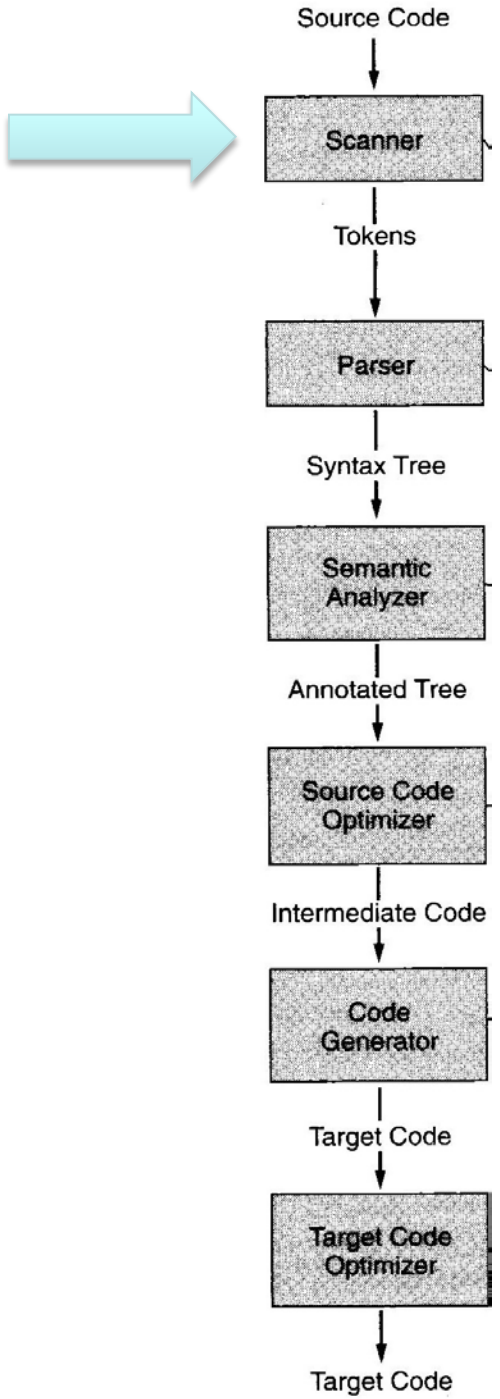
Skanning – del II

Kapittel 2



INF 5110 - 2013

Overgang fra NFA (M) til DFA (M')

$(\overline{M})$

"The subset construction"

- IDÉ VED KONSTRUKSJONEN
  - Hver tilstand i M' er definert ved en mengde av M-tilstander
  - Gitt en (sub-)streng α. I M' skal denne føre til tilstanden S definert ved *alle de M-tilstander som α kan føre fram til* ved forskjellige ikke-deterministiske valg i M.

Definisjoner

Gitt en mengde (et ”utplukk”) S av M-tilstander.

- S_a:
  - alle M-tilstander som kan nås fra S ved å følge én a-kant.

- ε-tillukningen av S:
  - S selv
  - alle M-tilstander som kan nås ved (en eller flere) ε-kanter fra S

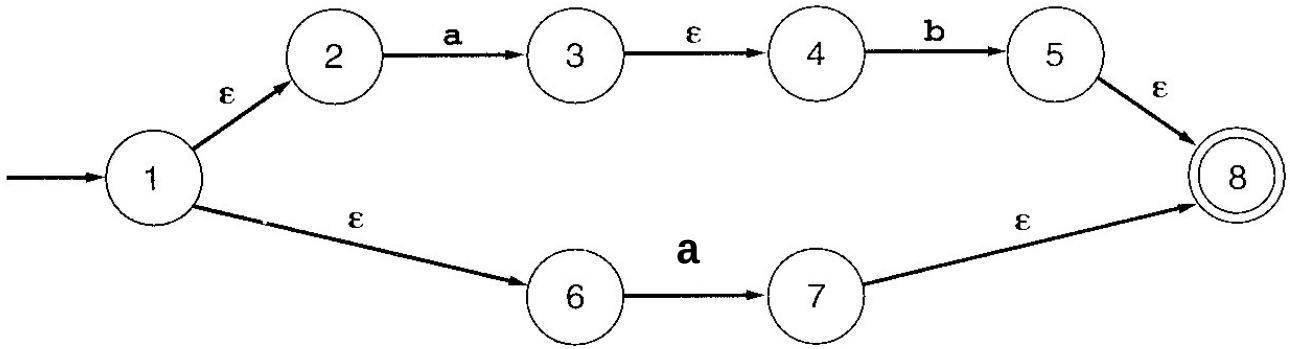
Algoritme fra NFA (M) til DFA (M')

