CS412/CS413

Introduction to Compilers

Tim Teitelbaum

Lecture 16:
Attribute Grammars and Natural Semantics

Semantic Analysis

- This lecture:
  - Example of OAG plan construction
  - Natural semantics and attribute grammars
    - Restrictions on inference rules to allow “proof discovery” by attribution
    - Intuition for interpreting type inference rules so you can implement a type checker with visitors

Example

- Productions

  S → E

  E → E + E

  E → NUM

  E → ID

  E → **let** ID **=** E **in** E

- Sample sentence

  let x **=** 1 **in let** y **=** x+1 **in** x+y

- Attributes

  Inherited: E.env

  Synthesized: S.value, E.value, NUM.value, ID.name

Example, cont.

$S \rightarrow E$

E.env = EmptyEnvironment()

S.value = E.value

$E_0 \rightarrow E_1 + E_2$

E_1.env = E_0.env

E_2.env = E_0.env

E_0.value = E_1.value + E_2.value

$E \rightarrow$ NUM

E.value = NUM.value

$E \rightarrow$ ID

E.value = Lookup(ID.name, E.env)

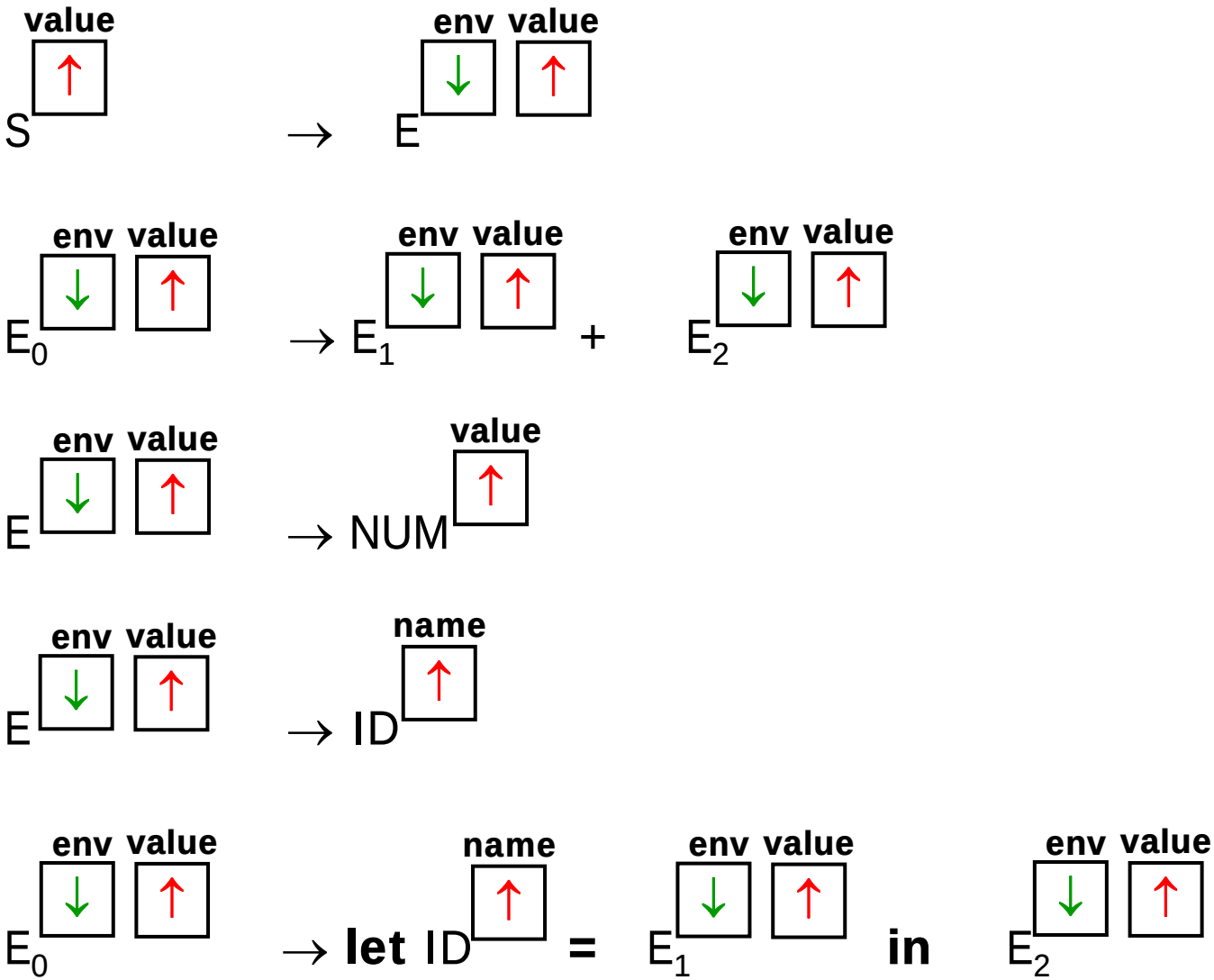
$E_0 \rightarrow$ **let** ID **=** E_1 **in** E_2

E_1.env = E_0.env

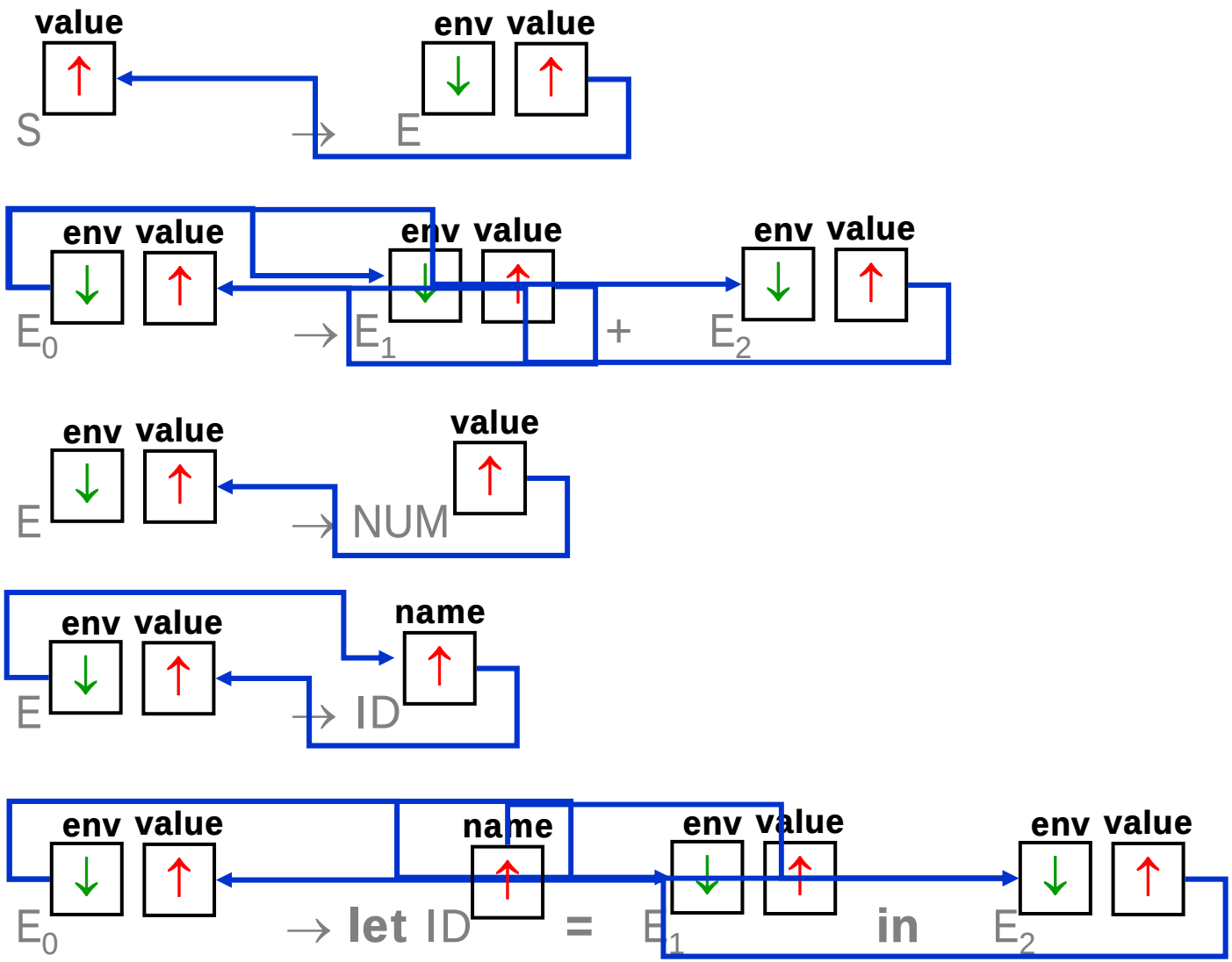
E_2.env = Insert(ID.name, E_1.value, E_0.env)

