Domination-Based Scoping
and
Static Single Assignment Languages

M. Anton Ertl
TU Wien
Domination-based Scoping

• Frequent Mistake: Uninitialized Variables

• Initialize variables on definition

• Visible only where initialized

• Visible everywhere where initialized

• ⇒ in scope where initialization dominates

• Can differ from block scoping
  ... depending on control structure
Domination-based scope same as block scope

if (...) {
    int a=...;
    ...
} else {
    int b=...;
    ...
}

while(...) {
    int c=...;
    ...
} else {
    ...
}

block

domination-based

Explains why block scoping works
Domination-based scope larger than block scope

```
  do {
    int d=...;
    ...
  } while (...);

  for (int i=0; i<n; i++)
    if (a[i] == key)
      break;
  ...
```
Domination-based scope smaller than block scope

```c
if (flag)
    goto entry;
{
    int e=...;
entry:
    ... = e;
}
```

```c
switch (...) {
    int f=...;
    case ...: ...=f; break;
    case ...: ...=f; ...  
}
```

block

domination-based
Experiences

• Implemented in Gforth since 1994

• No problems reported by users

• Occasionally useful

Pros and Cons

– Block scoping is well-known and accepted

+ If the language supports jumps into blocks
Static Single Assignment Programming

- Each variable has only one assignment (=definition).

- Facilitates understanding the data flow.

- Also for imperative languages.

- Initialization on definition helps a lot, but

- ... what about control flow joins?
$\phi$-functions in programming language: confusing

```c
unsigned fib(unsigned n)
{
    for (;;) {
        unsigned a=phi(0,b);
        unsigned b=phi(1,a+b);
        unsigned i=phi(n,i-1);
        if (i==0)
            break;
    }
    return a;
}
```
Stack-based languages (Gforth)

: fib { n -- n2 }
    0 1 n BEGIN { a b i }
        i 0 <> WHILE
        b a b + i 1-
    REPEAT
    a ;
Algol-family language without $\phi$-functions

func fib(n)
0,1,n -> start -> a,b,i;
   if i=0 then
      exit;
   b, a+b, i-1 ->
repeat
   a ->
end;
Conditional and compound updates

x=0;
a={1,2,3};
...
if ...
    x=1;
a[x]=4;
...
... x, a;
... x, a;
Experiences

• Works well in Gforth

• Algol-family language implemented in compiler course

• Grammar: statement/sequence/block tradition does not work

• SSA for conditional and compound updates?

• Different declaration syntax for SSA and SMA variables?
Conclusion

- Domination-based scoping
  - ensures that variables are initialized
  - useful for languages with jumps into blocks
  - helps understanding the interaction of scoping and control flow

- Static Single Assignment Programming
  - Makes programs easier to understand
  - Syntax for Algol-family languages interesting

- Paper: http://www.complang.tuwien.ac.at/papers/ert107kps.ps.gz
Expertise

• Code generation:
  Instruction Selection
  Instruction Scheduling
  Register Allocation

• Implementation of efficient interpreters