G-Opgave

Michael Thulin
Philip Munksgaard

15. december 2010

1 Tilføjelser til Lexer.lex

Vi har tilføjet de manglende keywords og tokens til Lexer.lex.

2 Parser.grm

Her startede vi med at tilføje de manglende tokens i grammatikken. Derefter tilføjede vi de manglende typer og så de manglende produktioner. Så endte vi op med mere end 15 shift/reduce konflikter. For at slippe af med nogle af disse, eliminerede vi venstre-rekursion i adskillige af produktionerne, heriblandt Types, Pats og Exps. I nogle produktionerne, f.eks. Dec og Match, valgte vi at elmiminere tvetydighed ved at sætte associativitet korrekt. Den eneste shift/reduce-fejl der voldte os nævneværdige problemer var tvetydighed omkring $LPARExpRPAR$, som vi valgte at løse ved at dele produktionen $Exp \rightarrow LPARExpsRPARCOLONID$ op i to produktioner, $Exp \rightarrow LPARExpRPARCOLONID$ og $Exp \rightarrow LPARExpCOMMAExpsRPARCOLONID$, jvf. bilag 1.

3 Type.sml

4 Compiler.sml

5 Bilag 1: Produktioner for Exp

```
Exp :     NUM              { Cat.Num $1 }
        | TRUE             { Cat.True $1 }
        | FALSE            { Cat.False $1 }
        | NULL COLON ID    { Cat.Null (#1 $3, $1) }
        | ID               { Cat.Var $1 }
        | LPAR Exp COMMA Exps RPAR COLON ID
                           { Cat.MkTuple ( $2 :: $4, #1 $7, $1 ) }
        | LPAR Exp RPAR COLON ID
                           { Cat.MkTuple ([$2], #1 $5, $1) }
        | Exp PLUS Exp     { Cat.Plus ($1, $3, $2) }
        | Exp MINUS Exp    { Cat.Minus ($1, $3, $2) }
        | Exp EQUAL Exp    { Cat.Equal ($1, $3, $2) }
        | Exp LESSTHAN Exp
                           { Cat.Less ($1, $3, $2) }
        | NOT Exp          { Cat.Not ($2, $1) }
        | Exp AND Exp      { Cat.And ($1, $3, $2) }
        | Exp OR Exp       { Cat.Or ($1, $3, $2) }
```

```
        | IF Exp THEN Exp ELSE Exp
                        { Cat.If ($2, $4, $6, $1) }
        | LET Dec IN Exp
                        { Cat.Let ($2, $4, $1) }
        | CASE Exp OF Match END
                        { Cat.Case ($2, $4, $1) }
        | ID Exp %prec WRITE
                        { Cat.Apply (#1 $1, $2, #2 $1) }
        | READ          { Cat.Read $1 }
        | WRITE Exp     { Cat.Write ($2, $1) }
        | LPAR Exp RPAR { $2 }
;
```