Declarations and Blocks

Declaration: 
(\(ty, loc) \downarrow loc[i \leftarrow i]\)  
for \(i \in \text{rng}(loc) \cup \text{dom}(mem)\)

Command (as Block):
\( (C, (mem, loc)) \downarrow C (mem', loc') \)
\( (D, (mem, loc)) \downarrow_{\text{Block}} (mem', loc') \)

Dec. Seq.:
\( (D, loc) \downarrow loc' (Blk, (mem, loc')) \downarrow (mem'', loc'') \)
\( (D, Block, (mem, loc)) \downarrow (mem'', loc'') \)

Block: 
\[ [(Blk, (mem, loc)) \downarrow C (mem', loc)] \]

While Command

While-false:
\( (B, (mem, loc)) \downarrow false \)
\( (while \ B \ do \ C \ od, (mem, loc)) \downarrow (mem, loc) \)

While-true:
\( (B, (mem, loc)) \downarrow true (Blk, (mem, loc)) \downarrow (mem', loc') \)
\( (while \ B \ do \ Blk \ od, (mem', loc)) \downarrow (mem'', loc'') \)

Adding Procedures to SIMPL1

I \in Identifiers  
N \in Numerals  
E ::= N | I | E + E | E * E | E - E  
B ::= true | false | B & B | B or B | not B | E < E | E = E  
C ::= skip | C | {Block} | I ::= E | I aliases I  
\( if \ B \ then \ Block \ else \ Block \ fi \) while B do Block od  
yt ::= int  
D ::= ty I  
Block ::= C | D; Block

If Then Else Command

If-true:
\( (B, (mem, loc)) \downarrow true (Blk, (mem, loc)) \downarrow (mem', loc') \)
\( (if \ B \ then \ Blk \ else \ Blk' \ fi, (mem, loc)) \downarrow (mem', loc) \)

If-false:
\( (B, (mem, loc)) \downarrow false (Blk', (mem, loc)) \downarrow (mem', loc') \)
\( (if \ B \ then \ Blk \ else \ Blk' \ fi, (mem, loc)) \downarrow (mem', loc) \)

Scoped Constructs for SIMPL1

I \in Identifiers  
N \in Numerals  
E ::= N | I | E + E | E * E | E - E  
B ::= true | false | B & B | B or B | not B | E < E | E = E  
C ::= skip | C | {Block} | I ::= E | I aliases I  
\( if \ B \ then \ Block \ else \ Block \ fi \) while B do Block od  
yt ::= int  
D ::= ty I  
Block ::= C | D; Block

Slides based in part on previous lectures by Grigore Roșu  
Elsa L Gunter  
2112 Siebel Center, UIUC  
egunter@illinois.edu  
http://courses.engr.illinois.edu/cs422/sp2017  
February 21, 2017
Semantic Objects to Support Procedures

- To evaluate a procedure declaration
  - What do we need? Procedure body
  - What do we produce? Binding of procedure name to body code
- To evaluate a procedure call
  - How do we interpret procedure body?
  - Answer to all these depends on static scope versus dynamic scope

Static versus Dynamic Scope

- **Scope** is the “region” of a program in which a variable name is bound to a specific variable location (a binding)
- **Static scope** of declaration of $x$:
  - “region” = nodes in the parse tree that are
    - within a block of the declaration,
    - and not within an inner block declaring $x$
  - An instance of $x$ is bound to the declaration of $x$ of nearest outer scope
- **Dynamic scope** of declaration of $x$:
  - “region” = steps of execution where declaration is “visible”, ie
    - the declaration was made in a block that execution is still in
    - execution has not proceeded past another declaration of the same variable that is in a block the current execution is in and after the given declaration

Static Scope

- Scope of top int $x$ declaration in green

Dynamic Scope

- Green arrow indicates temporally which declaration binds which $x$

Program Execution (time)

