Recursion

Recursion in $\lambda_{\to}$

- In $\lambda_{\to}$, all programs terminate. (Cf. Chapter 12.)
- Hence, untyped terms like `omega` and `fix` are not typable.
- But we can *extend* the system with a (typed) fixed-point operator...

Example

```
ff = λie:Nat→Bool.
       λx:Nat.
         if iszero x then true
         else if iszero (pred x) then false
         else ie (pred (pred x));

iseven = fix ff;

iseven 7;
```

New syntactic forms

$\mathtt{t} ::= \ldots$ *terms*

$\quad\mid\ \mathtt{fix\ t}$ *fixed point of* $\mathtt{t}$

New evaluation rules

$\boxed{\mathtt{t} \longrightarrow \mathtt{t}'}$

$$\mathtt{fix}\ (\lambda \mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_2) \longrightarrow [\mathtt{x} \mapsto (\mathtt{fix}\ (\lambda \mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_2))]\mathtt{t}_2 \qquad \text{(E-FixBeta)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{fix\ t}_1 \longrightarrow \mathtt{fix\ t}_1'} \qquad \text{(E-Fix)}$$

New typing rules

$\boxed{\Gamma \vdash \mathtt{t} : \mathtt{T}}$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_1 \to \mathtt{T}_1}{\Gamma \vdash \mathtt{fix}\ \mathtt{t}_1 : \mathtt{T}_1} \qquad \text{(T-Fix)}$$

A more convenient form

$$\texttt{letrec } x{:}T_1{=}t_1 \texttt{ in } t_2 \stackrel{\text{def}}{=} \texttt{let } x = \texttt{fix } (\lambda x{:}T_1.t_1) \texttt{ in } t_2$$

```
letrec iseven : Nat→Bool =
  λx:Nat.
    if iszero x then true
    else if iszero (pred x) then false
    else iseven (pred (pred x))
in
  iseven 7;
```