Tentamen

Översättarteknik, 5p
22 December 1993

- Skrivtid 9-15
- Hjälpmedel: Parsons. Introduction to Compiler Construction
- Maxpoäng: 64
- Godkäntgräns: 49
- VG-gräns: 57
- Uppgifterna är <u>inte</u> slumpmässigt ordnade
- Fråga om någonting är oklart
- L ycka till!

Uppgift 1 (12 p)

För varje uppgift i denna tentamen.

a) Börja varje uppgift på nytt blad och skriv namn och uppgiftsnummer längst upp till <u>höger</u> på bladet.

b) Skriv endast på en sida av varje blad.

Uppgift 2 (12 p)

Följande grammatik beskriver uttryck i ett funktionellt språk.

expr → atom
expr → expr atom
atom → **num**
atom → **id**
atom → (expr)

a) Skriv om grammatiken så att den blir LL(1). Förklara vilka omskrivningar du gör.

b) Konstruera en LL(1)-parser för den omskrivna grammatiken i a).

c) Visa hur LL-parsern analyserar strängen `id (id num)`.

Uppgift 3 (12 p)

Utgående från följande reguljära uttryck:
$((\varepsilon \mid a)\, b^*)^*$

a) Konstruera en icke-deterministisk finit automat som accepterar det språk som beskrivs av uttrycket. Använd de metoder som beskrivs i boken med följande tillägg:
   1. Ingen övergång får finnas från ett accepterande tillstånd.
   2. Ingen övergång får finnas till starttillståndet.

b) Konstruera en ekvivalent deterministisk automat utifrån den ickedeterministiska automaten i a).

c) Förklara varför tilläggen i a) behövs. Ge exempel.

Uppgift 4 (10 p)

Betrakta följande grammatik som beskriver lambdakalkyl.

expr → λ **id .** expr
expr → **(** expr expr **)**
expr → **id**
expr → **(** expr **)**

a) Konstruera en SLR(1)-parser för grammatiken. Om du stöter på konflikter välj någon tolkning och kommentera denna. Använd _ istället för **.** i LR(0)-items för att undvika missförstånd.

b) Visa hur SLR-parsern analyserar strängen λx**.**λy**.**(xy). Visa för varje reduktion hur parseträdet byggs.

Uppgift 5 (8 p)

Förklara kortfattat följande begrepp

a) En **tvetydig** grammatik.
b) En **vänsterrekursiv** gramatik.
c) En **recursive descent** parser.
d) En **prediktiv parser**.

Uppgift 6 (4 p)

Använd tekniken med listor av kvadrupelnummer och *backpatching* för att skriva ett översättningsschema för följande produktioner:

a)

$S \rightarrow \textbf{do}\ S_1\ \textbf{while}\ (E)$

b)

$E \rightarrow E_1\ \textbf{nand}\ E_2$

E_1 **nand** E_2 är falskt om både E_1 och E_2 är sanna, annars sant.

Uppgift 7 (2 p)

Utgående från en grammatik som beskriver uttryck utan prioritet och associativitet. Ge en generell beskrivning av hur man skriver om grammatiken så att den tar hänsyn till prioritet och associativitet hos operatorer . Du ska kunna hantera godtyckligt antal prioritetsnivåer och godtycklig associativitet hos operatorerna.

Uppgift 8 (2 p)

För typkontroll av språk utan överladdade operatorer kan man för uttryck använda ett attribut *type*. Detta attribut representerar typen hos uttrycket. Om språket innehåller överladdade operatorer kan ett uttryck innan man har utrett alla detaljer ha flera typer och överladdade operatorer kan ta flera typer på operander och ge flera typer på resultat.
Specificera ett översättningsschema där man kontrollerar typen på uttryck bestående av identifierare och funktionsapplikation

$$E \rightarrow \textbf{id}$$
$$E \rightarrow E\ (\ E\ )$$

Beräkna i detta översättningsschema attributet *types* hos E som är mängden av alla möjliga typer hos E.

Uppgift 9 (2 p)

När man konstruerar LR-parsrar kan det uppkomma konfliker .

a) När uppkommer en shift/reduce- respektive en reduce/reduce-konflikt ?
   Vad återspeglar dessa konflikter för stackinnehåll respektive inputsymbol ?

b) Beskriv en metod för att lösa shift/reduce- respektive reduce/reduce-konflikter .