Procedures with Dynamic Scope

- To evaluate a procedure declaration
  - What do we need? Procedure body, location of identifiers at dec
  - What do produce? Procedure body, location of identifiers at dec
- To evaluate a procedure call
  - How do we interpret procedure body?
    - $\text{mem(loc}(f)) = (\text{Block}, \text{loc'}) \cup (\text{mem'}, \text{loc'})$
    - $(\text{call } f) \cup (\text{mem'}, \text{loc})$
      - if $\text{loc}(f)$ exists, $\text{mem(loc}(f))$ a Block

Procedures with Static Scope

- To evaluate a procedure declaration
  - What do we need? Procedure body, location of identifiers at dec
  - What do produce? Binding of procedure name to body code and location mapping
    - Add Block × (Identifiers → Locations) to our Values
- To evaluate a procedure call
  - How do we interpret procedure body?
    - $\text{mem(loc}(f)) = (\text{Block}, \text{loc'}) \cup (\text{mem'}, \text{loc'})$
    - $(\text{call } f) \cup (\text{mem'}, \text{loc})$
      - if $\text{loc}(f)$ exists, $\text{mem(loc}(f))$ a Block
Adding Parameterized Procedures to SIMPL1

- To evaluate a procedure declaration
  - What do we need? Procedure body and parameters
  - What do produce? Binding of procedure name to parameters and body code
    - Add Params × Block to our Values

Procedure Declaration:

\[
\text{procedure } I \text{ Params } \{ \text{Block} \}, \text{(mem, loc)} \downarrow (\text{mem}' , \text{loc}') \\
(\text{Block}, \text{(mem, loc)}) \downarrow (\text{mem}' , \text{loc}') \\
\text{call } I \text{Args} \\
\text{if loc(I) exists , mem(loc(I)) a (Block, loc)}
\]

Parameterized Procedures with Dynamic Scope

- To evaluate a procedure declaration
  - What do we need? Procedure body, params, ... Language Design (CS 422)

Procedure Declaration:

\[
\text{procedure } I \text{ Params } \{ \text{Block} \}, \text{(mem, loc)} \downarrow (\text{mem}[i \leftarrow (\text{Params}.\text{Block}], \text{loc}[l \leftarrow i]) \\
\text{for } i \notin \text{rng(loc)} \cup \text{dom(mem)}
\]

- To evaluate a procedure call
  - How do we interpret procedure body?
    - mem(loc(I)) = (Params, Block)
    - (\text{Params}, \text{Args}), (\text{mem, loc}) \downarrow (\text{mem}' , \text{loc}')
    - (\text{Block}, (\text{mem, loc}')) \downarrow (\text{mem}' , \text{loc}')
    - (call I(), (\text{mem, loc}')) \downarrow (\text{mem}', \text{loc})
    - if loc(I) exists , mem(loc(I)) a (Block, loc)

Parameterized Procedures with Static Scope

- To evaluate a procedure call
  - How do we interpret procedure body?
    - mem(loc(I)) = (Params, Block, loc')
    - (\text{Params}, \text{Args}), (\text{mem, loc}) \downarrow (\text{mem}' , \text{loc}')
    - (\text{Block}, (\text{mem, loc}')) \downarrow (\text{mem}' , \text{loc}')
    - (\text{call I()} \downarrow (\text{mem}', \text{loc})
    - if loc(I) exists , mem(loc(I)) a Block

Parameterized Procedures with Static Scope

- To evaluate a procedure declaration
  - What do we need? Procedure body, params, dec location mapping
  - Add Params × Block × (Identifiers ⇒ Locations) to our Values

Procedure Declaration:

\[
\text{(procedure } I \text{ Params } \{ \text{Block} \}, \text{(mem, loc)}) \\
\text{mem}(i \leftarrow (\text{Params}.\text{Block}.\text{loc}), \text{loc}(l \leftarrow i)) \\
\text{for } i \notin \text{rng(loc)} \cup \text{dom(mem)}
\]