Handout G: Nested Subprograms

Christoph Reichenbach

November 18, 2021

Subprograms allow us to encapsulate and reuse functionality. However, sometimes that functionality needs to be *contextual*, i.e., depend on properties that we would prefer to not have to pass around as parameters. One language feature that can help here are *nested subprograms*. If a subprogram g is *nested* within subprogram f, then g may access the referencing environment of f at the point at which it appears.

For example, consider a Python subprogram that parses a file. The subprogram should collect all errors in that file, but there may be many different types of errors. To simplify collecting errors, we introduce a nested helper subprogram report_error in the code below:

```
def parse_file(filename):
    file = open(filename, 'r')
    errors = []
    line_nr = 1

    # A nested subprogram:
    def report_error(message):
        errors.append(filename + ', line ' + str(line_nr) + ': ' + message)

    for line in file.readlines():
        if ...:
            report_error('syntax error')
        if ...:
            report_error('some other error')
        line_nr = line_nr + 1
    ...
```

Note that report_error looks and works like a regular subprogram, except that it is defined within another subprogram and reads the local variable line_nr and the parameter filename, and updates the list of errors errors, all of which are part of the referencing environment of its surrounding (or *outer*) subprogram parse_file.

To write the same code without nested subprograms, we have to pass all of the above variables explicitly:

```
# NOT a nested subprogram
def report_error(errors, filename, line_nr, message):
    errors.append(filename + ', line ' + str(line_nr) + ': ' + message)

def parse_file(filename):
    file = open(filename, 'r')
    errors = []
    line_nr = 1

    for line in file.readlines():
        if ...:
            report_error(errors, filename, line_nr, 'syntax error')
        if ...:
            report_error(errors, filename, line_nr, 'some other error')
    ...
```

Nested subprograms are supported e.g. in Scala, JavaScript, Go, Haskell, Ocaml, and Scheme.