Spørsmål og svar rundt oblig 1 og verktøy

Sven-Jørgen Karlsen,
<svenjok@ifi.uio.no>
gruppelærer i INF5110 våren 2006

Obligen i ett nøtteskall

- Del 1:

  1) Lage en parser og skanner for Diss, med en grammatikkforvikling (den venstrefaktoriserte LL(1)-grammatikken).

  2) Designe og implementere ett syntakstre.
  To ytterpunkter:

  1) 1-3 low-level, veldig generelle nodetyper (hvis man leser Louden-eksemplene for bokstavlig, skyldes egentlig svakheter i C).

  2) Ca. 25 klasser og grensesnitt: Mitt løsningsforslag, kan nok kritiseres for å være overdrevent og detaljert.

  Dere bør antakeligvis plassere dere ett sted midt mellom disse.

  Designet bygges på normal OO-måte fra den opprinnelig Diss-grammatikken, venstresider blir stort sett typer, alternativer på høyresidene subtyper.

Obligen i ett nøtteskall (2)

- Del 2: Sjekke statisk semantikk i språket. Stort sett std. tester i to hovedkategorier:
  - Bruk av navn vs. deklarasjoner.
  - Typesjekking, med litt typekonvertering.
- "Eat your own dog food": Parseren og syntakstreet fra del 1, skal brukes i del 2, så for å unngå merarbeid, lønner det seg å gjøre en skikkelig jobb i del 1.

Skopet til Oblig 1

- Oblig 1 dreier seg bare om syntaks og utforming av abstrakt syntakstre. Derfor trenger dere fra språknotatet bare:
  - Beskrivelsene av syntaktisk og leksikalsk struktur fra til avsn. “Semantikk”. Hopp over forklaringer som grenser opp til semantikk, det gjelder særlig
    - “Indre deklarasjoner”
    - Evaluering, beregning av uttrykk, “Short-circuit evaluation”
    - Typekonvertering
    - Std.bibliotek
  - Resten skumleses for oversiktens skyld og inputs til designet av syntakstreet.

Oppgaveteksten - omskriving

- Hvilken rekkefølge bør man foreta omskrivingene i?
  1) Presedenskaskade (se 3.4 i Louden)
  2) Skriv om til riktig assosiativitet (se. 3.4.2 i Louden)
  3) Fjerne venstrerekursjon (se 4.2.3 i Louden)
  4) [Skriv om til EBNF] (vanligvis er EBNF å foretrekke, se også spm. om korrigering av assosiativitet)
     Det viktigste er skillet mellom det første steget og de andre, de tre siste kan blandes sammen.
- NB: Omskriving av uttr.grammatikken må gjøres i begge grammatikkene, derfor lønner det seg å starte med den flertydige og konsise, og bruke den som input til venstrefaktorisering

Oppgaveteksten – korrigere assosiativitet

- Hvordan rette opp syntakstrær som har blitt høyreassosiative etter fjerning av venstrerekursjon?
- Generelt, så er det to løsninger:

  1) Bruk EBNF. Antlr genererer kode omtrent som beskrevet i 4.2 av Louden.

  2) Bruk en parameter for å sende venstresida til en syntakstrenode med i en høyrerekursiv BNF-regel.

  For å gjøre en lang historie kort: Det første alternativet er å foretrekke, det er kortere og klarere, og unngår komplikasjonene i aksjonskoden som det andre tilfører.

Oppgaveteksten – utskrift

- Hvordan skrive ut noder som ikke er vist i Canonical.ast?
- Generell syntaks:

```
node       : "(" NODE_NAME (attributes)* (node)* ")"
attributes : "ref"
```

hvor NODE_NAME er vs. til produksjonen noden er utledet fra i Diss-grammatikken.

Antlr – staving av identifikatorer

- Symptom:
  program : (Decl)* EOF ;

  i stedet for:

  program : (decl)* EOF ;

  genererer og kompilerer, men gir parsefeil run-time.
- Lærepenge: Pass på store og små bokstaver.
  Stor forbokstav => terminal/token.
  Liten forbokstav => nonterminal.

Antlr - vokabularer

- Symptom: line 19:13: expecting '.', found '.'

  skyldes at parseren og skanneren har forskjellig oppfatning av kodeverdier for den samme terminalen i konstant-tabellen.
- Lang historie kort: Bruk oppskriften på oppsett av to parsere og en skanner fra FAQ-en.

Antlr - skanning

- Hvordan begrense lengden til terminaler?
  - Kort svar: Dessverre støtter ikke Antlr lengdebegrensninger i skannerspråket.
    Siden lengden av id. ikke er en viktig del av obligen, er det greitt å droppe dette.
  - Lengre svar: De som er interessert kan prøve ut ett semantisk validerende predikat, noe á la:
    NAME : LETTER (DIGIT | LETTER | '_')* { text.length() <= 16 }? ;

    NB: Dette skaper nye problemer med Antlrs defaulte feilhåndtering, og krever noe mere arbeid.

Antlr - feilhåndtering

- Hva betyr "Det er ikke krave om noen spesielt intelligent feilmelding"?
  - Betyr det at antlr sine feilmeldinger kan brukes uten noe mer?
- Ja, med to unntak beskrevet forrige gang:
  - Linjetelling i skanneren
  - Deteksjon av feilstatus i parserklassen.
    Begge deler er demonstrert i eksemplene.