E_0.value = E_2.value

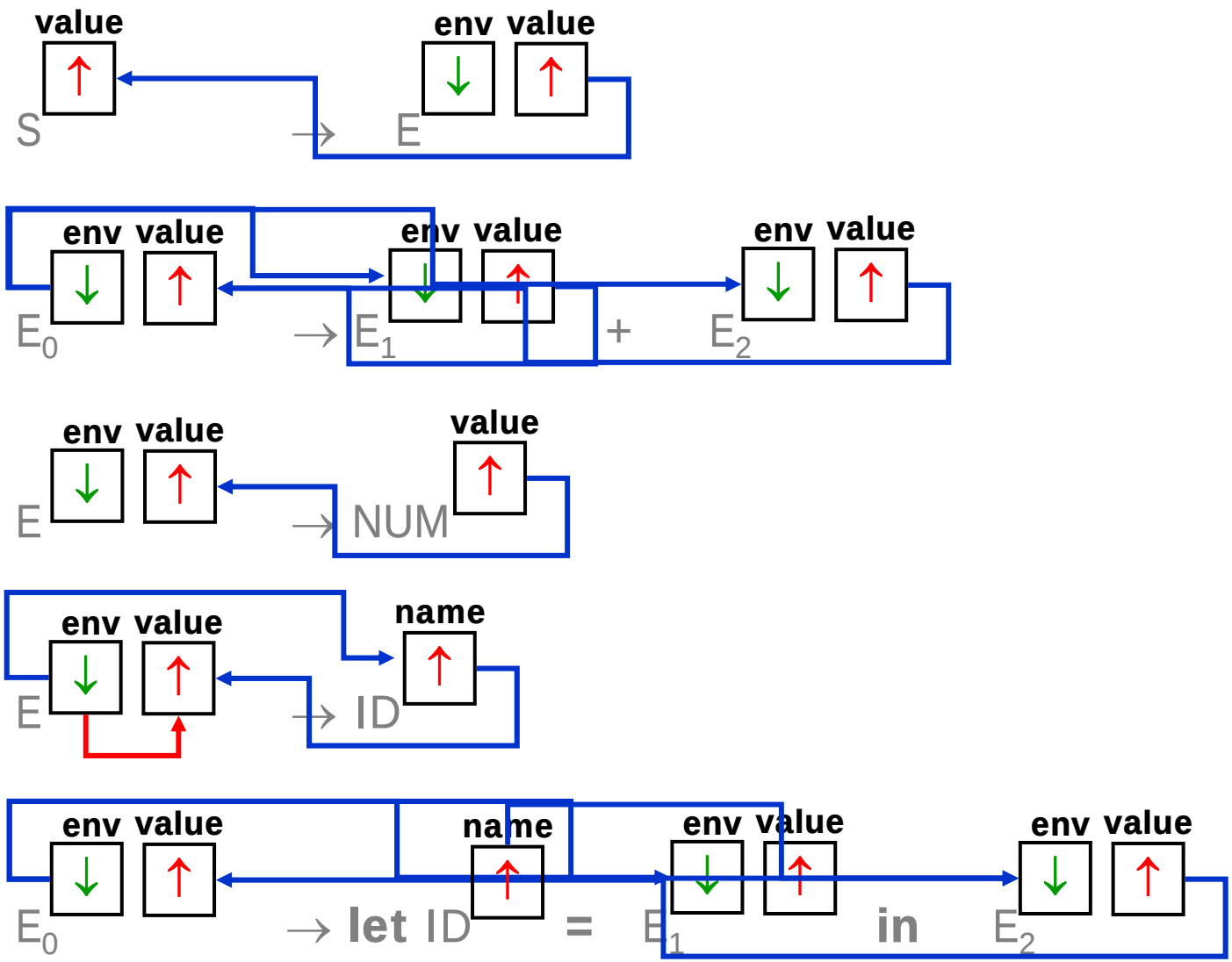
Attribute Occurrences



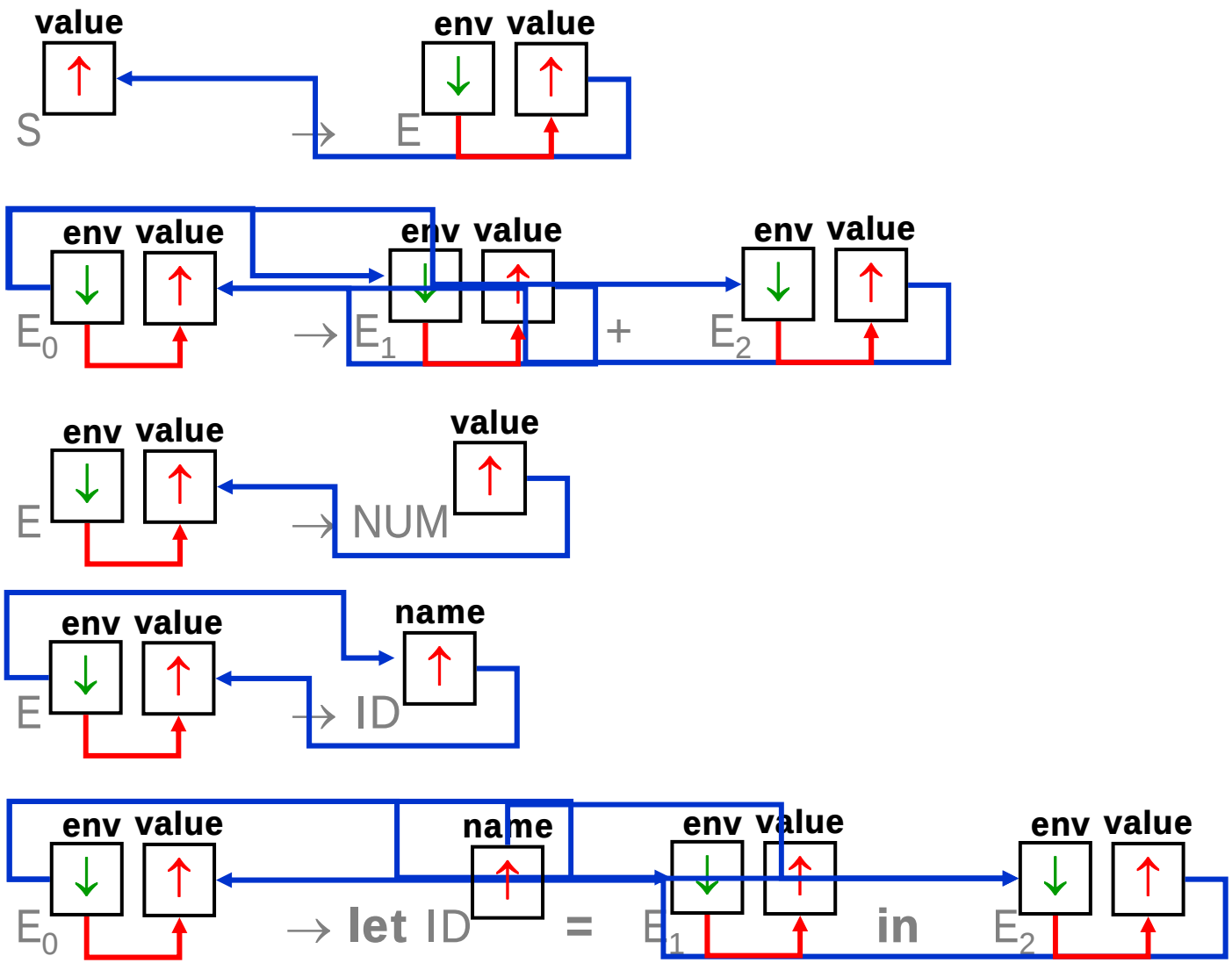
Direct Dependence Graphs



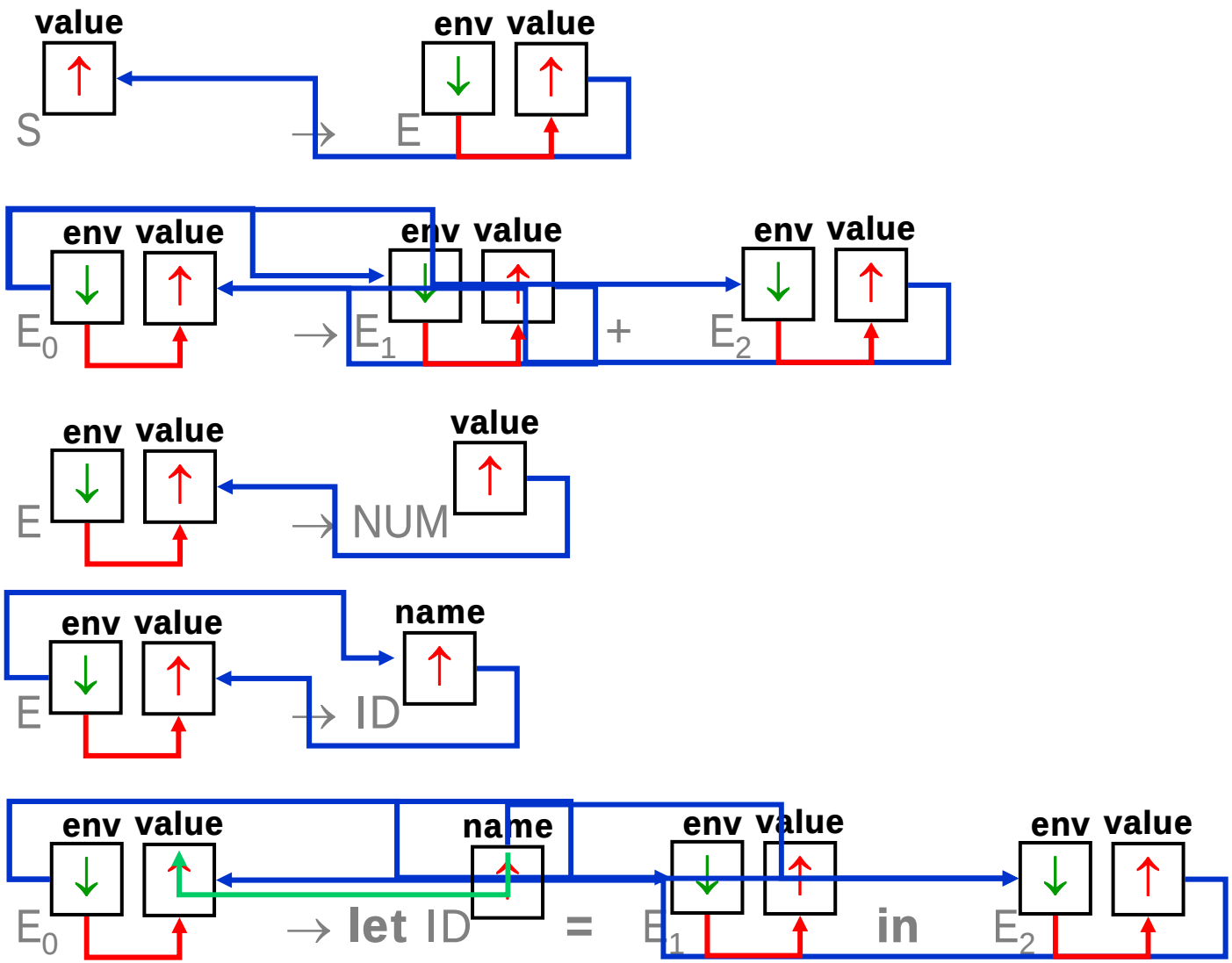
Step 1: Transitive Closures



