

# Ukeoppgaver i INF3110/4110

Uke 44 (29.-31.10.2003)

## Her er en grammatikk for eksempelspråket L

$\langle program \rangle \rightarrow \langle setningsListe \rangle$   
 $\langle setningsListe \rangle \rightarrow \langle setning \rangle^+$   
 $\langle setning \rangle \rightarrow \langle tilordning \rangle \mid \langle inputSetning \rangle \mid \langle outputSetning \rangle$   
 $\langle tilordning \rangle \rightarrow \langle variabel \rangle = \langle variabel \rangle \langle operator \rangle \langle operand \rangle$   
 $\langle operator \rangle \rightarrow + \mid -$   
 $\langle operand \rangle \rightarrow \langle variabel \rangle \mid \langle tall \rangle$   
 $\langle inputSetning \rangle \rightarrow ? \langle variabel \rangle$   
 $\langle outputSetning \rangle \rightarrow ! \langle variabel \rangle$   
 $\langle variabel \rangle \rightarrow v \langle siffer \rangle$   
 $\langle siffer \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$   
 $\langle tall \rangle \rightarrow \langle siffer \rangle^+$

## Oppgave 1

Bli kjent med språket.

Skriv L-programmer som

- leser inn to tall fra brukeren og skriver ut summen
- skriver ut et tall  $n$
- leser inn et tall  $n$  og returnerer tallet  $10 \times n$

## Oppgave 2

Du skal nå skrive ferdig en parser for L. Her er litt repetisjon fra forelesningene:

### En enkel *recursive descent*-parser i ML

Idé: La hvert terminalsymbol være en konstruktør av type token:

```
datatype token = Pluss | Gange | Navn;  
exception Parserfeil of string; (* for å gi feilmeldinger *)
```

Vi forutsetter at en skanner gjør om strenger til lister av tokens. (Skanneren er i dette tilfellet en funksjon som tar “**navn \* navn + navn**” som argument og som gir listen `[Navn, Gange, Navn, Pluss, Navn]` av tokens.)

For hvert metasympol ønsker vi nå en funksjon fra token list til token list. Vi lar denne funksjonen ha samme navn som metasympolet.

## En enkel *recursive descent*-parser i ML

$\langle \text{uttrykk} \rangle \rightarrow \langle \text{term} \rangle \langle \text{xterm} \rangle$

```
fun uttrykk tokens =  
  let val tokens1 = term tokens  
      val tokens2 = xterm tokens1  
  in tokens2 end
```

$\langle \text{xterm} \rangle \rightarrow + \langle \text{term} \rangle \langle \text{xterm} \rangle \mid \varepsilon$

```
and xterm tokens =  
  case tokens of Pluss :: resttokens =>  
    (let val tokens1 = term resttokens  
        val tokens2 = xterm tokens1  
        in tokens2 end)  
  | _ => tokens
```

## En enkel *recursive descent*-parser i ML

$\langle \text{term} \rangle \rightarrow \text{navn} \langle \text{xnavn} \rangle$

```
and term tokens =  
  case tokens of Navn :: resttokens => xnavn resttokens  
  | _ => raise Parserfeil("<term> begynner ikke med 'navn'")
```

$\langle \text{xnavn} \rangle \rightarrow * \text{navn} \langle \text{xnavn} \rangle \mid \varepsilon$

```
and xnavn tokens =  
  case tokens of Gange :: Navn :: resttokens => xnavn resttokens  
  | _ => tokens;
```

Og til slutt:

```
fun parse tokens =  
  let val resultat = uttrykk tokens  
  in if resultat = [] then resultat  
    else raise Parserfeil("Noe igjen etter godkjent streng") end;
```

I filen `Lscanner.sm1` finner du en ferdig scanner og en nesten ferdig parser for L. Legg merke til at det hvert metasymbol er en tilsvarende funksjon som tar en liste av tokens som argumenter.

Hjelpfunksjonene og skanneren trenger du ikke å skjønne fullstendig. Funksjonen `scanFil` tar et filnavn som argument og returnerer en liste av tokens som parseren kan overta.

- skriv ferdig funksjonen `setning`
- skriv ferdig funksjonen `inputSetning`

Sjekk at parseren virker ved å kjøre den på noen eksempler, f.eks. de tre programmene fra første oppgave.

## Oppgave 3

Vi endrer nå litt på grammatikken for  $\langle \text{tilordning} \rangle$ :

$\langle \text{tilordning} \rangle \rightarrow \langle \text{variabel} \rangle = \langle \text{operand} \rangle \langle \text{operator} \rangle \langle \text{operand} \rangle \mid \langle \text{variabel} \rangle = \langle \text{tall} \rangle$

- Skriv om parserern slik at denne endringene ivaretas.

Sjekk at alt fungerer ved å kjøre parseren på enkle eksempler. F.eks. skal nå følgende godtas som ikke ble godtatt tidligere:

```
- parse(scan "v1=123");
```

```
Parsing vellykket!
```

```
- parse(scan "v1=123 + v9");
```

```
Parsing vellykket!
```

## Oppgave 4

Her skal du skrive en enkel interpret for L ved å gjøre noen små endringer på parseren. Anta at følgende funksjoner er gitt:

- `setLager( $i, x$ )` som legger tallet  $x$  inn i lagerplass nummer  $i$
- `getLager( $i$ )` som tar et tall  $i$  som argument og som gir det som ligger i lagerplass nummer  $i$
- `lesTilLager( $i$ )` som ber brukeren om en verdi for  $v_i$  og som legger det på lagerplass  $i$
- `lesFraLager( $i$ )` som skriver ut verdien i lagerplass  $i$

Ta utgangspunkt i filen `Linterpret.sml`. Det eneste du trenger å fylle inn er funksjonene `tilordning`, `operator` og `operand`. Funksjonene `inputSetning` og `outputSetning` er allerede skrevet om. (Bortsett fra disse er alt helt identisk med parseren.)