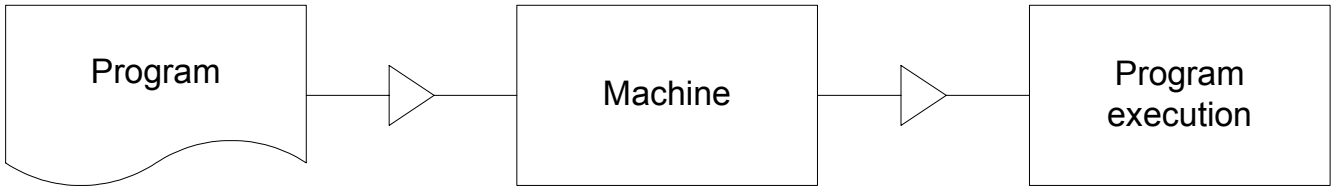
Syntax/semantics



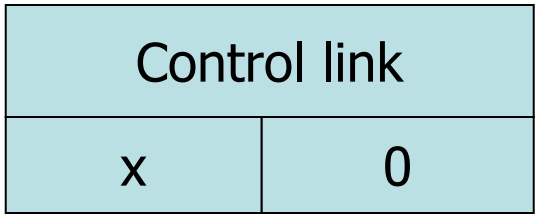
Syntax

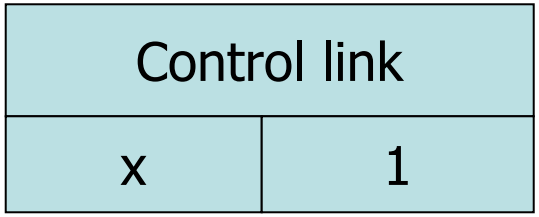
Semantics

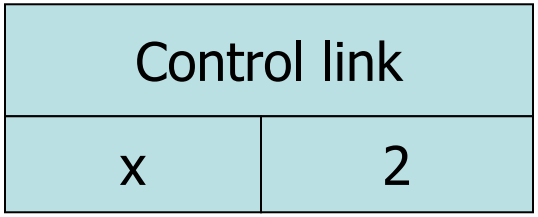
- Differences and similarities between
  - Parse trees
  - (Abstract) syntax trees
  - Metamodels

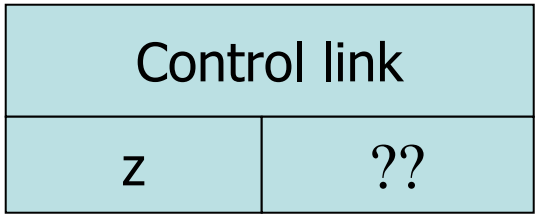
Static/dynamic scoping

```
{
   int x=0;
   {                  -- block 1
      int x=1;
   };
   {                  -- block 2
      int x=2;
   };
   {                  -- block 3
      int z = x;
   }
}
```