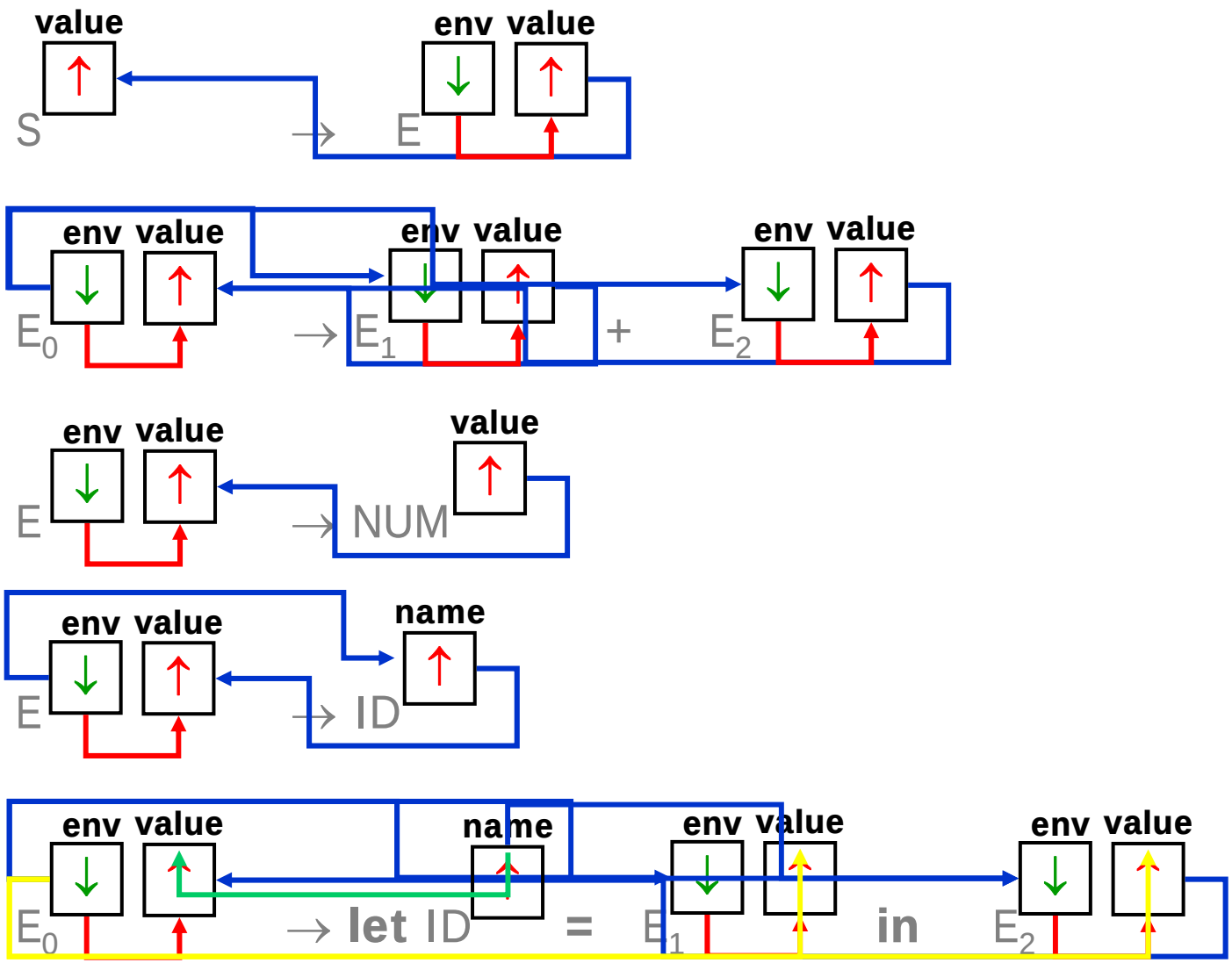
Step 1: Transitive Closures



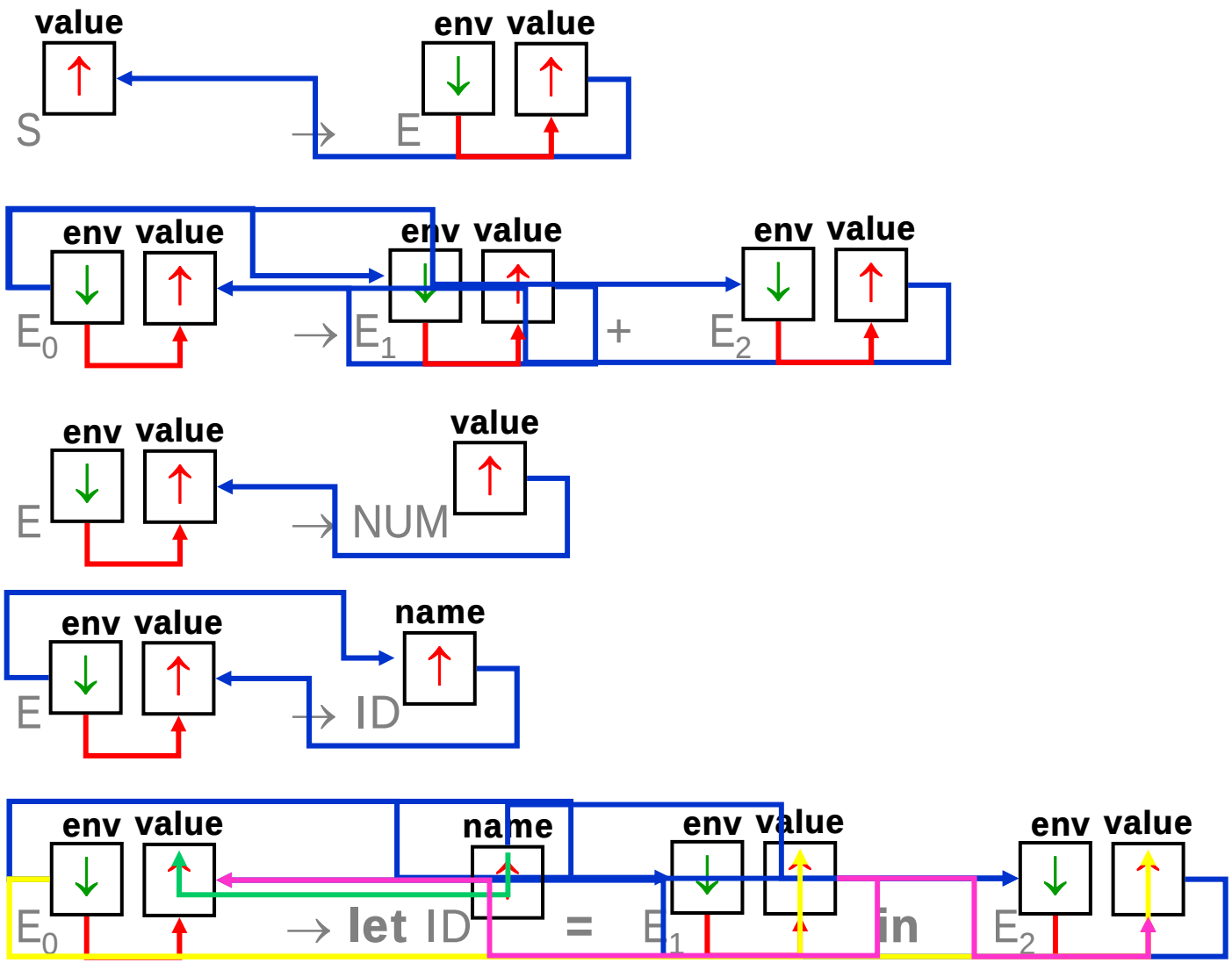
Step 1: Transitive Closures



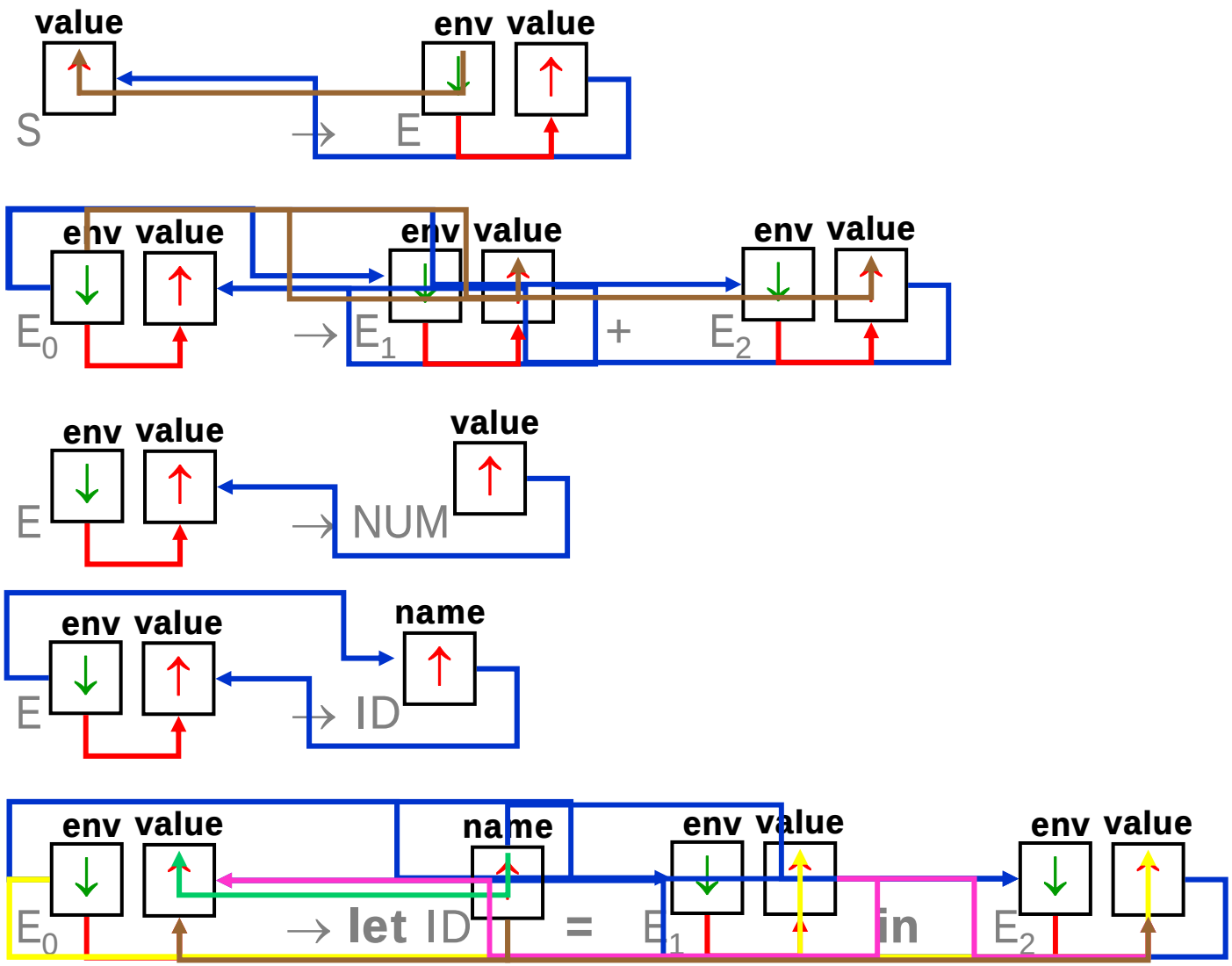
