Termination

Eventuality

Transformation

Transitions

- Program: s1 $\rightsquigarrow$ s2 $\rightsquigarrow$ s3 $\rightsquigarrow$ ...
- Transformation si $\mapsto$ si
- Schema: s1 $\rightsquigarrow$ s2 $\rightsquigarrow$ s3 $\rightsquigarrow$ ...
- s $\rightsquigarrow$ s' if s $\rightsquigarrow$ s'

Homework

Example

$x - 0 \Rightarrow x$

$sx - sy \Rightarrow x - y$

$0 \div sy \Rightarrow 0$

$sx \div sy \Rightarrow s((x-y) \div sy)$

$0 + y \Rightarrow y$

$sx + y \Rightarrow s(x+y)$

$(x-y) - z \Rightarrow x - (y+z)$

Easy Rules

$x - 0 \Rightarrow x$

$0 \div sy \Rightarrow 0$

$0 + y \Rightarrow y$

Precedence

÷ , + > s > ~ (lr?)

Hard Rule

$$sx \div sy \Rightarrow s((x-y) \div sy)$$

Solution

$$sx \div sy \Rightarrow s((x-\cancel{y}) \div sy)$$

Problem

$$sx \div sy \Rightarrow s((x-y) \div sy) \Rightarrow s((u+v) \div sy)$$

Pairs

$sx - sy \Rightarrow x - y$

$sx \div sy \Rightarrow (x-y) \div sy$ $\quad$ $sx \div sy \Rightarrow x-y$

$sx + y \Rightarrow x + y$

$(x-y) - z \Rightarrow x - (y+z)$ $\quad$ $(x-y) - z \Rightarrow y + z$

Pairs - Colored

$sx \color{red}{-} sy \Rightarrow x \color{red}{-} y$

$sx \color{red}{\div} sy \Rightarrow (x-y) \color{red}{\div} sy$ $\quad$ $sx \color{red}{\div} sy \Rightarrow x \color{red}{-} y$

$sx \color{red}{+} y \Rightarrow x \color{red}{+} y$

$(x-y) \color{red}{-} z \Rightarrow x \color{red}{-} (y+z)$ $\quad$ $(x-y) \color{red}{-} z \Rightarrow y \color{red}{+} z$

Pairs - Separated

$\boxed{sx - sy \Rightarrow x - y}$

$\boxed{sx \div sy \Rightarrow (x-y) \div sy \qquad sx \div sy \Rightarrow x-y}$

$\boxed{sx + y \Rightarrow x + y}$

$\boxed{(x-y) - z \Rightarrow x - (y+z) \qquad (x-y) - z \Rightarrow y + z}$

Pairs - Separated

$$\boxed{sx + y \Rightarrow x + y}$$

Pairs - Separated

$$\boxed{sx \div sy \Rightarrow (x-y) \div sy \qquad sx \div sy \Rightarrow x-y}$$

