Dagens tema

- Syntaksanalyse og parsing
  (Komp. 47, kap. 3 og 4)
  - Repetisjon, begreper, skanner
  - LL(1)-parsering
  - Eksempler
  - Noen teknikker: fjerning av venstrerekursjon, venstrefaktorisering
  - En enkel *recursive descent*-parser i ML
  - Teori: Beregning av (utvidede) startmengder mm.

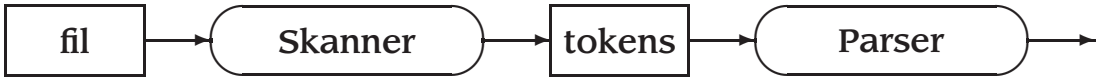
Repetisjon

Velkjente begreper nå:

- grammatikker, språk, alfabeter, terminaler, metasymboler, produksjoner
- parsing, syntakstrær, bottom-up og top-down parsing
- avledninger

Skanner

Det er vanlig at en parser har en «preprosessor» som kalles en *skanner*:



Den setter sammen tegn til *symboler* (ofte kalt *tokens*).

LL(1)-parsering

- LL(1)-parsering er en top-down strategi der vi foretar en *venstreavledning* fra startsymbolet.

LL(1)-parsering

- LL(1)-parsering er en top-down strategi der vi foretar en *venstreavledning* fra startsymbolet.
- LL(1) = **a)** Leser fra venstre mot høyre,**b)** finner venstre "syntaksparenteser" og **c)** trenger bare å se *ett symbol* for at neste steg skal være gitt.

LL(1)-parsering

- LL(1)-parsering er en top-down strategi der vi foretar en *venstreavledning* fra startsymbolet.
- LL(1) = **a)** Leser fra venstre mot høyre,**b)** finner venstre "syntaksparenteser" og **c)** trenger bare å se *ett symbol* for at neste steg skal være gitt.
- Grammatikker som tillater slik parsering kalles gjerne LL(1)-grammatikker.

LL(1)-parsering

Recursive descent

- Til hvert metasymbol svarer en metode.

LL(1)-parsering

Recursive descent

- Til hvert metasymbol svarer en metode.
- Metoden tar seg av det ene metasymbolet, men kan kalle andre metoder.

LL(1)-parsering

Recursive descent

- Til hvert metasymbol svarer en metode.
- Metoden tar seg av det ene metasymbolet, men kan kalle andre metoder.
- Når metoden kalles, skal skanneren inneholde første symbol i den aktuelle produksjonen (for syntaktisk korrekt setning).

LL(1)-parsering

Recursive descent

- Til hvert metasymbol svarer en metode.
- Metoden tar seg av det ene metasymbolet, men kan kalle andre metoder.
- Når metoden kalles, skal skanneren inneholde første symbol i den aktuelle produksjonen (for syntaktisk korrekt setning).
- Når metoden er ferdig, skal skanneren inneholde første symbol etter den leste teksten.

LL(1)-parsering

Hva er motivasjonen?

Vi ønsker å finne (kontekstfrie) grammatikker som er slik at

- hvis første *symbol* i input er gitt

LL(1)-parsering

Hva er motivasjonen?

Vi ønsker å finne (kontekstfrie) grammatikker som er slik at

- hvis første *symbol* i input er gitt
- og venstresiden av en produksjon er gitt,

LL(1)-parsering

Hva er motivasjonen?

Vi ønsker å finne (kontekstfrie) grammatikker som er slik at

- hvis første *symbol* i input er gitt
- og venstresiden av en produksjon er gitt,
- så skal valget av høyreside være unikt bestemt.

LL(1)-parsering

Hva er motivasjonen?

Vi ønsker å finne (kontekstfrie) grammatikker som er slik at

- hvis første *symbol* i input er gitt
- og venstresiden av en produksjon er gitt,
- så skal valget av høyreside være unikt bestemt.