Step 1: Transitive Closures



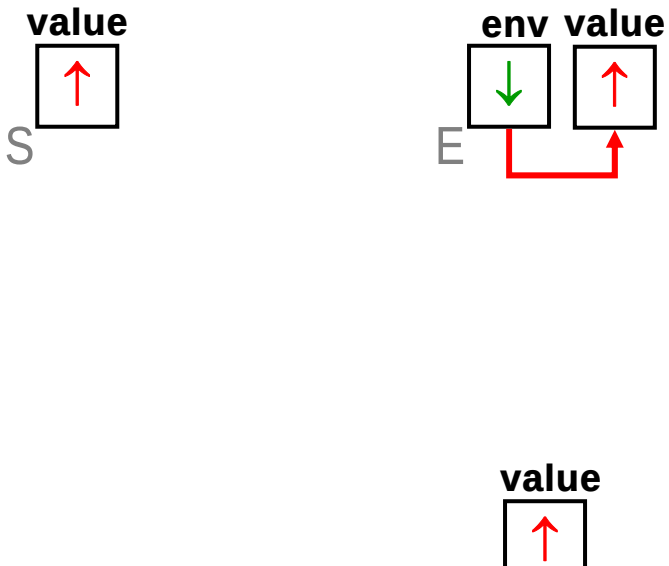
Step 1: Transitive Closures



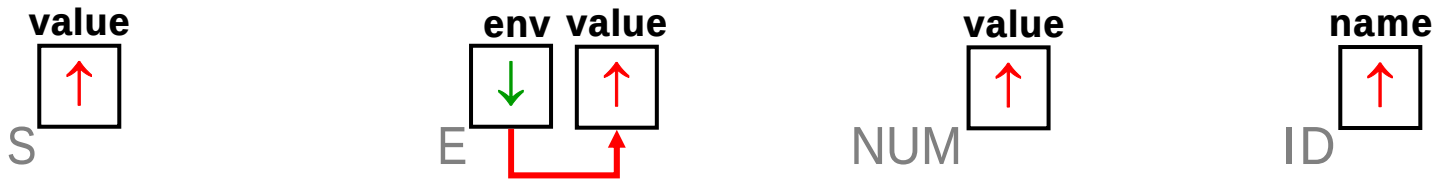
Step 1: Transitive Closures



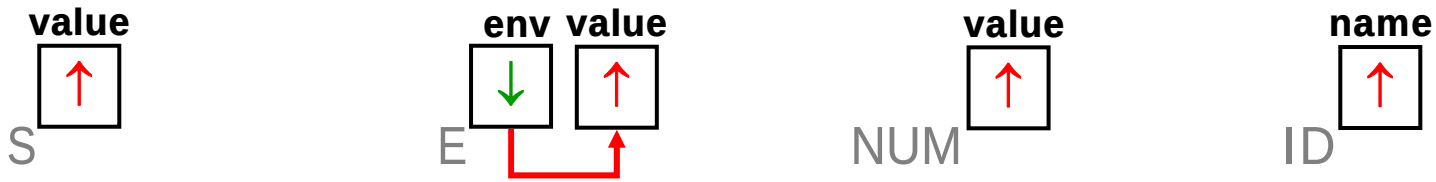