Pairs - Separated

sx ÷ sy ⇨ x~   ÷ sy   sx ÷ sy ⇨ x ~

Pairs - Separated

$\boxed{sx - sy \Rightarrow x - y}$

$\boxed{(x-y) - z \Rightarrow x - (y+z) \quad (x-y) - z \Rightarrow y + z}$

Rules

$x - 0 \Rightarrow x$

$sx - sy \Rightarrow x - y$

$0 \div sy \Rightarrow 0$

$sx \div sy \Rightarrow s((x-y) \div sy)$

$0 + y \Rightarrow y$

$sx + y \Rightarrow s(x+y)$

$(x-y) - z \Rightarrow x - (y+z)$

Rules -

$x - \Rightarrow x$

$sx - \Rightarrow x -$

$0 \div sy \Rightarrow 0$

$sx \div sy \Rightarrow s((x-) \div sy)$

$0 + y \Rightarrow y$

$sx + y \Rightarrow s(x+y)$

$(x-) - \Rightarrow x -$

Rules $\gtrsim$

$x{-} \gtrsim x$

$sx{-} \gtrsim x{-}$

$0 \div sy \gtrsim 0$

$sx \div sy \gtrsim s((x{-}) \div sy)$

$0 + y \gtrsim y$

$sx + y \gtrsim s(x+y)$

$(x{-}){-} \gtrsim x{-}$

$$0 \leq y \Rightarrow T$$

$$sx \leq 0 \Rightarrow F$$

$$sx \leq sy \Rightarrow x \leq y$$

$$0 - y \Rightarrow 0$$

$$sx - y \Rightarrow if(sx \leq y, sx, y)$$

$$if(T, sx, y) \Rightarrow 0$$

$$if(F, sx, y) \Rightarrow s(x-y)$$

$$0 \div sy \Rightarrow 0$$

$$sx \div sy \Rightarrow s((x-y) \div sy)$$

$0 \leq y \Rightarrow T$

$sx \leq y \Rightarrow F$

$sx \leq sy \Rightarrow x \leq y$

$psx \Rightarrow x$

$x - 0 \Rightarrow x$

$x - sy \Rightarrow p(x-y)$

$gcd(sx,0) \Rightarrow s(x)$

$gcd(sx,sy) \Rightarrow if(y \leq x, sx, sy)$

$if(T,sx,sy) \Rightarrow gcd(x-y,sy)$

$if(F,sx,sy) \Rightarrow gcd(y-x,sx)$

$$
\begin{aligned}
\mathsf{le}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{le}(x,y)\\
\mathsf{app}(\mathsf{nil},y) &\to y\\
\mathsf{app}(\mathsf{add}(n,x),y) &\to \mathsf{add}(n,\mathsf{app}(x,y))\\
\mathsf{low}(n,\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{low}(n,\mathsf{add}(m,x)) &\to \mathsf{if}_{\mathsf{low}}(\mathsf{le}(m,n),n,\mathsf{add}(m,x))\\
\mathsf{if}_{\mathsf{low}}(\mathsf{true},n,\mathsf{add}(m,x)) &\to \mathsf{add}(m,\mathsf{low}(n,x))\\
\mathsf{if}_{\mathsf{low}}(\mathsf{false},n,\mathsf{add}(m,x)) &\to \mathsf{low}(n,x)\\
\mathsf{high}(n,\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{high}(n,\mathsf{add}(m,x)) &\to \mathsf{if}_{\mathsf{high}}(\mathsf{le}(m,n),n,\mathsf{add}(m,x))\\
\mathsf{if}_{\mathsf{high}}(\mathsf{true},n,\mathsf{add}(m,x)) &\to \mathsf{high}(n,x)\\
\mathsf{if}_{\mathsf{high}}(\mathsf{false},n,\mathsf{add}(m,x)) &\to \mathsf{add}(m,\mathsf{high}(n,x))\\
\mathsf{quicksort}(\mathsf{nil}) &\to \mathsf{nil}\\
\mathsf{quicksort}(\mathsf{add}(n,x)) &\to \mathsf{app}(\mathsf{quicksort}(\mathsf{low}(n,x)),\\
&\qquad\quad \mathsf{add}(n,\mathsf{quicksort}(\mathsf{high}(n,x))))
\end{aligned}
$$

Dataflow

Top Graph

- Pierre Réty & al. (1987) : Narrowing
- Jürgen Giesl & al. (2000) : Rewriting

Argument Graph

- Shuki Sagiv & al. (1991) : Logic languages
- Neil Jones & al. (2000) : Functional languages

Induction

Leaves

```
leaves(t) :=
    if leaf(t)
    then 1
    else leaves(left(t)) + leaves(right(t))
```

Counting Leaves

```
s := push(t,empty)
n := 0
loop while s ≠ empty
    h := top(s)
    s := pop(s)
    if leaf(h)
    then n := n + 1
    else s := push(left(h),push(right(h),s))
```

Correctness

- if s=t.e and n=0
- then eventually s=e and n=#(t)

Lemma

- if s=t.r and n=k
- then eventually s=r and n=k+#(t)

Induction (1)

- if s=leaf.r and n=k
- then eventually s=r and n=k+#(leaf)
- then eventually s=r and n=k+1

Induction (2)

- if s=b(lt,rt).r and n=k
- then s=lt.rt.r and n=k
- then eventually s=rt.r and n=k+#(lt)
- then eventually s=r and n=k+#(lt)+#(rt)
- then eventually s=r and n=k+#b(lt,rt)

Termination

- if s=t.e
- then eventually s=e

Lemma

- if s=t.r
- then eventually s=r

Ackermann

```
t := 1
s[t] := m
loop m := s[t]
     t := t-1
     if m=0
     then n := n+1
     else if n=0
     then t := t+1
          s[t] := m-1
          n := 1
     else t := t+2
          s[t-1] := m-1
          s[t] := m
          n := n-1
     until t=0
```

Termination

- If t=k then eventually t=k-1 and s[0:k-1] same
- Induction on (m,n) just after m := s[t]
- Case 1, m=0: t' = t-1
- Case 2, m>0, n=0: t' = t; m' = m-1
- Case 3, m,n>0: t' = t+1; m' = m; n' = n-1; s[t'] = m-1
- By induction, eventually t''=t; m'' = m-1