OBLIG 1 INF 5110

Andreas Heim (andrheim)

Gjennomføring

Grammatikk
Jeg valgte å bygge syntakstreet ut i fra en flertydig grammatikk, hvor presedens og assosiativitet er styrt av cup-kommandoer.
Den flertydige grammatikken har 136 unike parseringstilstander, mot 142 i den entydige utgaven av grammatikken.
Det mest utfordrende med å bygge den entydige grammatikken var å få dot-notasjonen (klasse.instansvariabel) på rett plass, og jeg er fortsatt litt usikker på om det stemmer 100%.
Rent estetisk er den flertydige grammatikken mye enklere å lese enn den entydige. Jeg har valgt å dele opp grammatikken i mange ikke-terminaler for å unngå gjentakelse, noe som sannsynligvis gjør den noe vanskeligere å sette seg inn i.
Så vidt jeg kan se vil ikke en entydig grammatikk ha stor innvirkning på utformingen av syntakstreet.

Syntakstre
Treet er bygget opp med noen forskjellige "supertyper". Klasse-, variabel-, og funksjonsdeklarasjoner deler samme superklasse.

Expression er supertype for literaler og statements. Statements arver expressions fordi et CallStatement er et Expression, selv om ingen andre statements per definisjon er expressions.

Expressionshierarkiet består også av OpExpression, en superklasse for alle Expressions som evaluerer verdier med aritmetiske, logiske eller relasjonsoperander. Hver av operandene eksisterer i en egen klasse, og det er tenkt at disse skal kunne håndtere deler av kodegenerasjonen.
Klassen for logiske uttrykk LogOpExpression har to subklasser, AndOpExpression og OrOpExpression. Disse er designet annerledes enn AritOpExpression og RelOpExpression-klassene ved at de ikke har et sett operand-klasser som sendes inn i konstruktøren. Dette vil muligens måtte forandres i neste oblig, men slik som grammatikken utfoldet seg så falt dette mest naturlig.

Kjøring
Flertydig syntaks kjøres med "ant clean compile run" og entydig syntaks kjøres med "ant clean generate-alt-syntax compile run-alt-syntax"

Utskrift av Canonical.d (også vedlagt i code-examples/)
```
(PROGRAM
      (CLASS (NAME Complex)
            (VAR_DECL (TYPE float)(NAME Real))
            (VAR_DECL (TYPE float)(NAME Imag))
      )

      (FUNC_DECL (TYPE void) (NAME Swap)
            (PARAM_DECL ref (TYPE int)(NAME a))
            (PARAM_DECL ref (TYPE int)(NAME b))
            (VAR_DECL (TYPE int)(NAME tmp))
            (ASSIGN_STMT
```

```
                (NAME tmp)
                (NAME a)
)

          (ASSIGN_STMT
                (NAME a)
                (NAME b)
)

          (ASSIGN_STMT
                (NAME b)
                (NAME tmp)
)

     )

     (FUNC_DECL (TYPE (NAME Complex)) (NAME Add)
          (PARAM_DECL (TYPE (NAME Complex))(NAME a))
          (PARAM_DECL (TYPE (NAME Complex))(NAME b))
          (VAR_DECL (TYPE (NAME Complex))(NAME retval))
          (ASSIGN_STMT
                (NAME retval)
                (NEW (TYPE (NAME Complex)))
)

          (ASSIGN_STMT
                ( . (NAME retval) (NAME Real))
                (ARIT_OP +
                     ( . (NAME a) (NAME Real))
                     ( . (NAME b) (NAME Real))
                )
)

          (ASSIGN_STMT
                ( . (NAME retval) (NAME Imag))
                (ARIT_OP +
                     ( . (NAME a) (NAME Imag))
                     ( . (NAME b) (NAME Imag))
                )
)

          (RETURN_STMT
                (NAME retval)
          )

     )

     (FUNC_DECL (TYPE int) (NAME Max)
```

```
        (PARAM_DECL (TYPE int)(NAME a))
        (PARAM_DECL (TYPE int)(NAME b))
        (IF_STMT
            (REL_OP >
                (NAME a)
                (NAME b)
            )

        (
            (RETURN_STMT
                (NAME a)
            )

        )

        (RETURN_STMT
            (NAME b)
        )

    )

    (FUNC_DECL (TYPE void) (NAME Main)
        (FUNC_DECL (TYPE float) (NAME Square)
            (PARAM_DECL (TYPE float)(NAME val))
            (RETURN_STMT
                (ARIT_OP **
                    (NAME val)
                    (FLOAT_LITERAL 2.0)
                )
            )

        )
        (VAR_DECL (TYPE float)(NAME num))
        (ASSIGN_STMT
            (NAME num)
            (FLOAT_LITERAL 6.480740)
)

        (CALL_STMT (NAME print_float)
            (ACTUAL_PARAM (NAME num))

        (CALL_STMT (NAME print_str)
            (ACTUAL_PARAM (STRING_LITERAL "  squared is  "))

        (CALL_STMT (NAME print_float)
            (ACTUAL_PARAM (CALL_STMT (NAME Square)
    (ACTUAL_PARAM (NAME num))
)
```

```
            (RETURN_STMT)
        )
    )
```