Joint Evaluation Contexts

Ben Lippmeier
University of New South Wales
FP-Syd 2011/07/21

STLC with Algebraic Data Types

```
Inductive exp : Type :=
  | XVar    : nat -> exp
  | XLam    : ty  -> exp -> exp
  | XApp    : exp -> exp -> exp

  | XCon    : datacon -> list exp -> exp
  | XCase   : exp     -> list alt -> exp

with alt      : Type :=
  | AAlt    : datacon -> list ty  -> exp -> alt.
```

Small Step Evaluation -- Function Application

$$\frac{x1 \Rightarrow x1'}{x1\ x2 \Rightarrow x1'\ x2} \qquad \frac{x2 \Rightarrow x2'}{v1\ x2 \Rightarrow v1\ x2'}$$

$$(\backslash z : t11.\ x12)\ v2 \Rightarrow x12[v2/z]$$

Small Step Evaluation -- Data Constructors

$$\frac{\texttt{x1} \Rightarrow \texttt{x1'}}{\texttt{D x1 x2 .. xn} \Rightarrow \texttt{D x1' x2 .. xn}}$$

$$\frac{\begin{array}{c}\texttt{x2} \Rightarrow \texttt{x2'} \\ \hline \texttt{D v1 x2 .. xn} \Rightarrow \texttt{D v1 x2' .. xn}\end{array} \quad \cdots \quad \texttt{xn} \Rightarrow \texttt{xn'}}{\texttt{D v1 v2 .. xn} \Rightarrow \texttt{D v1 v2 .. xn'}}$$

Evaluation Contexts

```
__ x2                    case __ of { ALTS }

v1 __

D __ x2 x3 x4 .. xn
D v1 __ x3 x4 .. xn
D v1 v2 __ x4 .. xn
D v1 v2 v3 __ .. xn
D v1 v2 v3 v4 .. __
          .
          .
```

Filling a Context

```
C1 ~ case __ of {ALTS}
```

```
C1 thingo => case thingo of {ALTS}
```

Small Step Evaluation with Contexts

$$\frac{\texttt{x1} \Rightarrow \texttt{x1'}}{\texttt{C x1} \Rightarrow \texttt{C x1'}}$$

```
__ x2                    case __ of { ALTS }

v1 __
```

Evaluation Contexts

```
Inductive exp_ctx : (exp -> exp) -> Prop :=
  | XcTop
    : exp_ctx  (fun x => x)

  | XcApp1
    :  forall x2
    ,  exp_ctx  (fun xx => XApp xx x2)

  | XcApp2
    :  forall v1
    ,  value v1
    -> exp_ctx  (fun xx => XApp v1 xx)

  | XcCon
    :  forall dc C
    ,  exps_ctx C
    -> exp_ctx  (fun xx => XCon dc (C xx))

  | XcCase
    :  forall alts
    ,  exp_ctx  (fun xx => XCase xx alts).
```

Progress

```
Theorem progress
 :  forall ds x t
 ,  TYPE ds nil x t
 -> value x \/ (exists x', STEP x x').
```

Progress

```
Theorem progress
 :  forall ds x t
 ,  TYPE ds nil x t
 -> value x \/ (exists x', STEP x x').
```

```
D x1 x2 x3 x4 x5
```

```
D x1 x2 x3 x4 x5
D v1 x2 x3 x4 x5
D v1 v2 x3 x4 x5
D v1 v2 v3 x4 x5
D v1 v2 v3 v4 x5
D v1 v2 v3 v4 v5
```

```
D __ x2 x3 x4 x5
D v1 __ x3 x4 x5
D v1 v2 __ x4 x5
D v1 v2 v3 __ x5
D v1 v2 v3 v4 __
```

Existence of a context

```
Lemma exps_ctx_run
 :   forall (P: exp -> Prop) xs
 ,   Forall (fun x => value x \/ P x) xs
 -> Forall value xs
 \/ (exists C x, exps_ctx C
                 /\ xs = C x
                 /\ P x).
```

Converting Big to Small Step evaluation

```
Lemma steps_in_XCon
  :   forall xs vs dc
  ,   Forall2 STEPS xs vs
  -> Forall  value vs
  -> STEPS (XCon dc xs) (XCon dc vs).
```

Expansion

```
IF    (D v1 x2 x3 x4 x5) -*> (D v1 v2 v3 v4 v5)
 AND x2 -*> v2
THEN (D v1 v2 x3 x4 x5) -*> (D v1 v2 v3 v4 v5)

IF    (D v1 v2 x3 x4 x5) -*> (D v1 v2 v3 v4 v5)
 AND x3 -*> v3
THEN (D v1 v2 v3 x4 x5) -*> (D v1 v2 v3 v4 v5)

IF    (D v1 v2 v3 x4 x5) -*> (D v1 v2 v3 v4 v5)
 AND x4 -*> v4
THEN (D v1 v2 v3 v4 x5) -*> (D v1 v2 v3 v4 v5)
```

Reusing the same reasoning as before fails...

```
Lemma steps_in_XCon
  :   forall xs vs dc
  ,   Forall2 STEPS xs vs
  -> Forall  value xs
  -> STEPS (XCon dc xs) (XCon dc vs).
```

```
Lemma exps_ctx2_run_bad
  :   forall xs vs
  ,   Forall2 STEPS xs vs
  -> Forall  value xs
  \/ (exists C x v
            ,   exps_ctx C
            /\ STEPS x v
            /\ xs = C x
            /\ vs = C v).
```

Reusing the same reasoning as before fails...

```
Lemma steps_in_XCon
  :   forall xs vs dc
  ,   Forall2 STEPS xs vs
  -> Forall  value xs
  -> STEPS (XCon dc xs) (XCon dc vs).
```

```
Lemma exps_ctx2_run_bad
  :   forall xs vs
  ,   Forall2 STEPS xs vs
  -> Forall  value xs
  \/ (exists C x v
             ,   exps_ctx C
             /\ STEPS x v
             /\ xs = C x
             /\ vs = C v).
```

```
v1 v2 __ x4 x5 x6
v1 v2 __ v4 v5 v6
```

We need a different context to match the results

```
Lemma steps_in_XCon
  :   forall xs vs dc
  ,   Forall2 STEPS xs vs
  -> Forall  value vs
  -> STEPS (XCon dc xs) (XCon dc vs).
```

```
Lemma exps_ctx2_run
  :   forall xs vs
  ,   Forall2 STEPS xs vs
  -> Forall  value xs
  \/ (exists C1 C2 x v
             ,   exps_ctx2 C1 C2
             /\ STEPS x v
             /\ xs = C1 x          v1 v2 __ x4 x5 x6
             /\ vs = C2 v).        v1 v2 __ v4 v5 v6
```

Joint contexts allow exps not yet evaluated to differ.

```
Inductive exps_ctx2
    :  (exp -> list exp)
    -> (exp -> list exp) -> Prop :=

  | Xsc2Head
    :  forall xs ys
    ,  exps_ctx2 (fun xx => xx :: xs)
                 (fun yy => yy :: ys)

  | Xsc2Cons
    :  forall v C1 C2
    ,  value v
    -> exps_ctx2 C1 C2
    -> exps_ctx2 (fun xx => v :: C1 xx)
                 (fun yy => v :: C2 yy).
```