Lykkes vi i å finne en slik LL(1)-grammatikk for et gitt språk, så kan vi lage elegante og effektive *rekursive* metoder som parser input top-down og fra venstre mot høyre.
(= recursive descent)

Eksempler

Sist så vi grammatikken:

$\langle uttrykk \rangle \rightarrow \langle uttrykk \rangle$ **+** $\langle term \rangle \mid \langle term \rangle$
$\langle term \rangle \quad \rightarrow \langle term \rangle * \mathbf{navn} \mid \mathbf{navn}$

og uttrykket **navn * navn + navn**

Eksempler

Sist så vi grammatikken:

$\langle uttrykk \rangle \rightarrow \langle uttrykk \rangle$ **+** $\langle term \rangle \mid \langle term \rangle$
$\langle term \rangle \quad \rightarrow \langle term \rangle * \mathbf{navn} \mid \mathbf{navn}$

og uttrykket **navn * navn + navn**

Er det mulig å lage en slik *recursive descent*-parser for denne grammatikken?

Eksempler

Sist så vi grammatikken:

$\langle uttrykk \rangle \rightarrow \langle uttrykk \rangle$ **+** $\langle term \rangle \mid \langle term \rangle$
$\langle term \rangle \quad \rightarrow \langle term \rangle * \textbf{navn} \mid \textbf{navn}$

og uttrykket **navn * navn + navn**

Er det mulig å lage en slik *recursive descent*-parser for denne grammatikken?

Ikke uten videre. . .

Eksempler

Hvis det er et ⟨*uttrykk*⟩ som skal leses og symbolet **navn** er det første symbolet, hvilken høyreside skal velges?

⟨*uttrykk*⟩ → ⟨*uttrykk*⟩ **+** ⟨*term*⟩ | ⟨*term*⟩

Eksempler

Hvis det er et ⟨*uttrykk*⟩ som skal leses og symbolet **navn** er det første symbolet, hvilken høyreside skal velges?

⟨*uttrykk*⟩ → ⟨*uttrykk*⟩ **+** ⟨*term*⟩ | ⟨*term*⟩

Umulig å si ut ifra slik det er gitt over.

Eksempler

Hvis det er et ⟨*uttrykk*⟩ som skal leses og symbolet **navn** er det første symbolet, hvilken høyreside skal velges?

⟨*uttrykk*⟩ → ⟨*uttrykk*⟩ **+** ⟨*term*⟩ | ⟨*term*⟩

Umulig å si ut ifra slik det er gitt over.

Vi skal se på to teknikker for å gjøre grammatikkene mer håndterbare, og i beste tilfelle LL(1).

Fjerning av venstrerekursjon

$\langle uttrykk \rangle \rightarrow \langle uttrykk \rangle$ **+** $\langle term \rangle \mid \langle term \rangle$

Strategi:

Fjerning av venstrerekursjon

$\langle uttrykk \rangle \rightarrow \langle uttrykk \rangle$ **+** $\langle term \rangle \mid \langle term \rangle$

Strategi:

1) Skriv om til utvidet BNF:

$\langle uttrykk \rangle \rightarrow \langle term \rangle \; [\![$ **+** $\langle term \rangle \;]\!]^*$

Fjerning av venstrerekursjon

$\langle uttrykk \rangle \rightarrow \langle uttrykk \rangle$ **+** $\langle term \rangle \mid \langle term \rangle$

Strategi:

1) Skriv om til utvidet BNF:

$\langle uttrykk \rangle \rightarrow \langle term \rangle$ **[[** **+** $\langle term \rangle$ **]]** *

2) Før tilbake til klassisk BNF, men nå med høyrerekursjon (og eventuelt ekstra metasymboler):

$\langle uttrykk \rangle \rightarrow \langle term \rangle \langle xterm \rangle$
$\langle xterm \rangle \quad \rightarrow$ **+** $\langle term \rangle \langle xterm \rangle \mid \varepsilon$

Venstrefaktorisering

Ofte vil to alternative startsider kunne begynne på samme måte, men forskjellig avslutning, som i:

$$\langle setning \rangle \rightarrow \langle uttrykk \rangle \texttt{ + } \langle term \rangle \mid \langle uttrykk \rangle \texttt{ * } \langle term \rangle$$

Venstrefaktorisering

Ofte vil to alternative startsider kunne begynne på samme måte, men forskjellig avslutning, som i:

⟨*setning*⟩ → ⟨*uttrykk*⟩ **+** ⟨*term*⟩ |
⟨*uttrykk*⟩ ***** ⟨*term*⟩

Vi kan her innføre et nytt metasymbol som skal stå for den varierende delen:

⟨*setning*⟩ → ⟨*uttrykk*⟩ ⟨*xsetning*⟩
⟨*xsetning*⟩ → **+** ⟨*term*⟩ | ***** ⟨*term*⟩

(Vi *"faktoriserer"* på samme måte som i $(x * 3) + (x * 5) = x * (3 + 5)$.)

