))}
\end{aligned}
$$

□

Claim 4.2.7. *(eval ((+ (s z)) (s (s z))) (s (s (s z)))) is a valid judgement.*

Proof.

$$
\begin{aligned}
\texttt{((+ (s z)) (s (s z)))} &= \texttt{(((λ n (λ m (nat-fold n m (λ \_ (λ m (s m)))))) (s z)) (s (s z)))} \\
&\xrightarrow{\text{(app) via (app-compat), (app)}} \texttt{(nat-fold (s z) (s (s z)) (λ \_ (λ m (s m))))} \\
&\xrightarrow{\text{(nat-fold-succ)}} \texttt{(((λ \_ (λ m (s m))) (s z)) (nat-fold z (s (s z)) (λ \_ (λ m (s m)))))} \\
&\xrightarrow{\text{(app) via (app-compat), (app)}} \texttt{(s (nat-fold z (s (s z)) (λ \_ (λ m (s m)))))} \\
&\xrightarrow{\text{(nat-fold-zero) via (succ-compat)}} \texttt{(s (s (s z)))} \\
&\therefore \texttt{(eval ((+ (s z)) (s (s z))) (s (s (s z))))}
\end{aligned}
$$

□

Claim 4.2.8. *(eval ((+ (s (s z))) (s (s z))) (s (s (s (s z))))) is a valid judgement.*

Proof.

$$
\begin{aligned}
&\texttt{((+ (s (s z))) (s (s z)))} \\
&= \texttt{(((λ n (λ m (nat-fold n m (λ \_ (λ m (s m)))))) (s (s z))) (s (s z)))} \\
&\xrightarrow{\text{(app) via (app-compat), (app)}} \texttt{(nat-fold (s (s z)) (s (s z)) (λ \_ (λ m (s m))))} \\
&\xrightarrow{\text{(nat-fold-succ)}} \texttt{(((λ \_ (λ m (s m))) (s (s z))) (nat-fold (s z) (s (s z)) (λ \_ (λ m (s m)))))} \\
&\xrightarrow{\text{(app) via (app-compat), (app)}} \texttt{(s (nat-fold (s z) (s (s z)) (λ \_ (λ m (s m)))))} \\
&\xrightarrow{\text{by Claim 4.2.7 via (succ-compat)}} \texttt{(s (s (s (s z))))} \\
&\therefore \texttt{(eval ((+ (s (s z))) (s (s z))) (s (s (s (s z)))))}
\end{aligned}
$$

□