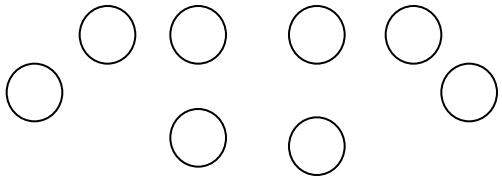
1. Lag start-tilstanden i M' som ε-tillukningen av {start-tilstanden i M}
2. Om S er en tilstand i M', så gjelder:

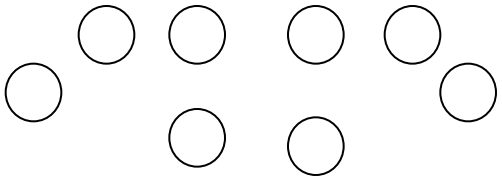
(Må eventuelt opprette ny M'-tilstand)

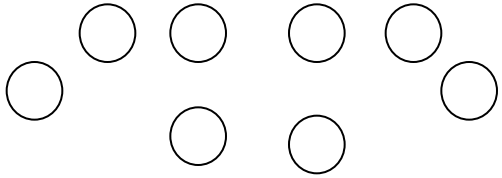
3. Gjenta steg 2 for alle S i M' og alle a i alfabetet til M' ikke får flere nye tilstander.
4. De aksepterende tilstander i M' er de som inneholder minst én av de aksepterende tilstander i M.

Eksempel 2.16 ab | a

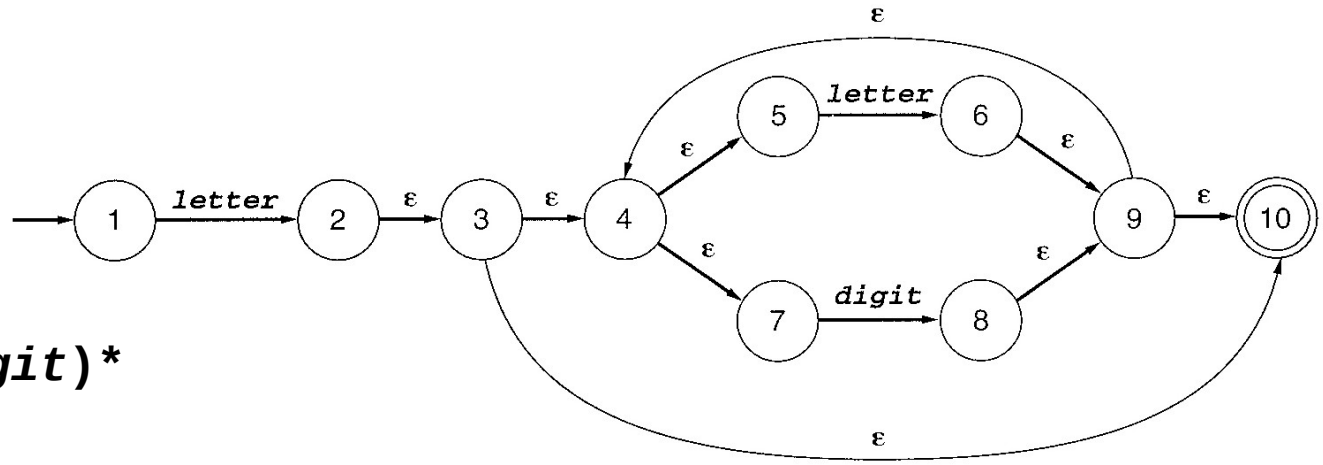








Eksempel 2.17



letter* (*letter* | *digit*)

Minimalisering av DFA

- Ved automatisk konstruksjon av DFA-er (f.eks. gjennom Thompson-konstr.) får vi ofte unødvendig mange tilstander.
- Skal se på algoritme som slår sammen tilstander så mye som mulig (uten å forandre språket som aksepteres).
- For denne algoritmen gjelder (vises ikke):
  - Alle DFA-er for samme språk transformeres til samme DFA.
  - Denne har så få tilstander som mulig.
- Som en bieffekt får vi derved en algoritme for å svare på:
  - Gitt to DFA-er. Aksepterer de samme språk?
  - Gitt to regulære uttrykk. Beskriver de samme språk?