Tilbake til eksemplet

$\langle uttrykk \rangle \rightarrow \langle uttrykk \rangle$ **+** $\langle term \rangle \mid \langle term \rangle$
$\langle term \rangle \rightarrow \langle term \rangle * \mathbf{navn} \mid \mathbf{navn}$

gir ved fjerning av venstrerekursjon:

$\langle uttrykk \rangle \rightarrow \langle term \rangle \langle xterm \rangle$
$\langle xterm \rangle \rightarrow$ **+** $\langle term \rangle \langle xterm \rangle \mid \varepsilon$
$\langle term \rangle \rightarrow$ **navn** $\langle xnavn \rangle$
$\langle xnavn \rangle \rightarrow * \mathbf{navn} \langle xnavn \rangle \mid \varepsilon$

Er det nå entydig gitt hvilken høyreside vi til enhver tid skal velge?

JA: **navn * navn + navn**

$$
\begin{aligned}
\langle uttrykk \rangle &\rightarrow \langle term \rangle \ \langle xterm \rangle \\
\langle xterm \rangle &\rightarrow \mathbf{+} \ \langle term \rangle \ \langle xterm \rangle \mid \varepsilon \\
\langle term \rangle &\rightarrow \mathbf{navn} \ \langle xnavn \rangle \\
\langle xnavn \rangle &\rightarrow * \ \mathbf{navn} \ \langle xnavn \rangle \mid \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
\langle uttrykk \rangle &\Rightarrow \langle term \rangle \ \langle xterm \rangle \\
&\Rightarrow \mathbf{navn} \ \langle xnavn \rangle \ \langle xterm \rangle \\
&\Rightarrow \mathbf{navn} * \mathbf{navn} \ \langle xnavn \rangle \ \langle xterm \rangle \\
&\Rightarrow \mathbf{navn} * \mathbf{navn} \ \langle xterm \rangle \\
&\Rightarrow \mathbf{navn} * \mathbf{navn} + \langle term \rangle \ \langle xterm \rangle \\
&\Rightarrow \mathbf{navn} * \mathbf{navn} + \mathbf{navn} \ \langle xnavn \rangle \ \langle xterm \rangle \\
&\Rightarrow \mathbf{navn} * \mathbf{navn} + \mathbf{navn} \ \langle xterm \rangle \\
&\Rightarrow \mathbf{navn} * \mathbf{navn} + \mathbf{navn}
\end{aligned}
$$

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 metasymbol ønsker vi nå en funksjon fra `token list` til `token list`. Vi lar denne funksjonen ha samme navn som metasymbolet.

En enkel *recursive descent*-parser i ML

$\langle uttrykk \rangle \rightarrow \langle term \rangle \ \langle xterm \rangle$

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

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

```
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

$\langle term \rangle \rightarrow$ **navn** $\langle xnavn \rangle$

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

$\langle xnavn \rangle \rightarrow *$ **navn** $\langle xnavn \rangle \mid \varepsilon$

