Functions

COP-3402 Systems Software

Paul Gazzillo

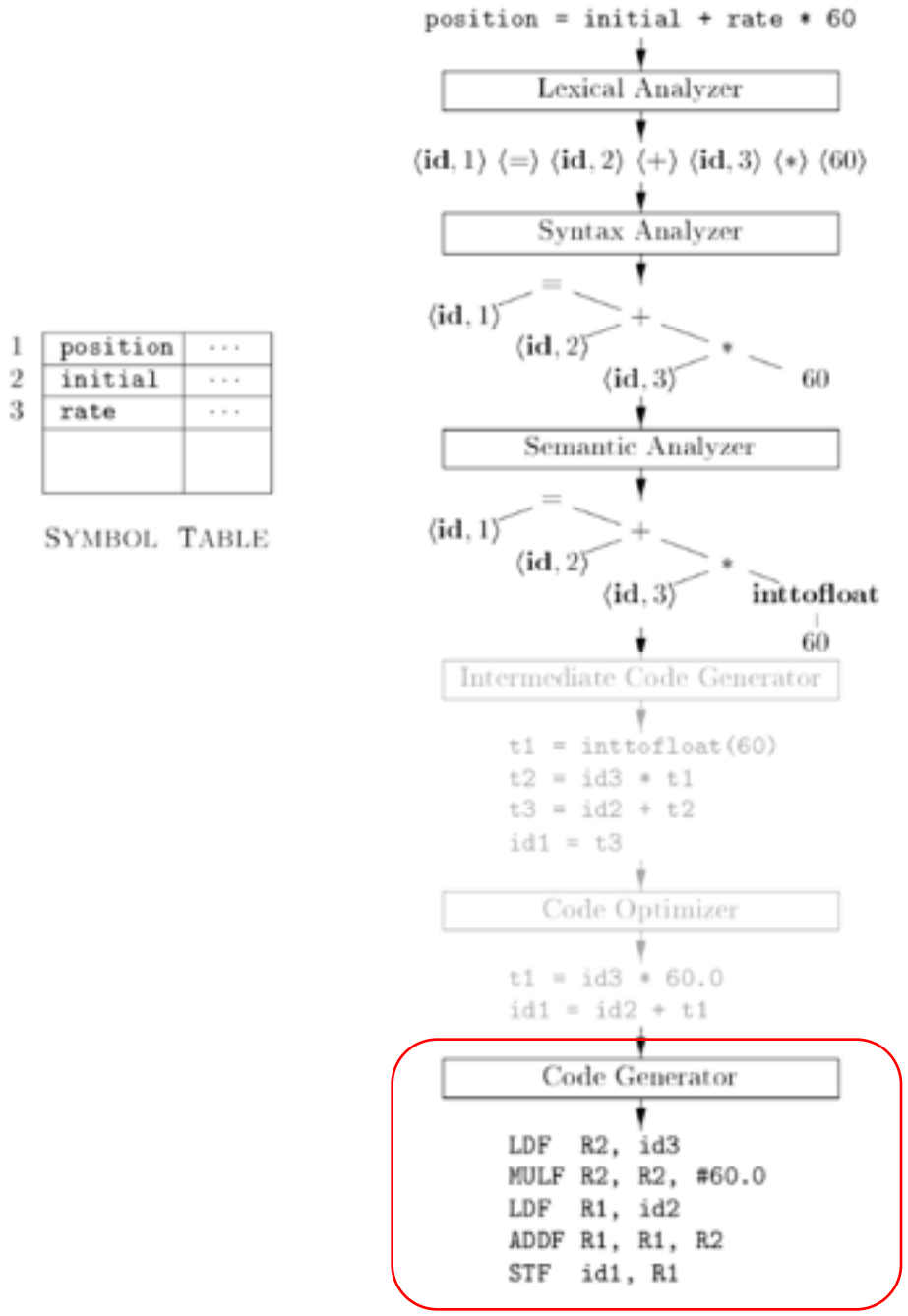


Figure 1.7: Translation of an assignment statement

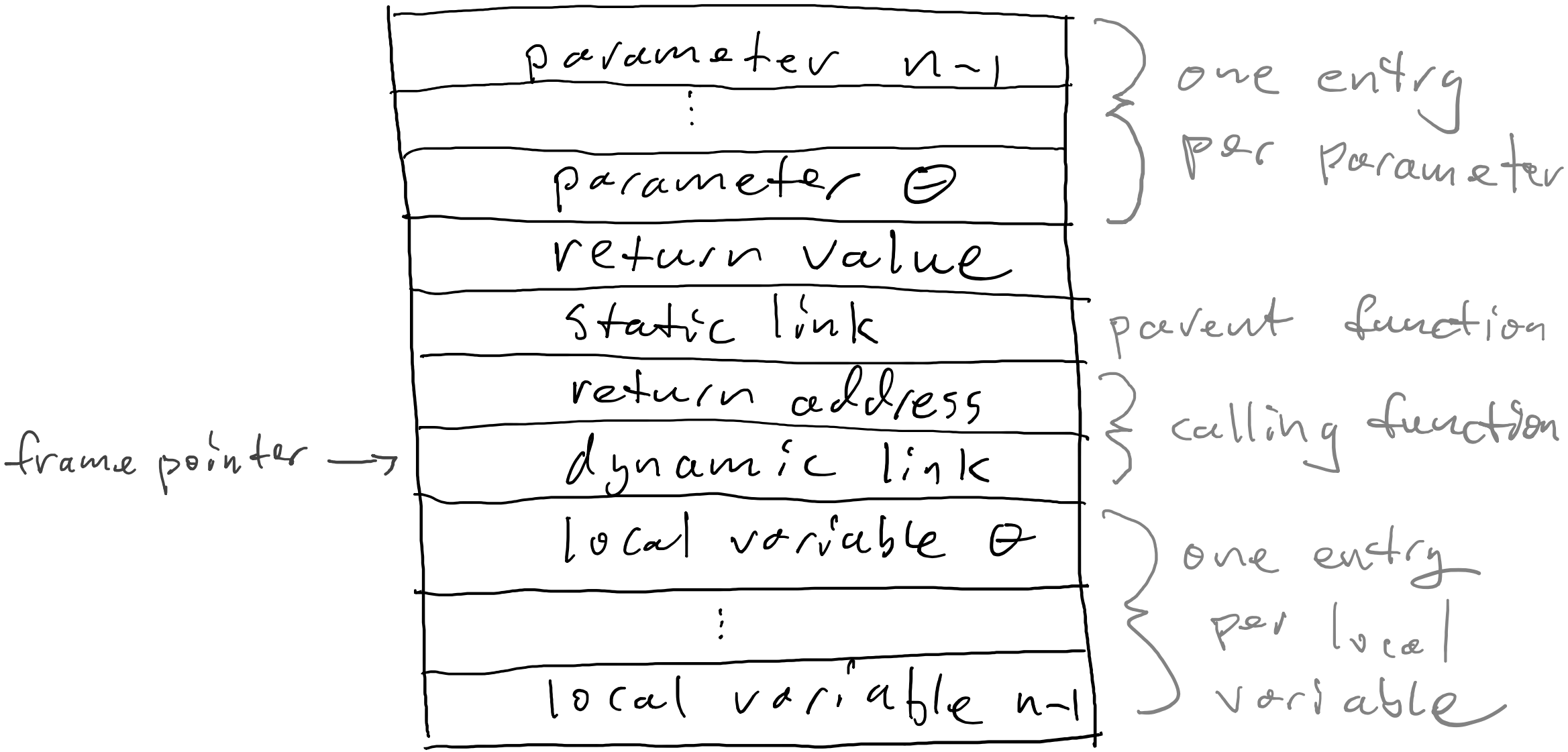
Core Idea: Simulate Functions by Saving State

- Functions encapsulate code
  - Name represents the code
  - Abstract away implementation into input and output
- Functions not processed by hardware
  - Some processors provide support
- Compiler ”simulates” functions with machine code
  - Function state stored on stack
  - Parameters and return values passed on stack (and registers)
  - Calling a function “freezes” the current function until return

Activation Records (Stack Frames)

- Function state stored on stack in *activation records*
  - Called a stack frame in its implementation
- Stores a function's entire state
  - So it can "freeze" and restore a running function
- Function state
  - Parameters
  - Return value
  - Pointer to parent function (static scoping)
  - Return address (calling function)
  - Pointer to calling function
  - Local variables