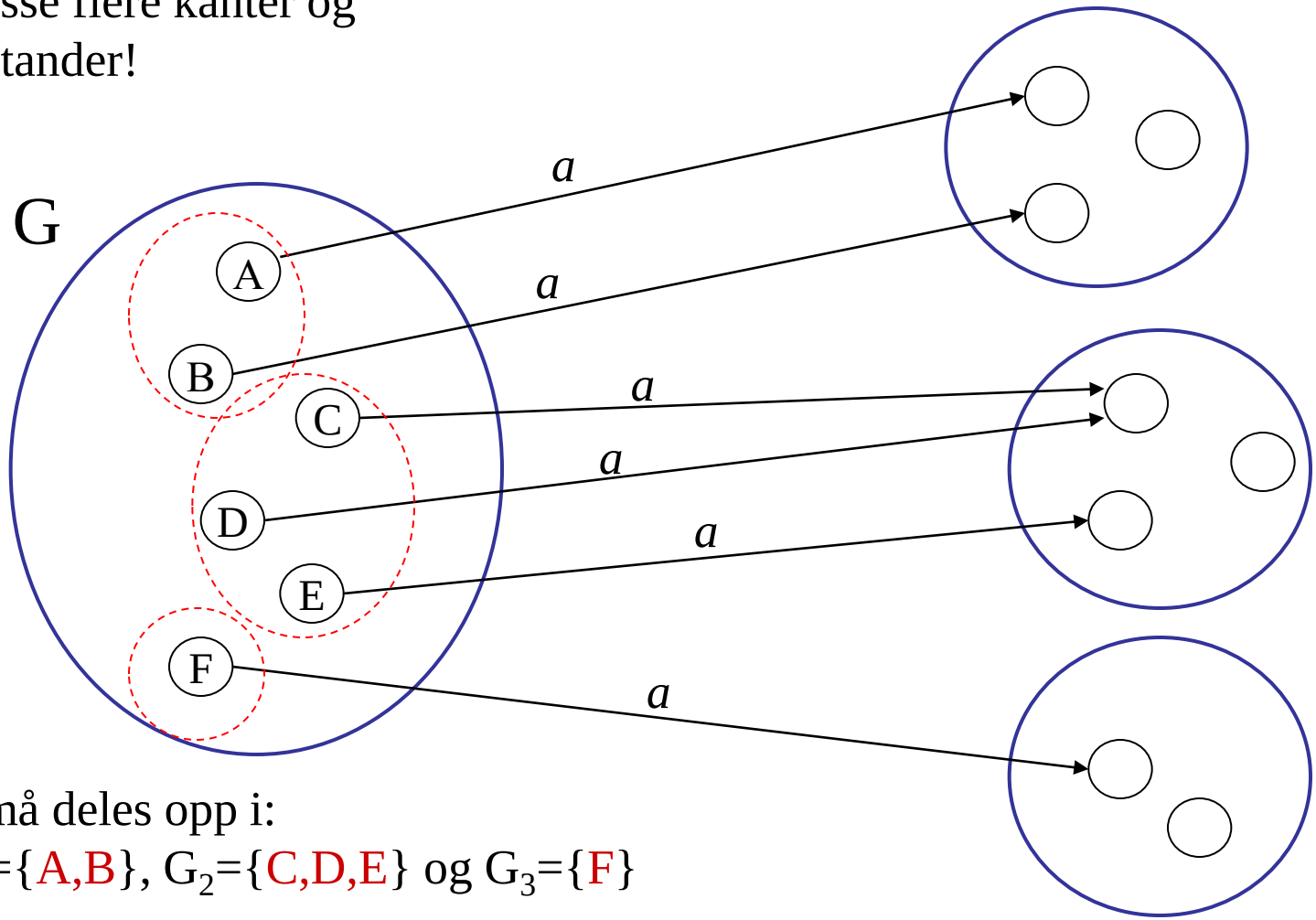
Algoritme for å minimalisere DFA

1. Utvid til ”fullstendig DFA” M, ved eventuelt å legge til ”feil-tilstand”.
2. Grupper sammen tilstandene i M så optimistisk som mulig:
   - Alle aksepterende tilstander slås sammen.
   - Alle ikke-aksepterende tilstander slås sammen.
3. Oppsplitting: Se på gruppe G = $\{s_1, s_2, \ldots, s_n\}$ og et symbol a.
   - Dersom $t(s_1,a), t(s_2, a), \ldots, t(s_n, a)$ alle er i samme gruppe er alt OK.
   - Ellers må man splitte opp G i $G_1, G_2, \ldots, G_m$ slik at kravet over gjelder for hver G_i.
