
Designing Semantic Structures

Thinking Like a Compiler Designer

Step 1-- Language Design

- | Start by defining the language cleanly and developing a sound grammar.
- | Good language design leads to good compiler design.
- | Languages should be largely LL(1) so that predictive parsing can be done by humans. Compilers need not be LL(1).

Example-- if-then-else

| `<stmt> ::=` `“if” <boolexpr> “->“`
 `<statements>`
 `[]`
 `<statements>`
 `“fi” “.”`
 `;`

- | This is a possible form for an if-then-else statement in MicroGCL.
- | It is but one form of statement (`<stmt>`).

Step 2. Look at an Example

- | A sample if-then-else statement might be:
- | if $a < b$ -> write a; [] write b; fi;
- | Pick a simple, but representative example to work with. (In some cases one example isn't sufficient to cover the possible options.)

Step 3. Write Output Code

- | By hand, write down the code for your example in the target language.

```
»          LD R0, $a$          if a < b
»          IC R0, $b$
»          JGE L3              ->
»          WRI $a$              write a;
»          JMP L4              []
» LABEL L3
»          WRI $b$              write b;
» LABEL L4                    fi ;
```

Step 4. Analyze

- Find the earliest point at which you can generate each piece of code.

```
»      LD R0, $a$           if a < b
»      IC R0, $b$
»      JGE L3                ->
»      WRI $a$                write a;
»      JMP L4                []
» LABEL L3                   ->
»      WRI $b$                write b;
» LABEL L4                   -> fi ;
```


Step 5. The Parameters

- | Now determine what information each semantic routine needs to initiate generation of the required code.
- | This gives you the input parameters of that routine.
- | The location at which that information is first known gives the output parameters of the routine at that location.

Step 5. continued

- | #iftest needs the operator from the <boolexpr> and a label. It can create the label. Therefore the <boolexpr> must return the operator and pass it to #iftest
- | #elsepart needs the label from #iftest and another label that it can generate
- | #endif needs the label from #elsepart.

Step 5. continued

- | #iftest receives an operator record and returns a label record with a new label.
- | #elsepart receives the label record from #startif and returns a modified label record with a new label.
- | #endif receives the label record returned by #elsepart.

Notes

- | If you have a complex situation, you may need to try more than one example.
- | Your goal is to place semantic routines to cover all cases, but to do so as cleanly as possible.
- | You also want to pass as little information as reasonable.

Notes continued

- | The values returned by a semantic routine are held as local variables in the parsing routine that contains the call to the semantic routine. Alternatively they are stored on a stack.
- | They are never stored in fixed global variables, since the language is likely recursive.