As is
z = 0

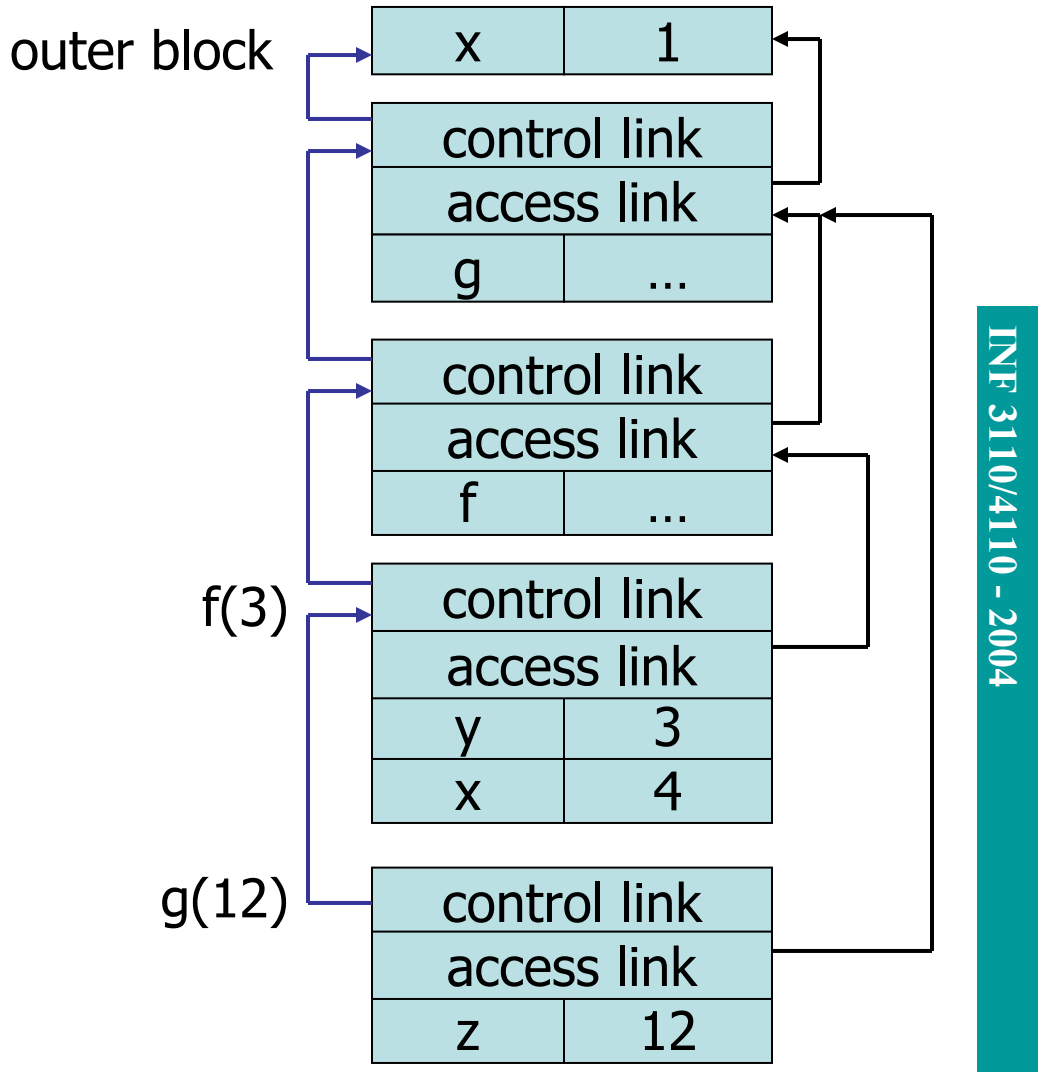
block 1 executes block 3
Static scope: z = 0
Dynamic scope: z = 1

block 2 executes block 3
Static scope: z = 0
Dynamic scope: z = 2

INF 3110/4110 - 2004

Static scope with access links

```
int x=1;
function g(z) = x+z;
function f(y) =
    { int x = y+1;
      return g(y*x) };
f(3);
```

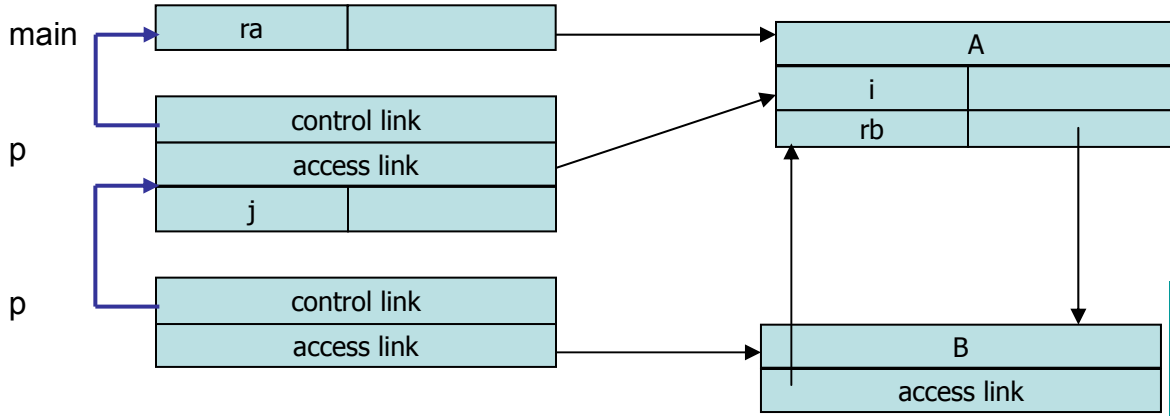


Use access link to find global variable:

