CS520, Handout 28, 5 December 2006

Positive Recursive Types

Not in [Pierce]

Syntax of Positive Recursive Types

(compare with [Handout 26, page 2])

V ranges over an infinite set of variables, the ***co-variables***.

W ranges over an infinite set of variables, the ***contra-variables***.

Co- and contra-variables are disjoint sets.

Syntax of Positive Recursive Types

(compare with [Handout 26, page 2])

V ranges over an infinite set of variables, the ***co-variables***.

W ranges over an infinite set of variables, the ***contra-variables***.

Co- and contra-variables are disjoint sets.

Co-types and ***contra-types*** are defined simultaneously.

$$
\begin{array}{lll}
R ::= & S ::= & T ::= R \mid S \\
\quad \texttt{Nat} \mid \texttt{Bool} & \quad \texttt{Nat} \mid \texttt{Bool} & \\
\quad V & \quad W & \\
\quad \texttt{Unit} & \quad \texttt{Unit} & \\
\quad \boxed{S \to R} & \quad \boxed{R \to S} & \\
\quad R \times R & \quad S \times S & \\
\quad R + R & \quad S + S & \\
\quad \mu V.R & \quad \mu W.S &
\end{array}
$$

The ***typing rules*** and the ***evaluation rules*** are the same as in Handout 26.

Simplifying for a comparison between ***recursive*** and ***positive recursive*** types.

Simplified Syntax of Positive Recursive Types

$$
\begin{array}{lll}
R ::= & S ::= & T ::= R \mid S \\
\quad \texttt{Nat} \mid \texttt{Bool} & \quad \texttt{Nat} \mid \texttt{Bool} & \\
\quad V & \quad W & \\
\quad \boxed{S \to R} & \quad \boxed{R \to S} & \\
\quad \mu V.R & \quad \mu W.S &
\end{array}
$$

With the simplified syntax of types we restrict attention to terms of the ***pure lambda-calculus*** (variables, abstractions and applications).

Simplifying for a comparison between ***recursive*** and ***positive recursive*** types.

Simplified Syntax of Positive Recursive Types

$$R ::= \mathtt{Nat} \mid \mathtt{Bool} \mid V \mid \boxed{S \to R} \mid \mu V.R \qquad S ::= \mathtt{Nat} \mid \mathtt{Bool} \mid W \mid \boxed{R \to S} \mid \mu W.S \qquad T ::= R \mid S$$

With the simplified syntax of types we restrict attention to terms of the ***pure lambda-calculus*** (variables, abstractions and applications).

> FACT: Every term of the pure lambda-calculus is typable in the system of ***recursive types***.

> FACT: If a term t of the pure lambda-calculus is typable in the system of ***positive recursive types***, then t is strongly normalizable.