Lecture 19

Data flow languages. SISAL examples and issues.

Assignment Submission

- Very few **correct** submissions
- Please resubmit using the following commands.

tar -cf <username>.tar <username directory>

mailfile pears@leopard <username>.tar

Assignment

- The following people need not submit again.
  - woelkerling
  - maxwell
  - woodman
  - tatarelli
  - bata
  - bruce
  - sullivan
  - rogowski
  - brooks
  - thong

Data-flow languages.

- There are a number of popular data-flow languages. The most important are probably:
  - VAL
  - SISAL
- This lecture examines their structure and syntax using programming examples.

Side effects

- Functions that modify global variables produce what is known as a side effect.
- Eg.

```
PROCEDURE  GetSq ( X: Real );
BEGIN
        SQ:= X*X;
END GetSq;
```

- Solution: Ban global variables, and reference parameters.

Requrements

- Deducibility: parallelism should be deducible from the program syntax.
- There are two principal requirements:
  - data dependencies can be deduced from the program operations.
  - sequencing constraints must be the same as data dependencies.
- These requirements are met by ensuring locality of effect and freedom from side-effects.

More side effects

- Aliasing: consider the following code....

```
PROCEDURE Rev ( VAR A, B: IntArr );
VAR
    J: INTEGER;
BEGIN
    FOR J := 1 to 10 DO
        B[J] := A[11-J];
    END;
END Rev;
```

- Called as follows:

```
Rev( Q, Q );
```

- Can the assignments proceed concurrently?

Locality of effect.

- No unnecessary far reaching dependencies.
  - No re-use of variable names within a single scope.
  - If a variable is re-used is it dead, or is it live (communicating data)?
- Solution:
  - define limited scopes associated with each variable.
  - limit entry and exit from such scopes.

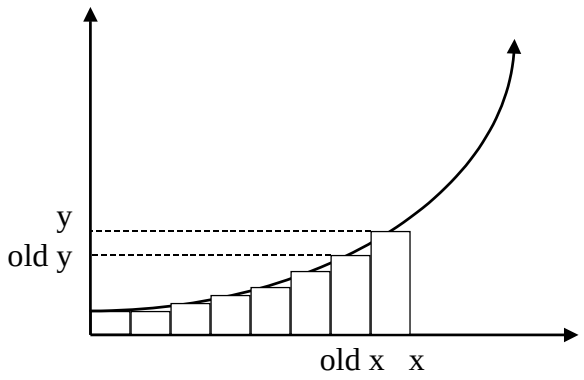
Loops

- The single assignment rule has implications for loop variables. Most loops compute results by re-assigning values to variables.
- Re-binding cannot occur within a scope, but it can occur on the boundary of that scope.
- Single assignment languages implement loops usings special syntax features which allow the compiler to detect when re-bindings occur.

More side effects

- Pointers introduce similar problems as do array indexing.
- The most complete solution is to enforce a completely functional programming model with no pointers or reference parameters.
- Array indexing for modification is handled by creating a new array instance holding the modified value in the appropriate position.

Integration



$$A = \sum (\text{old } y + y) / 2) * 0.02$$
$$= \sum (\text{old } y + y) * 0.01$$

Implications for assignment

- Assignment binds a variable to a value.
- This binding holds force during the scope in which the binding occured, and within this scope no other bindings can name this variable.
- Effect: A variable can appear in the LHS of an assignment statement exactly once in any scope.

SISAL Example

```
function Integral ( returns real )
for initial
   int := 0.0;
   y   := 0.0;
   x   := 0.02;
while x < 1.0 repeat
   int := 0.01 *( old y + y );
   y   := old x * old x;
   x   := old x + 0.02
returns
   value of sum int
end for
end function
```


VAL Example

```
function Integral (returns real )
for
   int: REAL := 0.0;
   y: REAL   := 0.0;
   x: REAL   := 0.02;
   z: REAL   := 0.0;
if x < 1.0 then
   iter
         y := x*x;
         int := int + 0.01*(y+z);
         z:= y;
         x:= x + 0.02;
else  int
endif
endfor
endfun
```