Step 1: DS(X) for each X



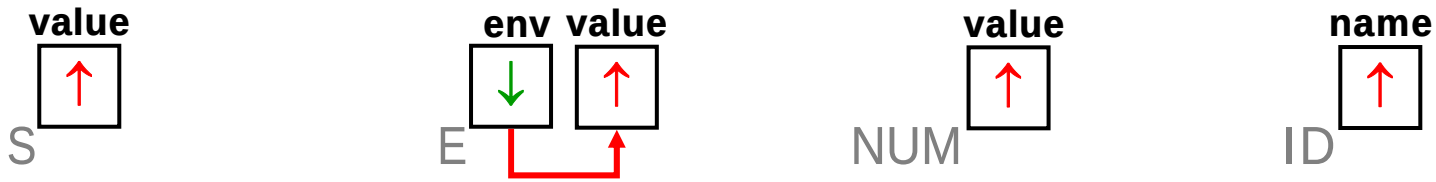
Step 1: DS(X) for each X



Step 2: Partitions for each X

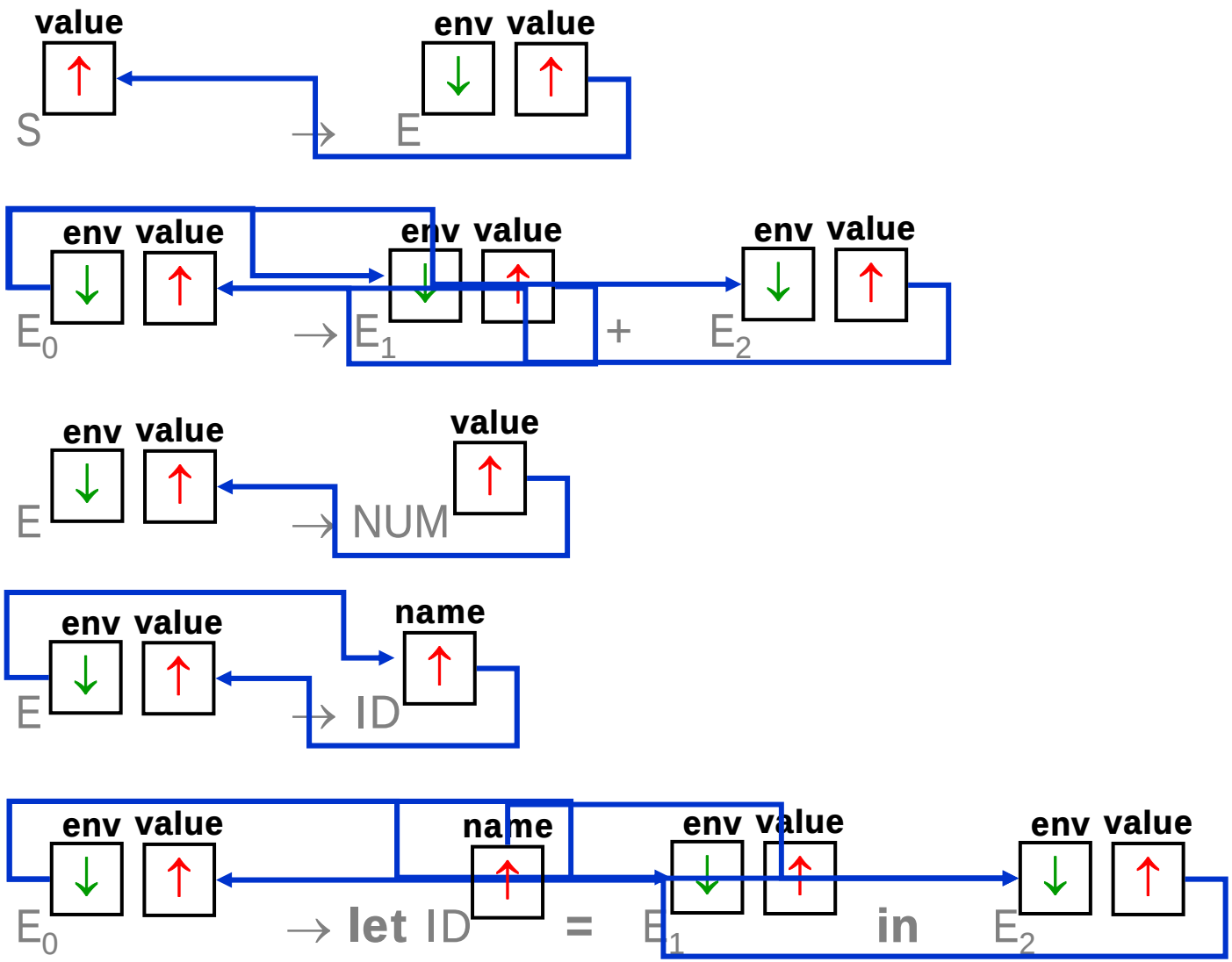


Step 2: Partitions for each X

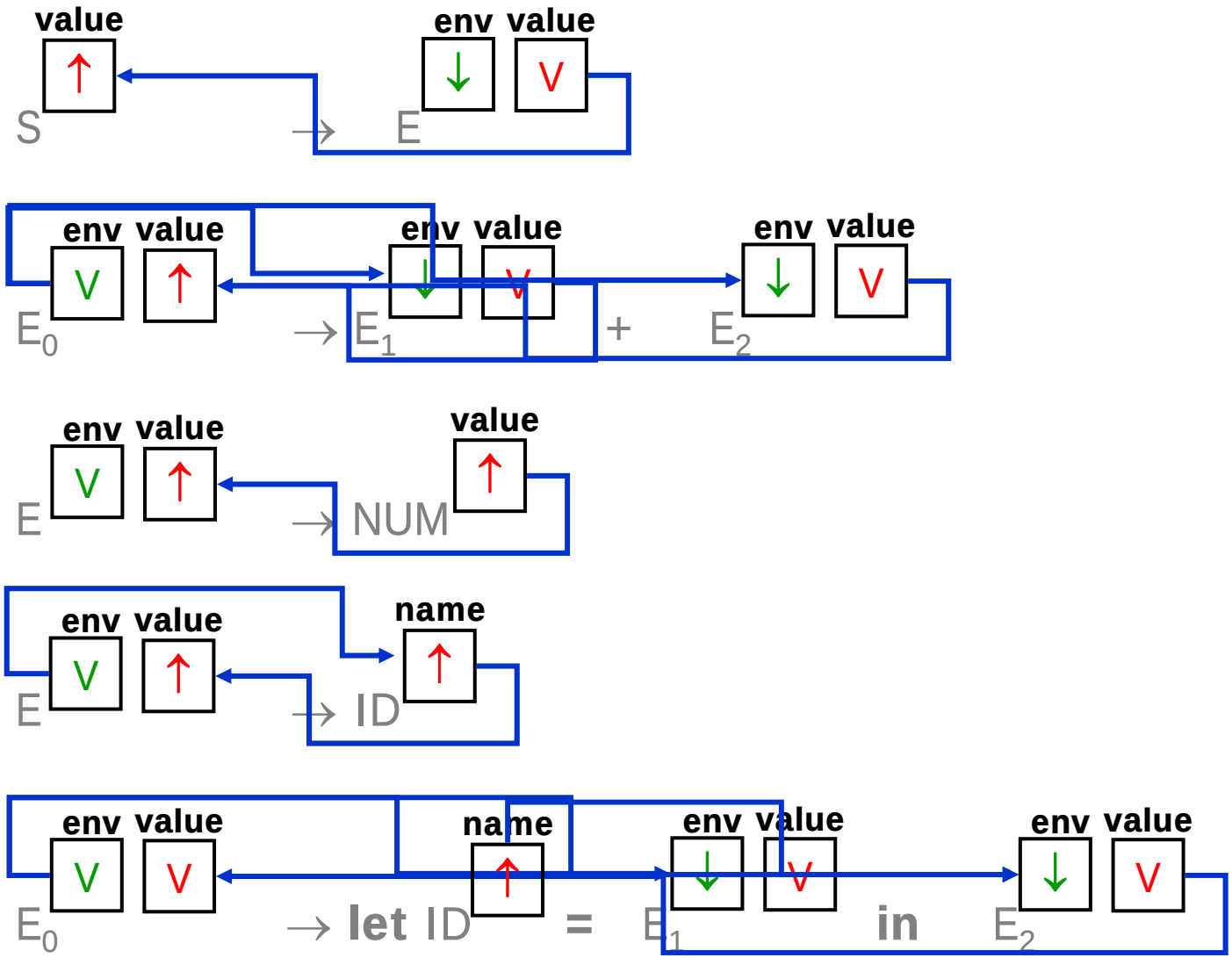


