

Scope in FL

Definitions of proposed syntactic terms

Here we use abstract syntax constructors; eg, $\text{seq}(e_1, \dots, e_n)$ denotes $\langle e_1, \dots, e_n \rangle$. This is just a way of avoiding the details of the syntax.

$\text{expr} := \text{atom} \mid \text{name} \mid \text{paren}(\text{expr}) \mid \text{seq}(\text{expr}_1, \dots, \text{expr}_n) \mid \text{comb}(\text{expr}_1 \dots \text{expr}_n)$
 $\text{appl}(\text{expr}_1, \text{expr}_2) \mid \text{lamb}(\text{pat}, \text{expr}) \mid \text{where}(\text{expr}, \text{env})$

$\text{deflist} := \{\text{def}\}^+$

$\text{bracket} := \text{'begin'} \mid \text{'hide'} (\text{namelist}) \mid \text{'export'} (\text{namelist})$

$\text{env} := \text{bracket deflist 'end'} \mid \text{env}_1 \{ \text{'sees'} \mid \text{'in'} \mid \text{'with'} \} \text{env}_2$

$\text{whereexpr} := \text{expr env} \mid \text{expr 'where' env} \mid \text{'let' env 'for' expr}$
 $= \text{where}(\text{expr}, \text{env})$

The definitions produced by compound environments

The defs produced by various compound envs, each within the containing environment V. The defs produced by an environment A are added to those of the containing V that don't conflict with those of A.

(A sees B)

produces the defs of A; each def in A sees the env A in B in V; each nrdef in A sees B in V.

(A in B)

produces the definitions of A plus the defs of B that do not conflict. The defs of A see the env A in B in V; those of B see B in V. The nrdefs of A see B in V; those of B see V.

(A with B)

produces the defs of both A and B, where each def can see (A with B) in V; each nrdef in A or B can see V.

Examples of where expressions

Assume a containing environment V.

e begin Def f1=e1 Def f2=e2 end sees begin Def f3=e3 end

e sees f1 and f2 (in V); e1 and e2 and e3 see {f1, f2} in {f3} in V. (The "in"s are significant only if, say, f1=f3. otherwise, eg, seeing {f1,f2} in {f3} is seeing {f1,f2,f3}.)

e where hide (f1) nrdef f1=e1 def f2=e2 end in begin nrdef f3=e3 end

e sees {f2} in {f3} in V; e1 sees {f3} in V; e2 sees {f1,f2} in {f3} in V; e3 sees V.

Problem: given an environment V for the expression e and a subexpression e1 of e what environment gets associated with e1? (I haven't dealt with patterns and conditions and lambda expressions)

Given V for e and

- 1) e = paren(e1). Then V for e1
- 2) e = seq(...e1...). Then V for e1
- 3) e = comb(..e1..). Then V for e1
- 4) e = where(e1, B). Then (B in V) for e1
- 5) e = where(expr, B) and B = bracket deflist end and def f≡e1 is in deflist. Then (B in V) for e1.
- 6) e = where(expr, B) and B = bracket deflist end and nrdef f≡e1 is in

deflist. Then V for $e1$.

The rest of these rules are restatements of the rules for forming compound environments and what the defns in them can see:

7) $e = \text{where}(\text{expr}, B \text{ sees } C) \text{ and } \text{def } f \equiv e1 \text{ is in } B$.

Then $(B \text{ in } C \text{ in } V) \text{ for } e1$.

8) $e = \text{where}(\text{expr}, B \text{ sees } C) \text{ and } \text{nrdef } f \equiv e1 \text{ is in } B$.

Then $(C \text{ in } V) \text{ for } e1$.

9) $e = \text{where}(\text{expr}, B \text{ in } C) \text{ and } \text{def } f \equiv e1 \text{ is in } B$.

Then $(B \text{ in } C \text{ in } V) \text{ for } e1$.

10) $e = \text{where}(\text{expr}, B \text{ in } C) \text{ and } \text{nrdef } f \equiv e1 \text{ is in } B$.

Then $(C \text{ in } V) \text{ for } e1$.

11) $e = \text{where}(\text{expr}, B \text{ with } C) \text{ and } \text{def } f \equiv e1 \text{ is in } B$.

Then $((B \text{ with } C) \text{ in } V) \text{ for } e1$.

12) $e = \text{where}(\text{expr}, B \text{ with } C) \text{ and } \text{nrdef } f \equiv e1 \text{ is in } B$.

Then $((B \text{ with } C) \text{ in } V) \text{ for } e1$.

Lemma: in is associative

$(A \text{ in } B) \text{ in } C = A \text{ in } (B \text{ in } C)$

Proof sketch:

let d be a defn in the left env, then d is in the first of $A \ B \ C$ that it occurs in.

Case 1: d in A , then a fn in d sees its first definition in $A \ B \ C$ or V .

Case 1.1, in A . On the right then fn see the same defn, hence d belongs to env on right and defines the same function (d is rec in this case).

Case 1.2 in B , then fn sees the same defn in B in $(B \text{ in } C)$ on the right, hence d belongs and has same meaning.

Case 1.3 in C , then fn not defined in A or B , hence it sees the same defn

in C on the right as on the left, hence d belongs, same meaning.
Case 1.4 in V, similar and ok.

Case 2: d in B; then a fn in d sees its first defn in B C or V.

Case 2.1 in B, then fn sees same defn on right, hence d belongs to rt env

Case 2.2 in C, etc.

Thus if d belongs to left env it belongs to right and has the same meaning. A similar argument shows that if d belongs to the right env, then it belongs to the left and has the same meaning.

Lemma: sees is not associative

$(A \text{ sees } B) \text{ sees } C \neq A \text{ sees } (B \text{ sees } C)$

Proof sketch:

both sides produce the definitions of A. but each definition in

$(A \text{ sees } B) \text{ sees } C$

sees defns in both B and C; and each defn in

$A \text{ sees } (B \text{ sees } C)$

cannot see defns in C.

Lemma: with is associative

no proof