```
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;
```

Alternative høyresider

For et metasymbol med alternative høyresider må vi beregne startmengden til hver produksjon, det vil si hvilke grunnsymboler som kan stå først i en setning avledet fra høyresiden i denne produksjonen.

Hvis de ulike produksjonene til et metasymbol har *disjunkte* startmengder, kan vi sjekke "klarsymbolet" (det siste symbolet lest av skanneren) for å finne ut hvilken produksjon vi skal velge.

LL(1)-grammatikker

Vi kan ikke lage en "recursive descent"-parser for alle grammatikker, de må (som vi har sett) oppfylle følgende krav:

- Grammatikken må være **kontekstfri**, dvs. kun ha ett metasymbol på venstre side.
- Under parseringen må vi vite hvilket alternativ vi skal velge; dvs. at mengden av startsymboler (den utvidede startmengden) for hvert alternativ må være disjunkte.

En LL(1)-grammatikk er en grammatikk som oppfyller dette, og slik at vi hele tiden kan se på "neste" symbol (men ikke mer) for å bestemme hvilket alternativ som skal velges.

Beregning av utvidede startmengder

For å beregne de utvidede startmengdene må vi først ha beregnet tre andre mengder:

Startmengden til et metasymbol: Mengden av de grunnsymbolene som kan stå først i en (del-)setning avledet fra dette metasymbolet.

Etterfølgermengden til et metasymbol: Mengden av de grunnsymboler som kan stå etter metasymbolet på et tidspunkt i avledningen.

Meta-til-tom mengden: Mengden av de metasymboler som kan videreavledes til den tomme setning.

Meta-til-tom mengden

Metasymboler som kan produsere den tomme setningen kompliserer ting litt, så det kan være greit å ha beregnet denne først:

1. Initialisering: La meta-til-tom mengden bestå av de metasymboler som har en tom høyreside.
2. Dersom det finnes et metasymbol med en høyreside som bare består av metasymboler som er med i meta-til-tom mengden, skal dette også inkluderes i mengden.
3. Gjenta inntil meta-til-tom mengden ikke øker mer.

Startmengder

For enkelthets skyld sier vi at grunnsymboler har seg selv som startmengde.

1. Initialisering: La startmengden for hvert metasymbol være de grunnsymbolene som står først i en høyreside for dette metasymbolet.
2. Se på en høyreside for et metasymbol M. For hvert (grunn- eller meta-)symbol x i høyresiden som enten står først, eller bare har metasymboler fra meta-til-tom mengden foran, øk startmengden til M med startmengden til x.
3. Gjenta inntil ingen av startmengdene kan økes mer.

Etterfølgermengder

1. Initialisering: For hvert metasymbol, la etterfølgermengden være unionen av startmengdene til de (grunn- eller meta-) symbolene som i en eller annen høyreside står enten
   - umiddelbart bak dette metasymbolet, eller
   - bak slik at de mellomliggende symboler er metasymboler i meta-til-tom mengden.
2. Hvis metasymbolet M i en høyreside til metasymbolet N enten står bakerst, eller det bak M bare står metasymboler i meta-til-tom mengden: øk etterfølgermengden til M med etterfølgermengden til N.
3. Gjenta til ingen av etterfølgermengdene kan økes mer.

Eksempel

Følgende er hentet fra eksamen 1996, oppg. 1a:

$U' \rightarrow U\ @$
$U \rightarrow A\ B \mid +$
$A \rightarrow (\ U\ ) \mid \varepsilon$
$B \rightarrow * \mid \varepsilon$

Oppgave: "Beregn og angi meta-til-tom mengden, og start- og etterfølgermengden til hvert av metasymbolene."

Metasymbol	MTT	Start		Etterfølger	
U'	Nei		+ (* @		
U	Ja	+	(*	) @	
A	Ja	(		*	@)
B	Ja	*			@)

Utvidede startmengder - endelig!

For en gitt produksjon består denne mengden av alle symbolene i startmengden til de symbolene i høyresiden som enten står først eller bare har metasymboler i meta-til-tom mengden foran seg.

Hvis høyresiden er tom eller bare består av meta-til-tom mengden, inngår også symbolene i etterfølgermengden til produksjonens venstreside.

Vi ser igjen på grammatikken:

$U' \rightarrow U$ @
$U \rightarrow A\ B \mid$ +
$A \rightarrow$ (U) $\mid \varepsilon$
$B \rightarrow$ * $\mid \varepsilon$

Produksjon	Utvidet startmengde	Disjunkt?
$U' \rightarrow U$ @		
$U \rightarrow A\ B$ $U \rightarrow$ +		
$A \rightarrow$ (U) $A \rightarrow \varepsilon$		
$B \rightarrow$ * $B \rightarrow \varepsilon$		

Oppgave: "Avgjør om dette er en LL(1)-grammatikk. Begrunn svaret."