{{}{value}} {{env}{value}} {{}{value}} {{}{name}}

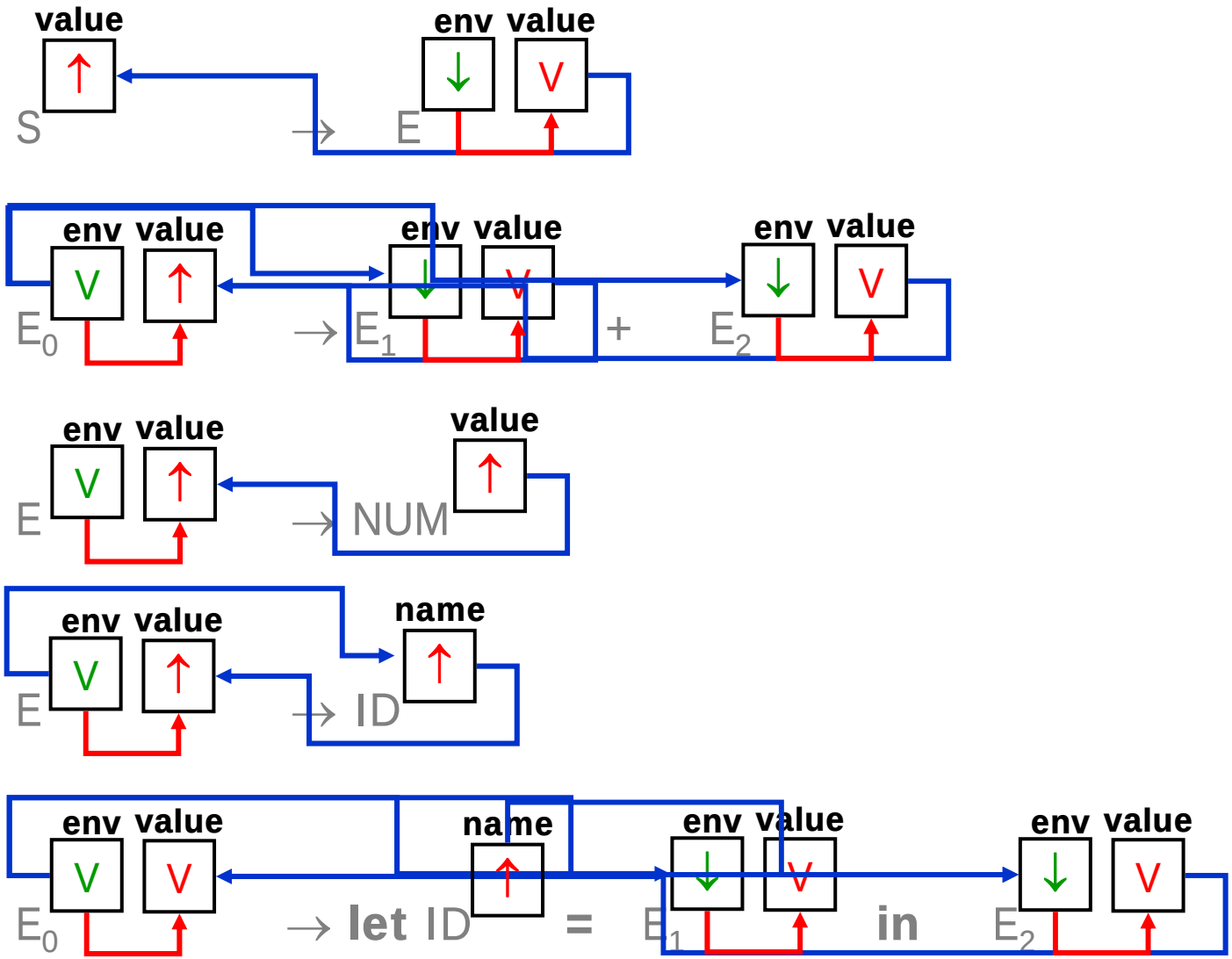
Step 3: Plan Construction (1)



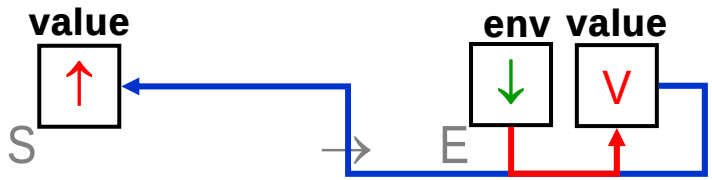
Step 3: Plan Construction (2)



