# Tilstandsmaskiner

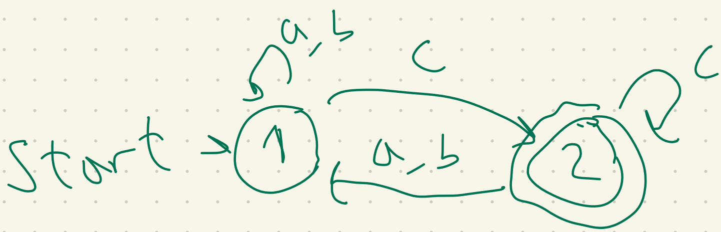
Endelige tilstandsmaskiner er modeller av enkle maskiner med begrenset funksjonalitet.

En slik maskin kan være i et gitt antall ulike tilstander. En av dem er starttilstanden, og en delmengde av dem er såkalt aksepterende. Maskinen

leser inn et ord over ett gitt alfabet som input, og kan endre tilstand avhengig av hva den leser.

Vi sier at maskinen godkjenner eller gjenkjenner en streng/ord over alfabetet om maskinen slutter i en aksepterende tilstand etter å ha lest strengen.

<u>Eks</u> La $A=\{a,b,c\}$. Si at vi vil finne en endelig tilstands-maskin som gjenkjenner/aksepterer nøyaktig ordene over $A$ som slutter på $c$



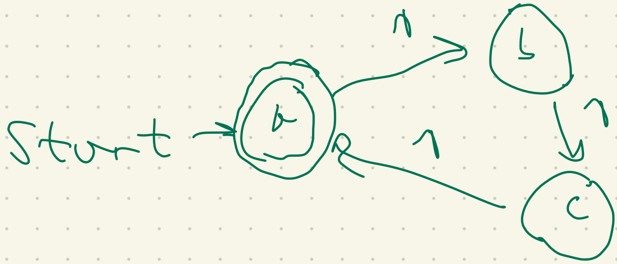

Vi indikerer start-tilstanden med en pil "start →". Aksepterende tilstander indikeres med dobbel rand.

Merk: For hver tilstand må vi for hver bokstav i alfabetet spesifisere hva vi gjør, altså hvilken tilstand vi flytter oss til, når vi i den tilstanden leser den bokstaven.

Eks La $A = \{1\}$. Si at vi ønsker å finne en tilstandsmaskin som gjenkjenner nøyaktig strengene i $A^*$ som har lengde som kan deles på 3. Vi kan skrive dette språket som

$$\{1^{3n} \in A^* \mid n \in \mathbb{N}\}$$

En måte å bygge en slik maskin på er å holde styr på "resten" etter vi har delt antallet tegn lest så langt på 3. Hvis denne resten er null har vi lengde som er et multiplum av 3.

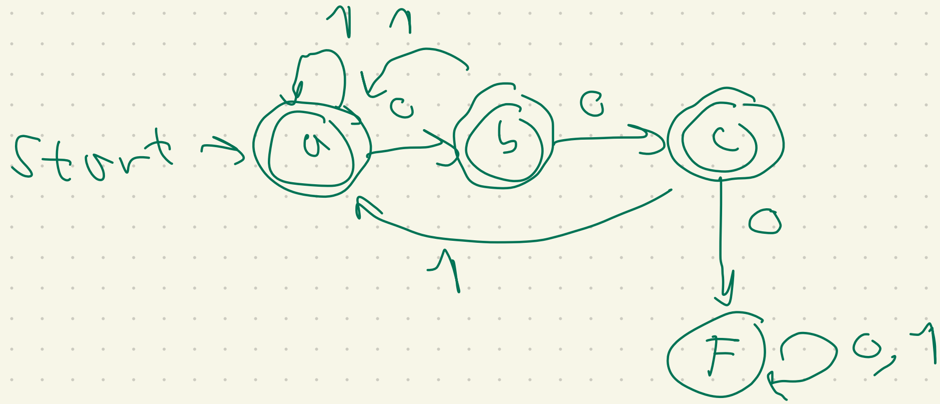



a ~ 0 i rest $\Rightarrow$ aksepterende

b ~ 1 i rest $\Rightarrow$ <u>ikke</u> aksepterende

c ~ 2 i rest $\Rightarrow$ <u>ikke</u> aksepterende.

<u>Eks</u> La $A = \{0, 1\}$. Si at vi vil finne en tilstandsmaskin som gjenkjenner nøyaktig de strenger i A som <u>ikke</u> inneholder 3 etterfølgende nullere. Plan: Hold styr på antall nullere satt på rekke.




Følgende teorem er spesielt viktig om endelige tilstandsmaskiner fordi det kobler dem til regulære språk.

Teorem La A være et alfabet.
Et formelt språk $S \subseteq A^*$ er regulært hvis og bare hvis det eksisterer en endelig tilstandsmaskin som gjenkjenner blant alle ord i $A^*$ nøyaktig dem som er i S.
Kalles "Kleene's teorem"

Ideene i beviset til dette teoremet kommer vi tilbake til senere.

Korollar/Konsekvens: Si at vi ønsker å bevise at et språk er regulært. Vi kjenner en direkte måte å gjøre dette på:

* Konstruer språket med regulære uttrykk.

Teoremet over gir oss en indirekte metode: finn en endelig tilstands-maskin som gjenkjenner nøyaktig det språket.

Teoremet impliserer også at om det ikke finnes noen endelig tilstandsmaskin som gjenkjenner nøyaktig ordene i et gitt språk, så er det språket ikke regulært.

Eks La $S \subseteq A^*$ være regulært.
Da er $\bar{S} = \{\sigma \in A^* \mid \sigma \notin S\}$ regulært.

Bevis: Velg en tilstandsmaskin, $M$, som gjenkjenner nøyaktig strengene i $S$.

