UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

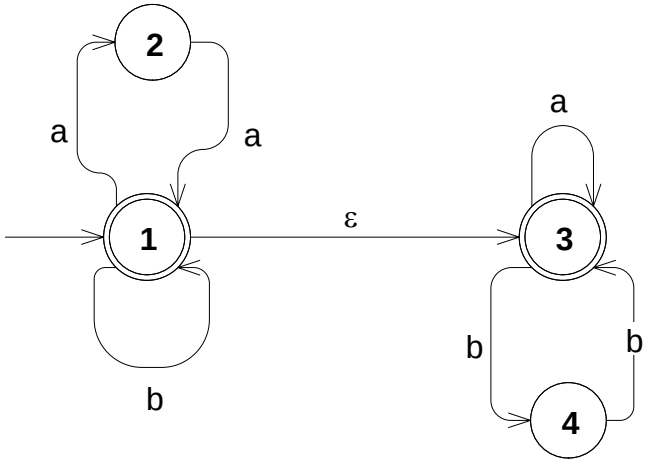
Eksamen i :	INF5110
Eksamensdag :	Torsdag 9. juni 2005
Tid for eksamen :	09.00 - 12.00
Oppgavesettet er på :	5 sider
Vedlegg :	intet
Tillatte hjelpemidler :	Alle trykte og skrevne

Les gjennom ***hele*** oppgavesettet før du begynner å løse den første oppgaven. Dersom du savner opplysninger i oppgaven, kan du selv legge dine egne forutsetninger til grunn og gjøre rimelige antagelser, så lenge de ikke bryter med oppgavens "ånd". Gjør i så tilfelle rede for disse forutsetningene og antagelsene.

Oppgave 1

a) Bruk Thompson's konstruksjon til å lage en NFA for det regulære uttrykket **(aa|b)*(a|cc)***

b) Skriv følgende NFA ut som et regulært uttrykk:



c) Gjør om NFAen fra pkt b) til en DFA.

Oppgave 2

Betrakt følgende grammatikk G_1:

```
E → S E | num
S → - S | + S | ε
```

hvor E og S er ikke-terminalsymboler, E er startsymbol og: **+, -, num** er terminalsymboler (med vanlig tolkning).

a) Beskriv kort hvordan setninger som G_1 kan produsere ser ut, og gi ett eksempel på en setning med 4 terminalsymboler.

b) Gi et regulært uttrykk som lager de samme setningene som G_1.

c) Gi et kort argument som bestemmer hvike(n) av følgende fem grupper G_1 hører med i:
   1. LR(0)
   2. SLR(1)
   3. LALR(1)
   4. LR(1)
   5. Ingen av de overstående.

   Hint: Finn ut om G_1 er entydig.

Vi skal nå se på en annen grammatikk G_2:

```
F → + F | - F | num
```

Hvor F er ikke-terminalsymbol (og startssymbol) og **+, -, num** igjen er terminalsymboler.

d) Du skal nå lage LR(0)-DFA-en for G_2 rett fra denne grammatikken hvor du har utvidet grammatikken med en ny produksjon F' → F (og hvor F' nå er startsymbolet). Nummerér hver av tilstandene.

e) Ut fra det svaret på d), angi med en kort begrunnelse hvilke(n) type grammatikk G_2 er (jfr. spørsmål c) ovenfor).

f) Lag parseringstabellen for G_2 ut fra den typen grammatikk den er.

g) Vis hvordan setningen: "- - 9" vil bli parsert ved å skrive opp, som i boka, stakk-innholdet og input for hver av skift- eller reduser-operasjonene du gjør under parseringen.

Oppgave 3

a) Det følgende er et fragment av en grammatikk for et språk med klasser:

class → **class** ***name*** *superclass* **{** *decls* **}**
decls → *decls* **;** *decl* | *decl*
decl → *variable-decl*
decl → *method-decl*
method-decl → *type* ***name*** **(** *params* **)** *body*
type → **int** |**bool** | **void**
superclass → ***name***

Ord i *kursiv* er meta-symboler, ord og tegn i **fet skrift** er terminal-symboler, mens ***name*** representerer et navn som scanneren leverer. Det kan antas at ***name*** har attributtet 'name'.

Metoder med samme navn som klassen er 'konstruktører', og det gjelder følgende regel: Konstruktører må være spesifisert med typen **void**

Lag semantiske regler for hver regel i følgende fragment av en attributtgrammatikk. Sett først opp hvilke attributter du trenger

Hint: Du kan klare deg uten en hel symboltabell.

Grammar Rule	Semantic Rule
class → ***class name {*** *decls* ***}***	
decls → *decls ; decl*	
decls → *decl*	
Decl → *variable-decl*	*Skal ikke fylles ut*
Decl → *method-decl*	
method-decl → *type* ***name (*** *params* ***)*** *body*	
Type → **int**	
Type → **bool**	
Type → **void**	

(skriv av tabellen og fyll den ut på et innføringsark; fyll ikke direkte inn på oppgavearket)

b) Anta ta vi har et språk med klasser og subklasser. Alle metoder er virtuelle, slik at de kan redefineres i subklasser.