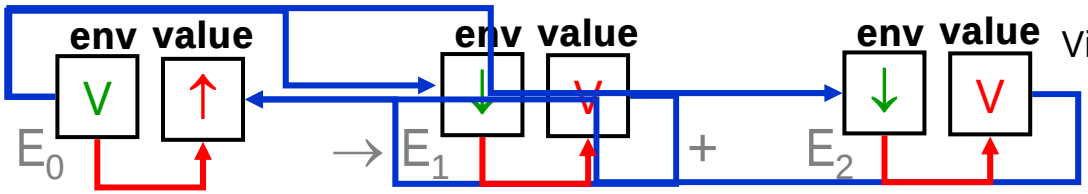
Step 3: Plan Construction (3)



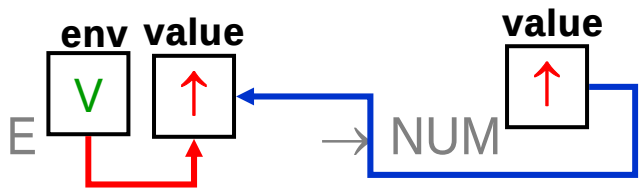
Step 3: Plan Construction (4)



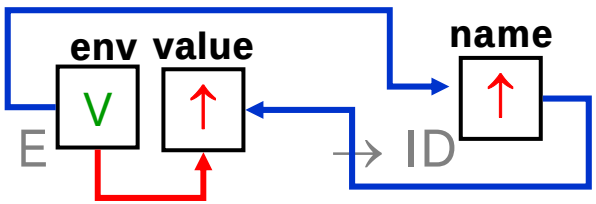
Eval(E.v);Visit(E);Eval(S.value)



Visit(E_0);Eval(E_1.env);Eval(E_2.env);
Visit(E_1);Visit(E_2);Eval(E_0.value)

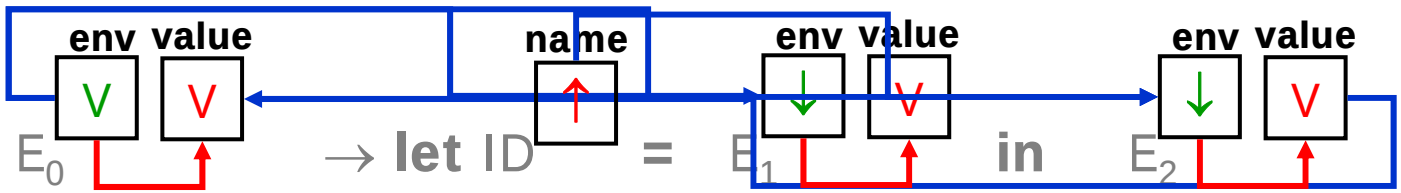


Visit(E);Eval(E.value)

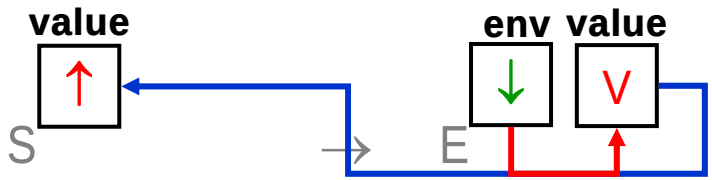


Visit(E);Eval(E.value)

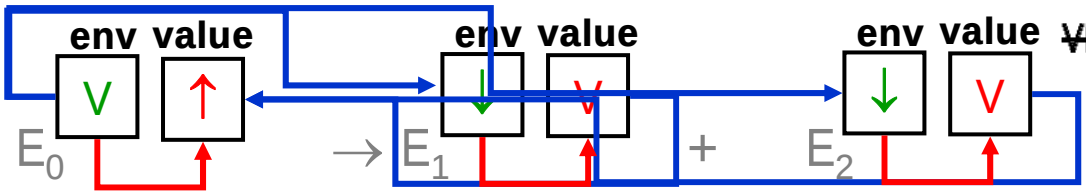
Visit(E_0);Eval(E_1.env);Visit(E_1);Eval(E_2.env);
Visit(E_2);Eval(E_0.value)



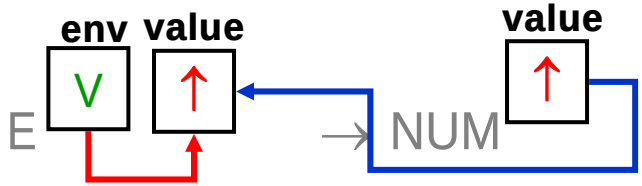
Step 3: Plan Construction (5)



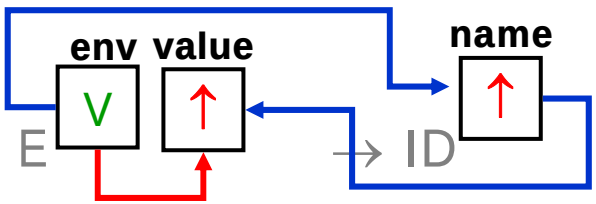
Eval(E.v);VIsIt(E);Eval(S.value);VIsIt(S)



~~VIsIt(E0);~~Eval(E1.env);Eval(E2.env);
VIsIt(E1);VIsIt(E2);Eval(E0.value);VIsIt(E0)

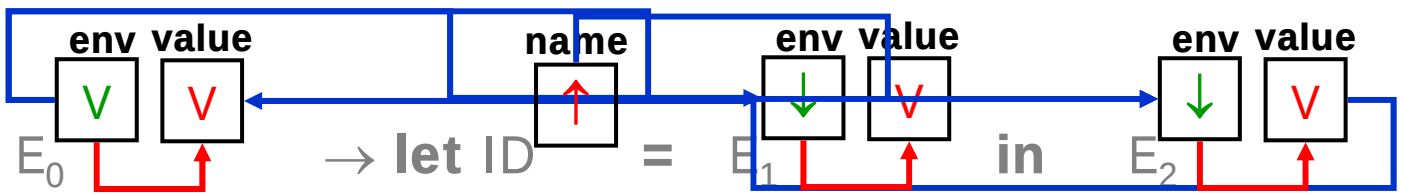


~~VIsIt(E);~~Eval(E.value);VIsIt(E)

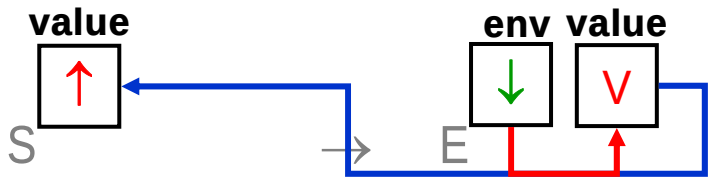


~~VIsIt(E);~~Eval(E.value);VIsIt(E)

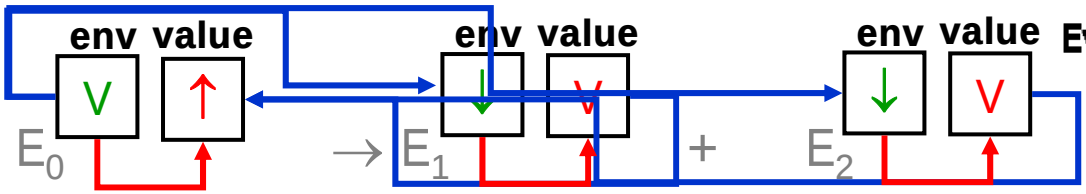
~~VIsIt(E0);~~Eval(E1.env);VIsIt(E1);Eval(E2.env);
VIsIt(E2);Eval(E0.value);VIsIt(E)



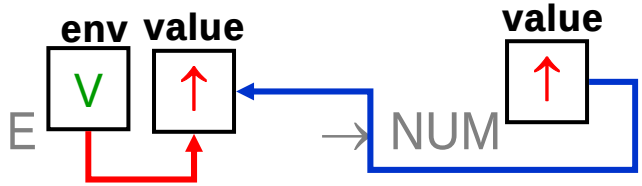
Step 3: Plan Construction (6)