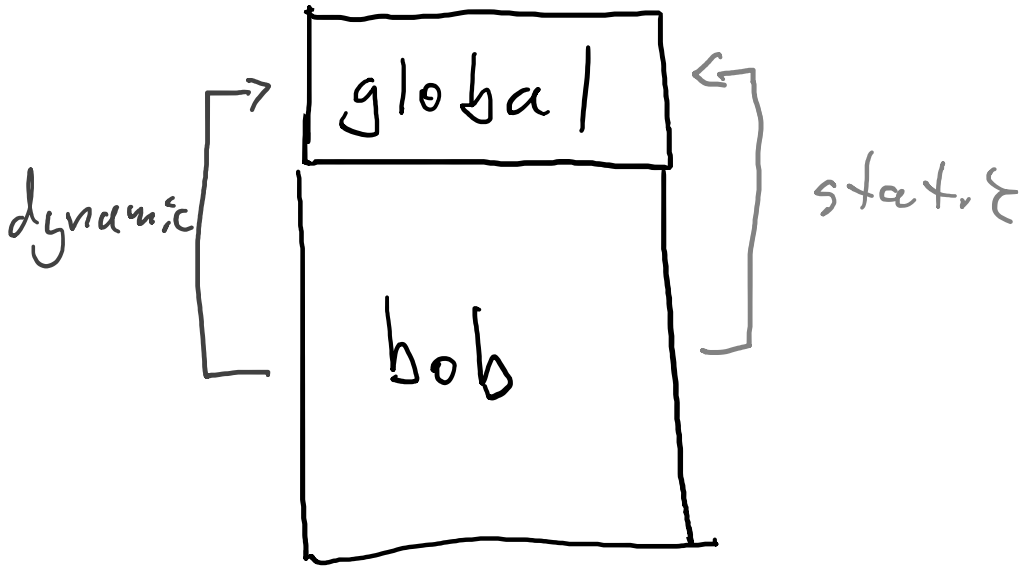
Stack Frame Layout



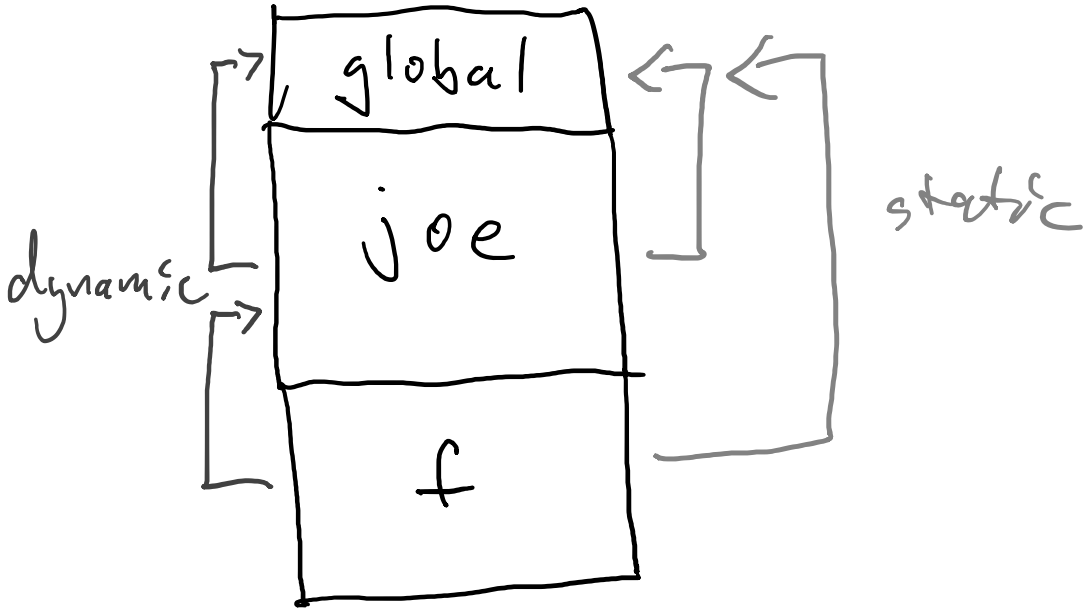
Dynamic Link: Restoring a Frozen Function

- Recall: stack frame stores all function state
- How do we save the state?
- Function call creates *new* function state, i.e., stack frame
  - *Push* a new stack frame
- How do we restore the frozen function?
- Save a pointer to the old stack frame, i.e., *dynamic link*
- Save the instruction pointer, i.e., the *return address*

```
function bob(x : int) : int
  begin
    write x
  end
bob(2)
```




```
function f(x : int)
  write x
function joe(x : int)
  f(x)
joe(2)
```