4. Gjenta steg 3 for alle (sammenslåtte) G og alle symboler a, inntil:
   - Enten oppsplittingen er blitt fullstendig (vi er tilbake til den opprinnelige DFA).
   - Eller det ikke blir mer oppsplitting.
5. Sett på transisjons-overgangene mellom de nye sammenslåtte tilstandene. Disse er nå entydig definerte.

Oppsplittingen (steg 3):

NB1: Bare *a*-kanter fra G-tilstander er tegnet. Det er masse flere kanter og tilstander!

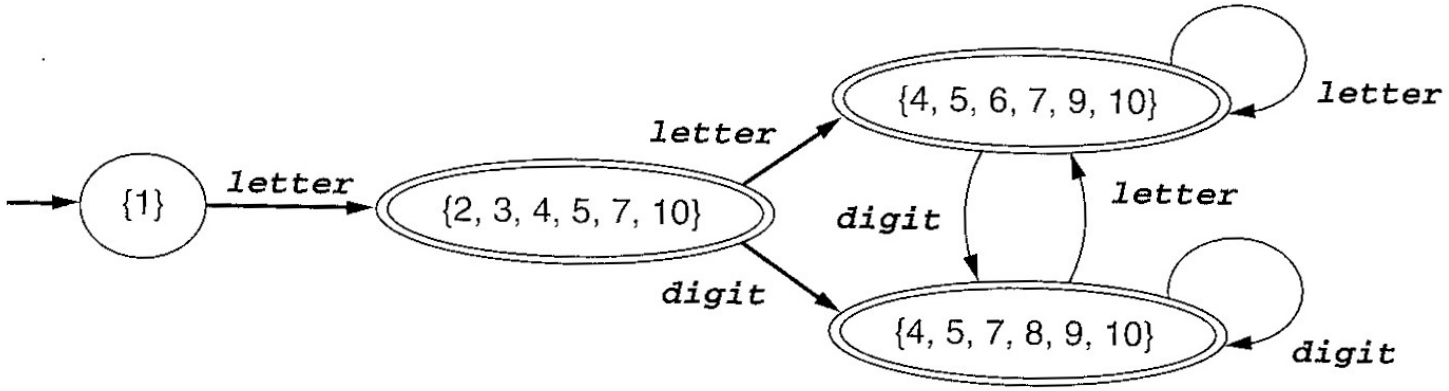
NB2: En av disse kan være G selv:



G må deles opp i:
G_1={A,B}, G_2={C,D,E} og G_3={F}

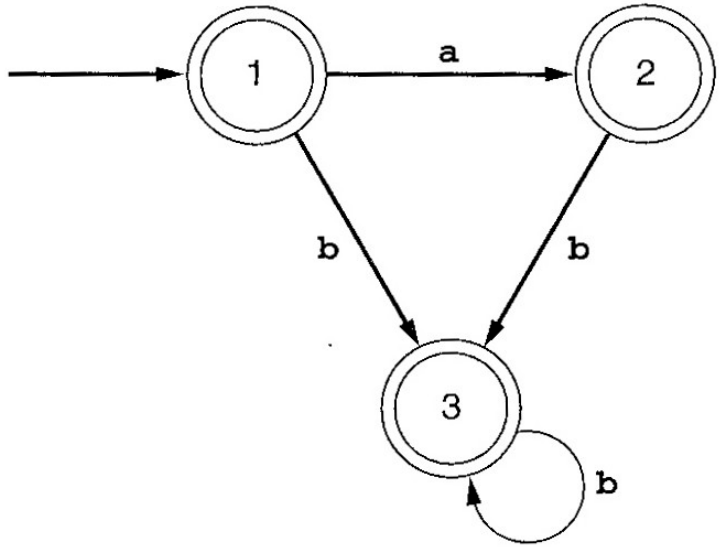
Eksempel 2.18

letter* (*letter* | *digit*)



