P0 Type-Checking Tests

Emil Sekerinski, McMaster University, revised February 2022

In [ ]:
```
import nbimporter; nbimporter.options["only_defs"] = False
from P0 import compileString
```

Procedure `compileerr(s)` returns an empty string if compiling `s` succeeds or the error message produced while compiling; the error message is also printed. The procedure is used here to test type-checking.

In [ ]:
```
def compileerr(s):
    try: compileString(s); return ''
    except Exception as e:
        print(e); return str(e)
```

Error "index out of bounds"

In [ ]:
```
assert "index out of bounds" in compileerr("""
var x: [5 .. 7] → integer
program p
  x[4] := 3
""")
```

Error "index not integer"

In [ ]:
```
assert "index not integer" in compileerr("""
var x: [5 .. 7] → integer
program p
  x[x] := 3
""")
```

Error "not an array"

In [ ]:
```
assert "not an array" in compileerr("""
program p
  var x: integer
    x[9] := 3
""")
```

Error "not a field"

In [ ]:
```
assert "not a field" in compileerr("""
var v: (f: integer)
program p
    v.g := 4
""")
```

Error "not a record"

In [ ]:
```
assert "not a record" in compileerr("""
program p
  var v: integer
    v.g := 4
""")
```

Error "identifier expected"

In [ ]:
```
assert "identifier expected" in compileerr("""
var v: (f: integer)
program p
    v.3 := 4
""")
```

Error "variable or constant identifier expected"

In [ ]:
```
assert "variable or constant identifier expected" in compileerr("""
program p
  var x: integer
    x := write
""")
```

Error "not boolean"

In [ ]:
```
assert "not boolean" in compileerr("""
program p
  var b: boolean
    b := ¬ 3
""")
```

Error "bad type"

In [ ]:
```
assert "bad type" in compileerr("""
program p
  var b: boolean
    b := 3 and true
""")
```

Error "bad type"

In [ ]:
```
assert "bad type" in compileerr("""
program p
  var x: integer
    x := - true
""")
```

Error "bad type"

In [ ]:
```
assert "bad type" in compileerr("""
program p
  var x: integer
    x := 3 + true
""")
```

Error "bad type"

In [ ]:
```
assert "bad type" in compileerr("""
program p
  var b: boolean
    b := 3 > true
""")
```

Error "variable for result expected"

In [ ]:
```
assert "variable for result expected" in compileerr("""
program p
  read()
""")
```

Error "variable identifier expected"

In [ ]:
```
assert "variable identifier expected" in compileerr("""
program p
  const c = 3
  var x: integer
    x, c := 5, 7
""")
```

Error "duplicate variable identifier"

In [ ]:
```
assert "duplicate variable identifier" in compileerr("""
program p
  var x: integer
    x, x := 5, 7
""")
```

Error "incompatible assignment"

In [ ]:
```
assert "incompatible assignment" in compileerr("""
program p
  var b: boolean
    b := 3
""")
```

Error "unbalanced assignment"

In [ ]:
```
assert "unbalanced assignment" in compileerr("""
program p
  var b: boolean
    b := true, false
```

```
""")
```

Error "procedure identifier expected"

In [ ]:
```
assert "procedure identifier expected" in compileerr("""
program p
  var x: integer
    x ← 3
""")
```

Error "procedure expected"

In [ ]:
```
assert "procedure expected" in compileerr("""
program p
  var b: boolean
    b ← true
""")
```

Error "incompatible call"

In [ ]:
```
assert "incompatible call" in compileerr("""
program p
  var b: boolean
    b ← read()
""")
```

Error "unbalanced call"

In [ ]:
```
assert "unbalanced call" in compileerr("""
procedure q()→(r, s: integer)
  r, s := 3, 5
program p
  var x: integer
    x ← q()
""")
```

Error "procedure expected"

In [ ]:
```
assert "procedure expected" in compileerr("""
program p
  var b: boolean
    b ← integer()
""")
```

Error "variable or procedure expected"

In [ ]:
```
assert "variable or procedure expected" in compileerr("""
program p
  const c = 7
    c := 4
""")
```

Error "incompatible parameter"

In [ ]:
```
assert "incompatible parameter" in compileerr("""
program p
  write(true)
""")
```

Error "extra parameter"

In [ ]:
```
assert "extra parameter" in compileerr("""
program p
  writeln(5)
""")
```

Error "incompatible parameter"

In [ ]:
```
assert "incompatible parameter" in compileerr("""
procedure q(x, y: integer)
  writeln()
program p
  q(3, true)
""")
```

Error "extra parameter"

Error "extra parameter"

In [ ]:
```
assert "extra parameter" in compileerr("""
program p
  write(5, 7)
""")
```

Error "too few parameters"

In [ ]:
```
assert "too few parameters" in compileerr("""
program p
  write()
""")
```

Error "boolean expected"

In [ ]:
```
assert "boolean expected" in compileerr("""
program p
  if 5 then writeln()
""")
```

Error "boolean expected"

In [ ]:
```
assert "boolean expected" in compileerr("""
program p
  while 5 do writeln()
""")
```

Error "type identifier expected"

In [ ]:
```
assert "type identifier expected" in compileerr("""
type T = writeln
program p
  writeln()
""")
```

Error "bad lower bound"

In [ ]:
```
assert "bad lower bound" in compileerr("""
const l = -1
const u = 5
var a: [l .. u] → integer
program p
  writeln()
""")
```

Error "bad upper bound"

In [ ]:
```
assert "bad upper bound" in compileerr("""
const l = 7
const u = 5
var a: [l .. u] → integer
program p
  writeln()
""")
```

Error "expression not constant"

In [ ]:
```
assert "expression not constant" in compileerr("""
var v: integer
program p
  const c = v
  writeln()
""")
```