CS362 2011F *Compilers*

Class 37: Translating Loops

Held: Monday, 28 November 2011

Summary: Today we consider issues in the translation of looping structures, particularly Pascal's three main looping constructs.

Related Pages:

- EBoard.

Notes:

- Are there questions on Phase 5?
- EC for Thursday extra on steganography.
- EC for Royce Wolf performance in Convo slot on Thursday.

Overview:

- Status Check.
- While Loops.
- Repeat-Until Loops.
- For Loops.

Background

- Yes, we're still discussing translation.
- We tend to think of translation in terms of processing the parse tree.
- However, in many cases, it is more appropriate to translate as we go, since that tells us about the rule that is being used.
- In addition, the parse tree is generally intended to be a theoretical construct; we generally don't need the tree itself, only some of the attributes (which get discarded soon after they are used).

- Last week, we discussed the translation of conditionals and Boolean expressions.
- We noted that there are two philosophies on the evaluation of Boolean expressions:
  - short-circuit evaluation, which stops as soon as the result is known;
  - strict evaluation, which evaluates all of the parts.
- Pascal chooses strict evaluation (although I can't find it in the report).

Kinds of Loops

- Pascal has three key looping constructions (if you don't count recursion): while loops, repeat-until loops, and for loops.
- For loops repeat their bodies a fixed number of times and have an associated counter.
- While and repeat-until loops repeat their bodies an indefinite number of times.
- While loops may never execute their bodies; repeat-until loops execute their bodies at least once.
- Most languages include some variant of these three kinds of loops, with varying interpretations.

Translating While Loops

- Recall that their are two key parts to a while loop: the test and the body.
- It's helpful to have three extra labels for the translated while loop
  - One will precede the test
  - One will precede the body
  - One will follow the body
- Translate the test
  - If you're doing short-circult evaluation, pass the body label as the true part and the follow label as the false part.
  - Since Pascal does strict evaluation, we'll deal with a result from the translation of the test.
- Translate the body.
- Join together:
  - New label for the test
  - Code for the test
  - Jump-if-zero Result-of-Test End-Loop
  - New label for the body
  - Code for the body
  - Jump test-label
  - New label for the end-of-loop.

Translating Repeat Loops

- Repeat-until loops are similar to while loops, except that the test can be in a different place.
- Another alternative is to do almost exactly the same translation (usually inverting the test) and then precede it with a Jump to the body-label.
  - This works particularly well for C's do-while loops, in which the meaning of the test is the same.

Translating For Loops

- Basic form

```
for counter := lower-bound to upper-bound step increment do
  body
```

- At first glance, For loops seem equally simple, since we can treat a for loop as something like the following while loop:

```
counter := lower_bound;
while (counter <= upper-bound) do
  begin
    body;
    counter := counter + increment;
  end
```

- However, Pascal introduces one subtle difference: If the counter variable is an enumerated type, it may be meaningless to increment beyond upper-bound.
  - You can also have this problem if the upper bound is the largest possible integer.
- What's the solution? Do a test for equality *after* the loop but before the increment.
  - Store lower_bound into counter
  - New label for body
  - Code for body
  - Jump-if-equal counter upper-bound end-label
  - Add increment to counter
  - Jump body-label
  - New label for end of loop
- Note that this impelmentation requires that the body be executed at least once and that the increment bring you exactly to the result.
  - Careful reading of the Pascal specifications will show the same requirements.
- Some other considerations
  - What happens if the counter exceeds the upper bound?
  - How often should the expression for the upper-bound be executed?