P0 AST Tests

Emil Sekerinski, McMaster University, revised February 2022

Also includes some symbol table tests.

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

Control Structures

In [ ]:
```
assert compileString("""
var a: [1..10] → integer
program p
  var x: integer
    x ← read()
    if x > 0 then
      while a[x] < 7 do
        x := x + 1
    else write(x)
    writeln()
""", target='ast') == """\
seq
  seq
    call Var(name = x, lev = 1, tp = <class 'ST.Int'>) read
    ifelse
      >
        Var(name = x, lev = 1, tp = <class 'ST.Int'>)
        Const(name = , tp = <class 'ST.Int'>, val = 0)
      while
        <
          Var(name = a, lev = 0, tp = Array(lower = 1, length = 10, base = <class 'ST.Int'>))[]
             Var(name = x, lev = 1, tp = <class 'ST.Int'>)
          Const(name = , tp = <class 'ST.Int'>, val = 7)
        :=
          Var(name = x, lev = 1, tp = <class 'ST.Int'>)
          +
            Var(name = x, lev = 1, tp = <class 'ST.Int'>)
            Const(name = , tp = <class 'ST.Int'>, val = 1)
      call  write
        Var(name = x, lev = 1, tp = <class 'ST.Int'>)
  call  writeln"""
```

In [ ]:
```
assert symTabStr() == \
[["Type(name = boolean, val = <class 'ST.Bool'>)",
  "Type(name = integer, val = <class 'ST.Int'>)",
  "Const(name = true, tp = <class 'ST.Bool'>, val = 1)",
  "Const(name = false, tp = <class 'ST.Bool'>, val = 0)",
  "StdProc(name = read, lev = 0, par = [], res = [Var(name = , lev = , tp = <class 'ST.Int'>)])",
  "StdProc(name = write, lev = 0, par = [Var(name = , lev = , tp = <class 'ST.Int'>)], res = [])",
  'StdProc(name = writeln, lev = 0, par = [], res = [])',
  "Var(name = a, lev = 0, tp = Array(lower = 1, length = 10, base = <class 'ST.Int'>))"]]
```

Records

In [ ]:
```
assert compileString("""
var a: integer
var r: (f, g: integer)
program p
    a := 3
    r.g := 5
    r.f := a
    a := r.g
""", target='ast') == """\
seq
  seq
    seq
      :=
        Var(name = a, lev = 0, tp = <class 'ST.Int'>)
        Const(name = , tp = <class 'ST.Int'>, val = 3)
      :=
        Var(name = r, lev = 0, tp = Record(fields = [Var(name = f, lev = 1, tp = <class 'ST.Int'>), Var(name = 
        Const(name = , tp = <class 'ST.Int'>, val = 5)
    :=
      Var(name = r, lev = 0, tp = Record(fields = [Var(name = f, lev = 1, tp = <class 'ST.Int'>), Var(name = g,
      Var(name = a, lev = 0, tp = <class 'ST.Int'>)
  :=
    Var(name = a, lev = 0, tp = <class 'ST.Int'>)
    Var(name = r, lev = 0, tp = Record(fields = [Var(name = f, lev = 1, tp = <class 'ST.Int'>), Var(name = g, le
```

```
In [ ]: assert symTabStr() == \
        [["Type(name = boolean, val = <class 'ST.Bool'>)",
          "Type(name = integer, val = <class 'ST.Int'>)",
          "Const(name = true, tp = <class 'ST.Bool'>, val = 1)",
          "Const(name = false, tp = <class 'ST.Bool'>, val = 0)",
          "StdProc(name = read, lev = 0, par = [], res = [Var(name = , lev = , tp = <class 'ST.Int'>)])",
          "StdProc(name = write, lev = 0, par = [Var(name = , lev = , tp = <class 'ST.Int'>)], res = [])",
          'StdProc(name = writeln, lev = 0, par = [], res = [])',
          "Var(name = a, lev = 0, tp = <class 'ST.Int'>)",
          "Var(name = r, lev = 0, tp = Record(fields = [Var(name = f, lev = 1, tp = <class 'ST.Int'>), Var(name = g, lev
```

Arrays

```
In [ ]: assert compileString("""
        var a: [1..10] → integer
        program p
          var i: integer
            a[5] := 3
            a[i] := 5
            a[i + 7] := i + 9
        """, target='ast') == """\
        seq
          seq
            :=
              Var(name = a, lev = 0, tp = Array(lower = 1, length = 10, base = <class 'ST.Int'>))[]
                 Const(name = , tp = <class 'ST.Int'>, val = 5)
              Const(name = , tp = <class 'ST.Int'>, val = 3)
            :=
              Var(name = a, lev = 0, tp = Array(lower = 1, length = 10, base = <class 'ST.Int'>))[]
                 Var(name = i, lev = 1, tp = <class 'ST.Int'>)
              Const(name = , tp = <class 'ST.Int'>, val = 5)
          :=
            Var(name = a, lev = 0, tp = Array(lower = 1, length = 10, base = <class 'ST.Int'>))[]
               +
               Var(name = i, lev = 1, tp = <class 'ST.Int'>)
               Const(name = , tp = <class 'ST.Int'>, val = 7)
            +
              Var(name = i, lev = 1, tp = <class 'ST.Int'>)
              Const(name = , tp = <class 'ST.Int'>, val = 9)"""
```

```
In [ ]: assert symTabStr() == \
        [["Type(name = boolean, val = <class 'ST.Bool'>)",
          "Type(name = integer, val = <class 'ST.Int'>)",
          "Const(name = true, tp = <class 'ST.Bool'>, val = 1)",
          "Const(name = false, tp = <class 'ST.Bool'>, val = 0)",
          "StdProc(name = read, lev = 0, par = [], res = [Var(name = , lev = , tp = <class 'ST.Int'>)])",
          "StdProc(name = write, lev = 0, par = [Var(name = , lev = , tp = <class 'ST.Int'>)], res = [])",
          'StdProc(name = writeln, lev = 0, par = [], res = [])',
          "Var(name = a, lev = 0, tp = Array(lower = 1, length = 10, base = <class 'ST.Int'>))"]]
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js