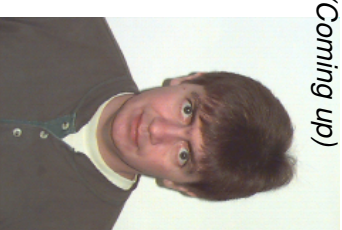


# Syntaks og semantik

Lektion 13

22 april 2008

- mundtlig
- 7-trins-skala
- 10 eksamensspørgsmål, **kendt på forhånd**. (*Coming up*)
- 20 minutters forberedelse
- 20 minutters eksamen
- hjælpemidler: ingen computer, ingen mobiltelefon, **ingen slides**
- ekstern censor: Jens Chr. Godskesen, ITU  
<http://www.itu.dk/~jcg/>
- syntaks- og semantikopgaven plus 8 andre
- de andre: incl. **prøveopgave**
- prøveopgaven dækker **kun en del af** opgavens pensum
- prøveopgavens besvarelse indgår **som en del af** en samlet præsentation



3/8

## Forord

- 1 Eksamen
- 2 Eksamenspensum
- 3 Eksempel på prøveopgave
- 4 Semantikopgaven

- *Sjipser* kapitel **1**, side 31 (31) til 82 (82)
- *Sjipser* kapitel **2**, side 101 (99) til 108 (106) og 111 (109) til 129 (127)
  - dvs. alt bortset fra Chomsky-normalformen
- *Hüttel* kapitel **3**, side 35 til 49
- *Hüttel* kapitel **4**, side 51 til 66 og 69 til 70
  - dvs. alt bortset fra sætning 4.13, lemma 4.14 og tilhørende beviser
- *Hüttel* kapitel **5**, side 73 til 76 og 78 til 87
  - dvs. alt bortset fra anden halvdel af beviset for sætning 5.2
- *Hüttel* kapitel **6**, side 89 til 98 og 102 til 103
  - dvs. alt bortset fra afsnittene 6.6.1 og 6.6.2
- *Hüttel* kapitel **7**, side 105 til 111
  - dvs. afsnittene 7.1 til 7.4
- *Hüttel* kapitel **14**, side 187 til 201
  - dvs. afsnittene 14.1 til 14.6

**Opgave 1** fra sidste år:

Lad  $L$  være sproget givet ved det regulære udtryk  $(a^*b)^*$ .

- 1 Konstruer en nondeterministisk endelig automat  $A$  som genkender  $L$ .
- 2 Forenkl evt.  $A$  så den kun har tre tilstande, og konverter  $A$  til en deterministisk endelig automat.

---

Emner som berøres af denne eksamensopgave:

- deterministiske og nondeterministiske endelige automater
- regulære udtryk
- konverteringer mellem de tre
- regulære sprog og deres lukningsegenskaber

5/8

Spørgsmål?

**Bof**, variant af **Bip** med *funktioner* ('udtryksprocedurer') i stedet for procedurer:

- én call-by-value-parameter
- returnerer en værdi
- ⇒ funktionskald er ikke *kommando*, men **aritmetisk udtryk**
- eksempel:

```
func f (x) is
  begin
    var i := 1;
    f := x + i
  end
...
y := f (17) + 4
```

- dvs. værdien returneres ved at tilskrive den til en ('alias') variabel med samme navn som funktionen

7/8

- ny syntaktisk kategori:
  - $f \in \mathbf{Fnavne}$  – funktionsnavne
  - vi sætter  $\mathbf{Fnavne} = \mathbf{Var}$
- nye opbygningsregler:

**Erkf:**  $D_F ::= \text{func } f(x) \text{ is } S; D_F \mid \varepsilon$   
**Kom:**  $S ::= \dots \mid \text{begin } D_V D_F S \text{ end}$   
**Aud:**  $a ::= \dots \mid f(a)$

6/8

8/8