Sequence 1.5 – Introduction to the Tiger Language

P. de Oliveira Castro S. Tardieu

The Tiger Language

- Introduced by A. Appel in 1998 his *Modern Compiler Implementation* series.
  - Imperative
  - Typed, with two primitive types: integers and strings
  - Has nested functions

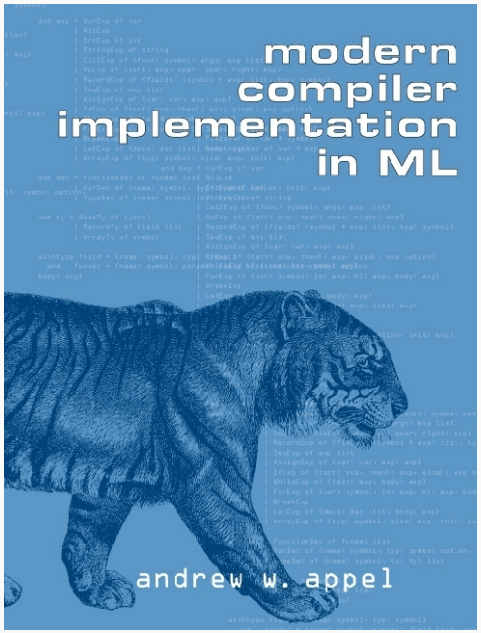

Figure 1: A. Appel, Modern Compiler Implementation

Hello World

```
print("Hello World!")
```

Types

- Three types:
  - `int` signed 32 bits integers from -2^{31} to $2^{31} - 1$.
  - `string` string of ASCII 8 bit characters
  - `void` return type of a function or block that returns nothing

Let-in-end blocks and variables

```
let
    /* Declarations */
    var thermostat : int := -17
in
    /* Expressions */
    thermostat := thermostat + 1;
    print_int(thermostat);
    print("\n")
end
```

Variables

```
let
  var a : int := 0
  var b := 1
  var c := "hello"
```

Functions declarations

```
let
    var thermostat : int := 17
    /* return type int */
    function get_temperature () : int =
      thermostat
    /* return type void */
    function increment (delta : int) =
      thermostat := thermostat + delta
    function print_temperature () =
      print_int(thermostat)
in
    ...
end
```

Factorial

```
let
  function fact(n : int) : int =
    if 1 < n then n * fact(n - 1) else 1
in
    print_int(fact(7));
    print("\n")
end
```

Nested functions

```
let
    function fact(n : int): int =
        let
            function f(n: int, acc: int) : int =
                if n <= 1 then acc else f(n - 1, acc * n)
        in
            f(n, 1)
        end
in
    print_int(fact(7))
end
```

Mutually recursive functions

```
let
    function odd(n : int)  : int =
        if n == 0 then 0 else even(n - 1)
    function even(n : int) : int =
        if n == 0 then 1 else odd(n - 1)
in
    if odd(5) then print("5 is odd")
```

Control Flow

```
let
    var j := 10
in
  /* A for loop */
  for i := 0 to 10 do
    (print_int(i); print("\n"));

  /* A while loop */
  while(j > 0) do
    (print_int(j); print("\n"); j := j - 1)
end
```

Primitive Functions (tiger standard library)

```
print(s : string)   print_int(i : int)
getchar() : string
ord(s : string) : int
chr(i : int) : string
size(s : string) : int
concat(s1 : string, s2 : string) : string
substring(s : string, f : int, n : int) : string
not(i : int) : int
exit(code : int)
```