Bygg nå en tilstandsmaskin $M'$ som er lik $M$ uten om at:

- Aksepterende tilstander i $M$ er ikke aksepterende i $M'$.
- Ikke-aksepterende tilstander i $M$ er aksepterende i $M'$.

Da gjenkjenner $M'$ nøyaktig de ord i $A^*$ som ikke er i $S$, altså $\bar{S}$

Da det finnes en tilstandsmaskin som gjenkjenner $\bar{S}$, er språket regulært.

<u>Eks</u>! Vi kan også argumentere andre veien: La for eksempel T være en tilstandsmaskin som kjenner igjen nøyaktig språket S.

Da er S regulært, så $S^{100}$ er regulært.

(Husk: $L, R \mapsto LR$. er en lovlig operasjon på regulære språk)

Dermed finnes det en tilstandsmaskin som gjenkjenner nøyaktig strengene i $S^{100}$.

Merk: Å bygge denne maskinen fra

Ter i prinsippet mulig, men kan bli veldig stort og komplisert. Enklere å do argumentere "via" regulære språk.

**Begrensninger** – hva kan og kan ikke tilstandsmaskiner gjøre?

Merk: En tilstandsmaskin har veldig begrenset med minne.

Når en tilstandsmaskin har lest den første delen av en streng, husker den ikke annet enn hvilken tilstand

$$\underbrace{abbccb\ldots\ldots}\ a\ b\ldots\ldots$$

Hvor er jeg nå?

den er i nå.

Altså: Eneste minne vi har er det vi kan enkode i et endelig antall tilstander.

Teorem: La $S = \{10, 1100, 111000, \dots\}$
$= \{1^n 0^n \mid n \geq 1, n \in \mathbb{N}\}$

Det finnes ingen tilstandsmaskin som gjenkjenner nøyaktig de bitstrengene som ligger i $S$.

Bevis:

En maskin som skal gjenkjenne nøyaktig bitstrengene i S må kunne skille på for eksempel

111000 ✓

1111000 ✗

Maskinen må har være i 2 ulike tilstander. Den må enkelt sagt kunne "telle" antall 1'ere.

Anta for eksempel at du påstår at du har funnet en tilstandsmaskin med 4 tilstander som gjør jobben, altså å nøyaktig gjenkjenne S. La maskinen kalles T.

Betrakt nå de fem strengene

1
11
111
1111
11111

Fra duehull-prinsippet må det finnes to strenger i listen oven slik at T er i samme tilstand etter å ha lest dem.

Si at de to strengene er 11 og 1111 (det viktige er at de er ulike).

T gjenkjenner alt i S, så T gjenkjenner derfor

11 00

1111 0000

Da $T$ er i samme tilstand etter å ha lest 11 og 1111, vil $T$ <u>også</u> gjenkjenne

11 0000 (!!!)

1111 00

Disse er <u>ikke</u> i $S$, og skal ikke gjenkjennes $\Rightarrow$ $T$ gjør noe feil.

Merk: Ikke noe spesielt med 5 tilstander. Generelt, så lenge vi har endelig mange tilstander får vi samme problem. Litt mer presist: Har vi $n$ tilstander, ser vi på de $n+1$ strengene

$a_1$: 1

$a_2$: 1 1

$a_3$: 1 1 1

$\vdots$

$a_{n+1}$: $\underbrace{1\,1\,1\,\ldots\,1}_{n+1}$

Duehollprinsippet gir oss da at det finnes to ulike tall $i \neq j$ mellom 1 og $n+1$ slik at den gitte maskinen er i samme tilstand etter å ha lest $S_i$ som etter å ha lest $S_j$.

La $b_k = \underbrace{000\ldots0}_{k \text{ ganger}} = 0^k$

Hvis maskinen gjenkjenner alt i $S$, gjenkjenner den $a_i b_i$ og $a_j b_j$, men også $a_i b_j$ og $a_j b_i$ (!).

Disse to strengene er utenfor S.

$\Rightarrow$ Ingen endelig tilstandsmaskin gjenkjenner nøyaktig strengene i S.

Hva betyr dette i praksis?

Betrakt følgende Python-skript:

S.py:

```
s=str(input())
p = s.find('0')
ans=(2*p==len(s))
print(ans)
```

Kjører vi nå S.py i terminalen vil programmet gjenkjenne strengene i S, og bare disse

Eksempel:

```
> python3 S.py
> 1110000
> False
```

Merk: En datamaskin med $n$ bits minne kan bare lagre $2^n$ ulike tilstander.

Vi kan på en ekte datamaskin derfor ikke bruke programmet over som ønsket på <u>alle</u> mulige bitstrenger.

Om vi har 4 bit med minne ville vi fått samme problem som en tilstandsmaskin med $2^4 = 16$ tilstander.

Om vi har 8 gb = 64 gigabit.

$\approx 60 \times 10^9$ bits

ville vi aldri, uansett valgt implementering, klart å skille mer enn $2^{60 \cdot 10^9}$

Strenger fra hverandre.
Veldig stort i praksis, men ikke uendelig.

Altså: Ekte maskiner med $n$ bits minne er ikke noe bedre enn Endelige tilstandsmaskiner med $2^n$ tilstander.

(i praksis: Bedre å bygge eks. 128 bits minne enn astronomiske $2^{128}$ tilstander....)

I teoretisk datavitenskap ser vi derfor for oss at datamaskiner kjører på uendelig mye minne.
Da kan vi trygt si:

Endelige tilstandsmaskiner kan ikke gjøre alt det datamaskiner generelt kan gjøre.

Det finnes også formelle språk $S \subseteq \{0,1\}^*$ som datamaskiner ikke evner å nøyaktig kjenne igjen.