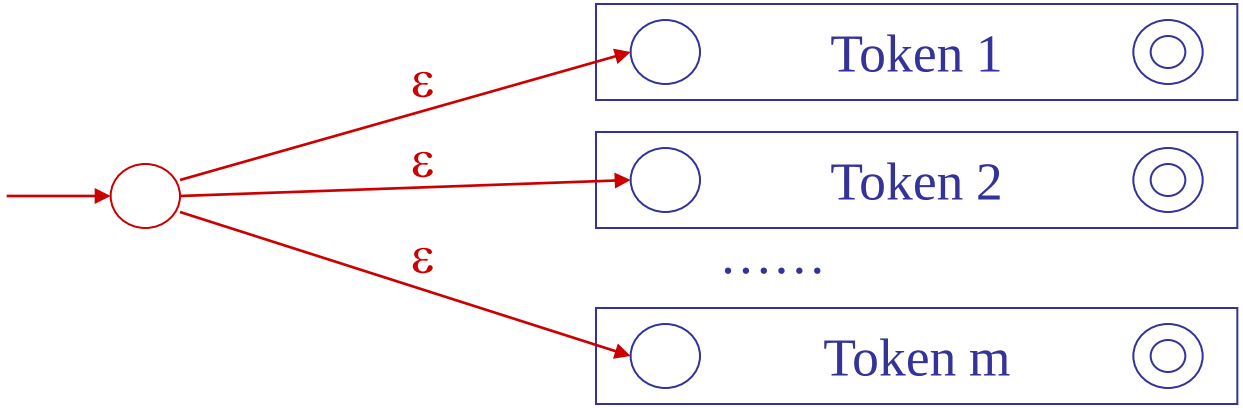
Eksempel 2.19

(a | ε) b*



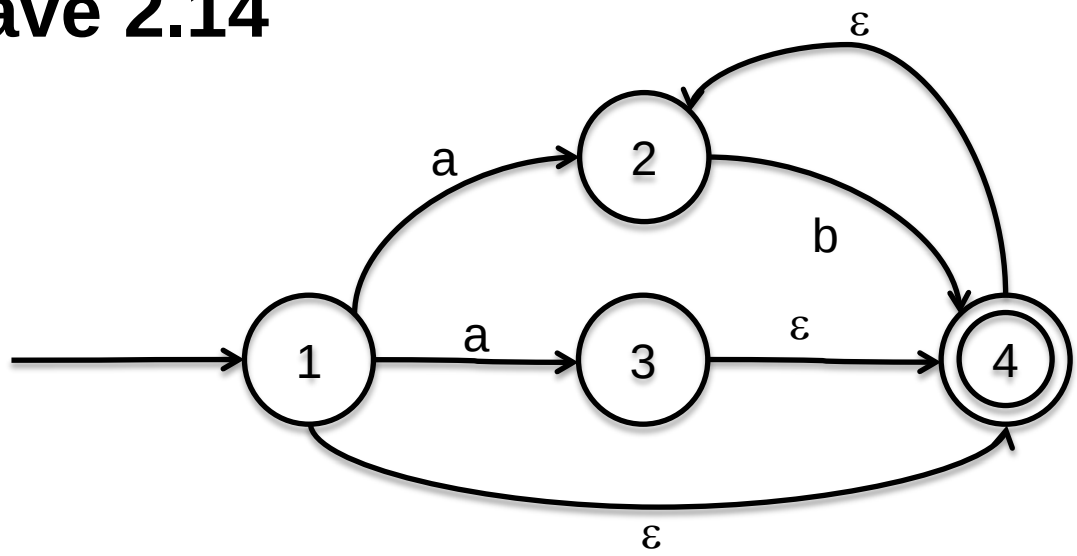
Hovedidé ved Lex/Flex og liknende

- Bruker setter opp regulær definisjon av hver token-klasse (samt tilhørende aksjoner som skal utføres når slikt token er funnet).
- Disse gjøres om til NFA-er (f.eks. ved Thompson konstr.)
- Disse enkelt-NFA-ene settes sammen til stor NFA slik:



- Denne gjøres om til DFA ved ”subset construction”.
- Gjør eventuelt minimalisering av DFA-en (med ”litt forsiktighet”, slik at token-klasser kan skilles).
- Implementer DFA-en, vanligvis tabell-drevet.

Oppgave 2.14



Oppgave 2.16