Gitt følgende klassedefinisjoner:

```
class A {
  int i;
  void P {... AP ...};
  void Q {... AQ ...};
}

class B extends A {
  int j;
  void Q {... BQ ...};
  void R {... BR ...};
}

class C1 extends B {
  void P {... C1P ...};
  void S {... C1S ...};
}
```

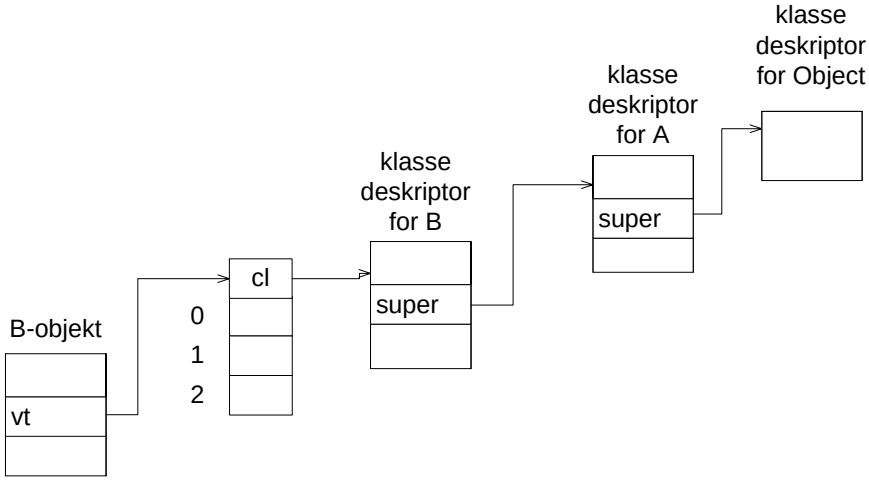
```
class C2 extends B {
  int k;
  void R {... C2R ...};
  void T {... C2T ...};
}
```

Vis hvordan objekter av klassene C1 og C2 vil være strukturert (layout) og tegn virtuell-tabellen for hver av klassene. Bruk navnene i metodekroppene over til å angi elementene i virtuell-tabellene.

c) Vi innfører nå muligheten for å kunne spesifisere en metode til å være **final**. Det skal bety, som i Java, at den ikke lenger er virtuell, dvs at den ikke kan redefineres i subklasser. Anta at vi i klassen B spesifiserer metoden Q til å være final. Må vi da endre på virtuell-tabellen for B-objekter? Begrunn svaret.

d) Vi innfører nå operatoren **`instanceof`** som i Java: Det boolske uttrykket '<refExpr> **`instanceof`** <class>' er True hvis objektet som <refExpr> peker på har en klasse som ikke er 'null', og som er klassen <class> eller en subklasse av klassen <class>, ellers False.

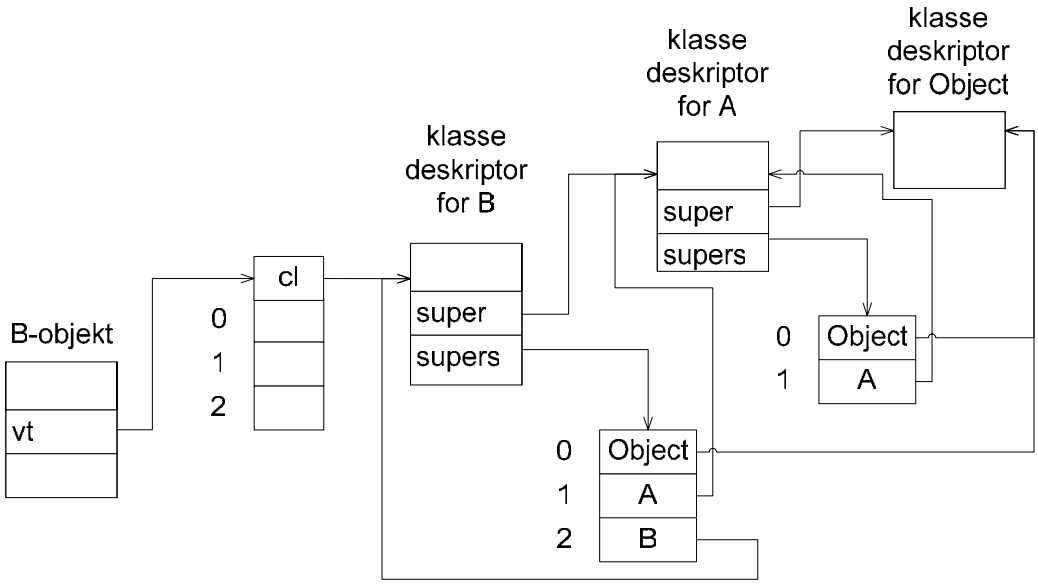
For å kunne implementere denne operatoren utvider vi virtuell-tabellen med en peker til klasse-deskriptoren, som det er en av for hver klasse i programmet. Klassedeskriptoren har da en variabel 'super', som peker til klasse-deskriptoren for superklassen. Klasser uten eksplisitt superklasse har den spesielle klasse Object som super. Eksemplet under viser dette for et objekt av klassen B:



Skisser algoritmen som beregner verdien av '<refExpr> instanceof <class>'.

e) For å effektivisere testen på **`'instanceof'`** innfører vi (inspirert av display/kontekst vektor for nestede blokker) at en klassedeskriptor har en tabell 'supers', som inneholder alle superklassene for klassen samt klassen selv. Denne tabellen har som indeks klassens 'subklassenivå' startende med 0 for Object, 1 for rotklassen i et subklassehierarki, 2 for neste nivå, osv. I vårt eksempel har klassen A subklasseniv1, B har 2, C1 og C2 har begge 3.

For klassene A og B ser klassedeskriptorene med supers-tabellene slik ut:



Du skal forklare hvordan denne tabellen kan effekstivisere **`instanceof`**-testen, og til å illustrere dette skal vi innføre ytterligere to klasser:

class C11 extends C1 { ...}
class C21 extends C2 {...}

Lag supers-tabellene for klassedeskriptorene for C11 og C21, og vis hvordan følgende tester gjøres:

```
rC11 = new C11;
rC11 instanceof C1   (1)
rC11 instanceof C2   (2)
```

---------- o 0 o ----------

Slutt på oppgavesettet – lykke til!

Arne Maus og Birger Møller-Pedersen