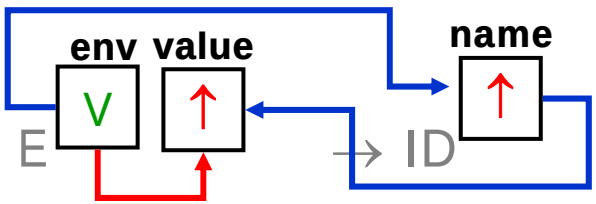
Eval(E.v);VIsIt(E);Eval(S.value);VIsIt(S)



Eval(E_1.env);Eval(E_2.env);VIsIt(E_1);
VIsIt(E_2);Eval(E_0.value);VIsIt(E_0)

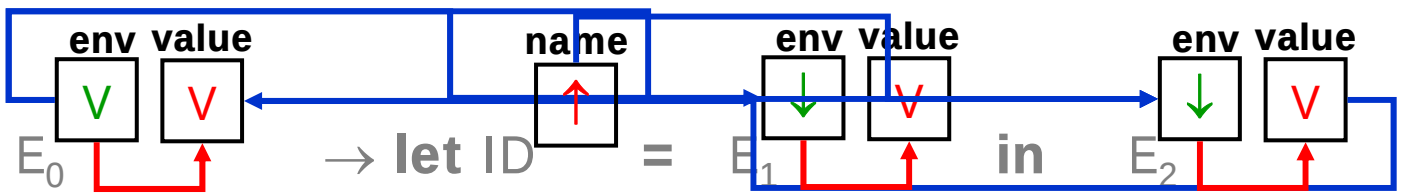


Eval(E.value);VIsIt(E)



Eval(E.value);VIsIt(E)

Eval(E_1.env);VIsIt(E_1);Eval(E_2.env);
VIsIt(E_2);Eval(E_0.value);VIsIt(E)



Natural Semantics / Attribute Grammars

- A type inference system is an example of a natural semantics system
- Show how to interpret (some) natural semantics systems as attribute grammars

Evaluation Example

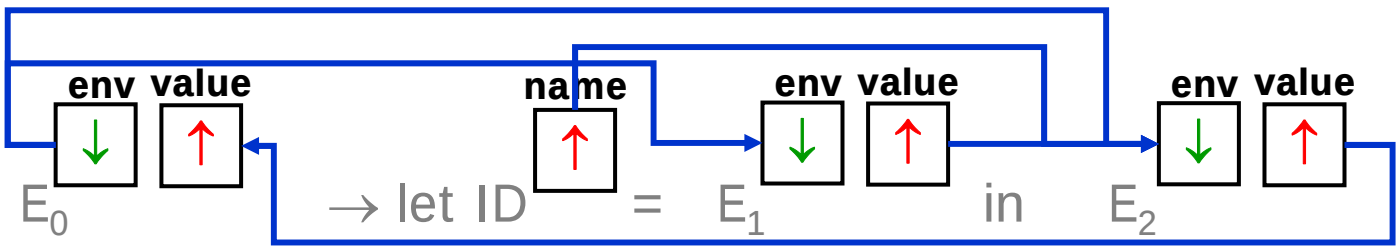
$$\frac{\vdash \text{ID}: \text{name} \quad \text{env} \vdash E_1: \text{value}_1 \quad \text{Insert}(\text{name}, \text{value}_1, \text{env}) \vdash E_2: \text{value}_0}{\text{env} \vdash \textbf{let}\ \text{ID} = E_1\ \textbf{in}\ E_2: \text{value}_0}$$

$E_0 \rightarrow \textbf{let}\ \text{ID} = E_1\ \textbf{in}\ E_2$

$E_1.\text{env} = E_0.\text{env}$

$E_2.\text{env} = \text{Insert}(\text{ID.name}, E_1.\text{value}, E_0.\text{env})$

$E_0.\text{value} = E_2.\text{value}$



Evaluation Example

$$\frac{env \vdash E_1 : value_1 \quad env \vdash E_2 : value_2}{env \vdash E_1 \ \mathbf{+} \ E_2 : value_1 + value_2}$$

$$\frac{\vdash NUM : value}{\vdash NumExpr(NUM) : value}$$

$$\frac{\vdash ID : name}{Insert(name,value,env) \vdash IdExpr(ID) : value}$$

$$\frac{\vdash ID : name \quad env \vdash E_1 : value_1 \quad Insert(name,value_1,env) \vdash E_2 : value_0}{env \vdash \mathbf{let}\ ID\ \mathbf{=}\ E_1\ \mathbf{in}\ E_2 : value_0}$$

Type Checking Example

$$\frac{env \vdash E_1 : Int() \quad env \vdash E_2 : Int()}{env \vdash E_1 + E_2 : Int()}$$

$$\frac{env \vdash E_1 : Int() \quad env \vdash E_2 : Real()}{env \vdash E_1 + E_2 : Real()}$$

$$\frac{env \vdash E_1 : Real() \quad env \vdash E_2 : Int()}{env \vdash E_1 + E_2 : Real()}$$

$$\frac{env \vdash E_1 : Real() \quad env \vdash E_2 : Real()}{env \vdash E_1 + E_2 : Real()}$$

Inference Rule Schema

$$\frac{H_1 \vdash S_1 : T_1 \quad \ldots \quad H_n \vdash S_n : T_n}{H_0 \vdash S_0 : T_0}$$

- H_i, T_i, and S_i are all terms (possibly) with variables
- H_i are called inherited positions
- T_i are called synthesized positions
- S_i are (abstract) syntax patterns
- H_0 , T_1 , ..., T_n are input positions
- T_0 , H_1 , ..., H_n are output positions
- In general, instantiation of rule requires unification

Restrictions on Inference Rules

1. *No missing definitions:* all variables occurring in output positions must also occur in input positions
2. *No missing rule:* the must be an applicable rule for each (abstract) syntax construct
3. *Determinism:* only one applicable rule for each (abstract) syntax construct
4. *No constraint:* input positions are just variables
5. *No link:* no common variables between input positions
6. *No dynamic rules:* premise's syntax constructs are strictly sub-constructs of consequent's syntax constructs

Sample Restriction Violations

1. $$\frac{e \mid\!- S_1 : a}{e \mid\!- p(S_1,S_2) : \mathbf{b}}$$

3. $$\frac{e \mid\!- S_1 : a}{e \mid\!- p(S_1,S_2) : f(a)}$$

3'. $$\frac{e \mid\!- S_2 : a}{e \mid\!- p(S_1,S_2) : f(a)}$$

4. $$\frac{e \mid\!- S_1 : \mathbf{f(a)} \qquad e \mid\!- S_2 : b}{e \mid\!- p(S_1,S_2) : b}$$

5. $$\frac{e \mid\!- S_1 : \mathbf{a} \qquad e \mid\!- S_2 : \mathbf{a}}{e \mid\!- p(S_1,S_2) : f(a)}$$

6. $$\frac{e \mid\!- S_1 : a \qquad e \mid\!- \mathbf{p(S_1,S_2)} : a}{e \mid\!- p(S_1,S_2) : a}$$

Make Implicit Unifications Explicit

4. $$\frac{e \vdash S_1 : \mathbf{f(a)} \qquad e \vdash S_2 : b}{e \vdash p(S_1,S_2) : b}$$

5. $$\frac{e \vdash S_1 : \mathbf{a} \qquad e \vdash S_2 : \mathbf{a}}{e \vdash p(S_1,S_2) : f(a)}$$

4'. $$\frac{e \vdash S_1 : a' \qquad unify(a',f(a)) \qquad e \vdash S_2 : b}{e \vdash p(S_1,S_2) : b}$$

5'. $$\frac{e \vdash S_1 : a \qquad e \vdash S_2 : a' \qquad unify(a,a')}{e \vdash p(S_1,S_2) : f(a)}$$

Summary

- If a natural semantic system satisfies the preceding restrictions, it corresponds to an attribute grammar
- In this case, it can be evaluated by formal attribution, or by the corresponding tree traversal conducted by visitors