- Access link is always set to frame of closest enclosing lexical block
- For function body, this is block that contains function declaration

Access link for Java-like languages

```
class Program {
   public static void
   main(String[] args) {
      A ra=new A(); ra.p();
   };
};
class A{
   int i=1;
   class B {
      void p(){ i=2; };
   };
   B rb = new B();
   void p(){
      int j=i;
      rb.p();
   };
};
```

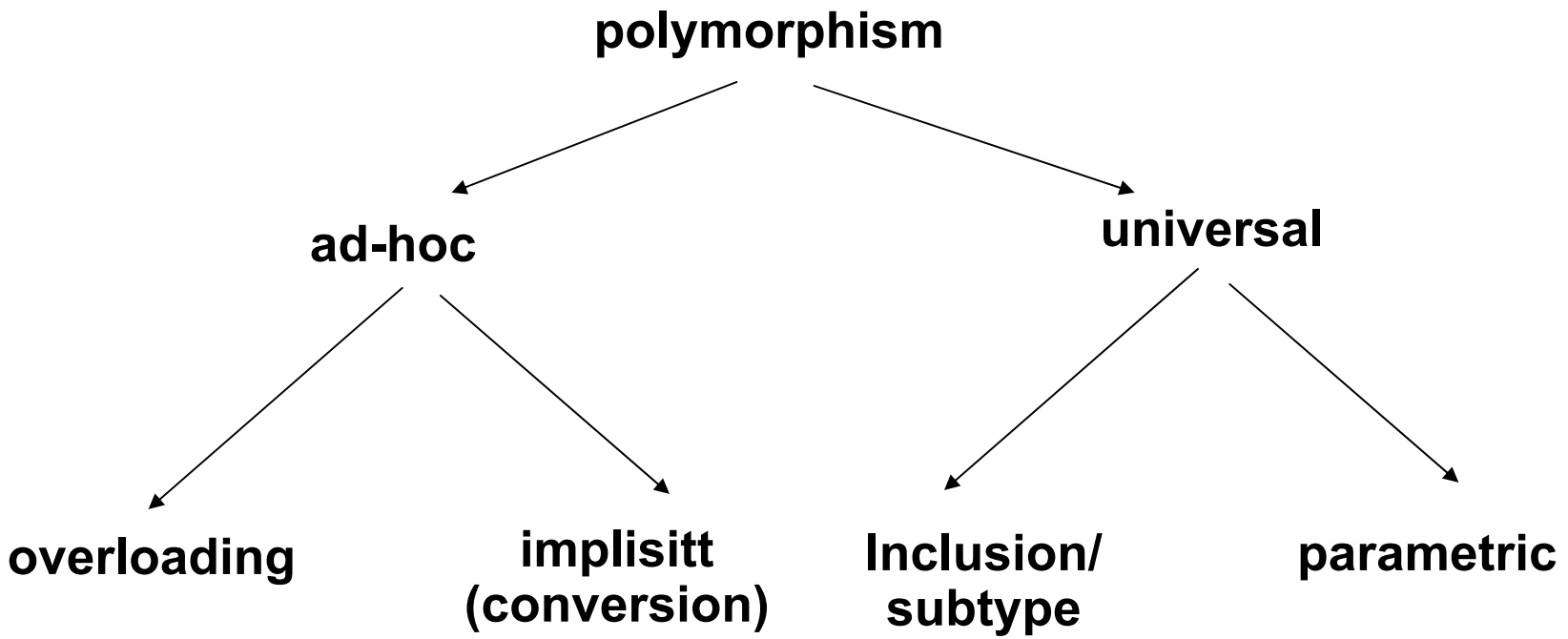


- Access links for method activations may be objects
- Objects may also have access links

Parameter passing

- Pass-by-value
- Pass-by-reference
- Pass-by-name

Classification of polymorphism



Type checking

- Static/compile-time
- Dynamic/run-time

- Assignments
- Method/function calls
  - Reference typing/virtual methods
  - Parameters (number and types)

- Overriding/redefinition of virtual procedures versus overloading
- Covariance - contravariance