Static Link: Accessing Static Scopes

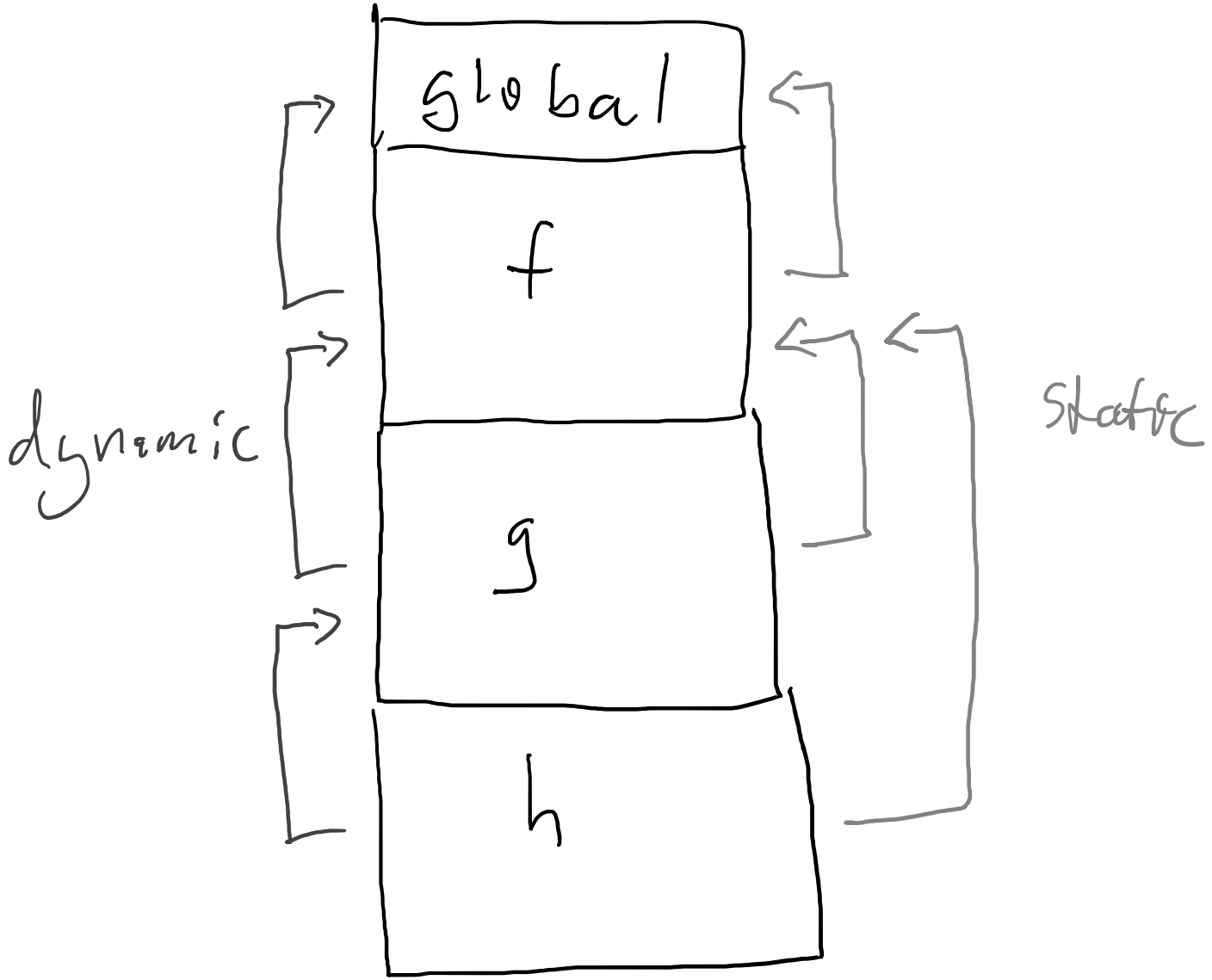
- Recall: nested functions may access symbols in any parent scope
- Recall: function state stored in stack frame
- How do we find a parent function's stack frame?
- We record it in the current stack frame, i.e., the *static link*
- How do we access *any* ancestor function's locals?
- We follow as many static links as needed
- We know at compile time exactly how many links to follow

```
function f(f_var : int) : bool
 function g(g_var : int) : bool
  return h(f_var + g_var)

 function h(h_var : int) : bool
  return h_var < 10

 g(4)

f(3)
```



Constructing Stack Frames

- At compile-time, we know
  - How many parameters there are
  - How many local variables there are
  - Where the function call was made
    - How to get the return address and dynamic link
  - Which function is the parent scope
    - Where to get find the static link from
  - Which scope the symbol is defined in
    - How many static links to follow
- All of this information is stored in the stack frame
- *We generate code that creates the stack frame*