Stockholms matematiska cirkel
Datorernas matematik

www.math-stockholm.se/cirkel

16.00 – 16.15: Välkomsthäng
16.15 – 17.15: Föreläsning

Om Cirkeln

- 7 föreläsningar, 7 övningstillfällen
- Övning 7 sker på distans.
- Schema, program, kartor osv finns på hemsidan:

www.math-stockholm.se/cirkel

Datorernas matematik

1. (19 sep) Vad är matematik, egentligen?
2. (10 okt) Hur kan en dator räkna?
3. (7 nov) Tal med decimaler
4. (12 dec) Tärningen är kastad
5. (5 feb) Formella språk
6. (5 mars) Tillståndsmaskiner
7. **(26 mars) Tillståndsmaskinernas språk**

Kapitel 7 – Inledning

Förra kapitlet:

- Deterministiska tillståndsmaskiner (DTM)

Kapitel 7 – Inledning

Förra kapitlet:

- Deterministiska tillståndsmaskiner (DTM)
- Icke-deterministiska tillståndsmaskiner (ITM)

Kapitel 7 – Inledning

Förra kapitlet:

- Deterministiska tillståndsmaskiner (DTM)
- Icke-deterministiska tillståndsmaskiner (ITM)
- Delmängdskonstruktionen

Kapitel 7 – Inledning

Förra kapitlet:

- Deterministiska tillståndsmaskiner (DTM)
- Icke-deterministiska tillståndsmaskiner (ITM)
- Delmängdskonstruktionen

ITM och DTM avgör samma språk. Men vilka språk är det?

Kapitel 7 – Inledning

Vi ska bevisa att ett språk kan avgöras av en tillståndsmaskin om och endast om det är reguljärt. Två delar:

Kapitel 7 – Inledning

Vi ska bevisa att ett språk kan avgöras av en tillståndsmaskin om och endast om det är reguljärt. Två delar:

1. För varje reguljärt språk L så finns det en tillståndsmaskin M vars språk är L.

Kapitel 7 – Inledning

Vi ska bevisa att ett språk kan avgöras av en tillståndsmaskin om och endast om det är reguljärt. Två delar:

1. För varje reguljärt språk L så finns det en tillståndsmaskin M vars språk är L.
2. För varje tillståndsmaskin M så finns det ett reguljärt språk L så att M:s språk är L.

Kapitel 7 – Inledning

Vi ska bevisa att ett språk kan avgöras av en tillståndsmaskin om och endast om det är reguljärt. Två delar:

1. För varje reguljärt språk L så finns det en tillståndsmaskin M vars språk är L.
2. För varje tillståndsmaskin M så finns det ett reguljärt språk L så att M:s språk är L.

Vi gör dessa fall var för sig.

Kapitel 7.1 – Reguljära språk och maskiner

Repetition:

Reguljära språk över ett alfabet Σ definieras rekursivt.

Kapitel 7.1 – Reguljära språk och maskiner

Repetition:

Reguljära språk över ett alfabet Σ definieras rekursivt.

Basfall $\emptyset$ och $\sigma \in \Sigma$ är reguljära språk.

Kapitel 7.1 – Reguljära språk och maskiner

Repetition:

Reguljära språk över ett alfabet Σ definieras rekursivt.

Basfall $\emptyset$ och $\sigma \in \Sigma$ är reguljära språk.

Rekursion Om L_1 och L_2 är reguljära språk, är $L_1 L_2$, $L_1 \cup L_2$ och L_1^* reguljära språk.

Kapitel 7.1 – Reguljära språk och maskiner

Repetition:

Reguljära språk över ett alfabet Σ definieras rekursivt.

Basfall $\emptyset$ och $\sigma \in \Sigma$ är reguljära språk.

Rekursion Om L_1 och L_2 är reguljära språk, är $L_1 L_2$, $L_1 \cup L_2$ och L_1^* reguljära språk.

Exempel: $ab \cup ba = \{ab, ba\}$,

Kapitel 7.1 – Reguljära språk och maskiner

Repetition:

Reguljära språk över ett alfabet Σ definieras rekursivt.

Basfall $\emptyset$ och $\sigma \in \Sigma$ är reguljära språk.

Rekursion Om L_1 och L_2 är reguljära språk, är $L_1 L_2$, $L_1 \cup L_2$ och L_1^* reguljära språk.

Exempel: $ab \cup ba = \{ab, ba\}$, $ab^* a = \{ab^n a : n = 0, 1, 2, \ldots\}$

Kapitel 7.1 – Reguljära språk och maskiner

Sats: *Alla reguljära språk kan avgöras av en tillståndsmaskin.*

Vi gör ett induktionsbevis. Det består av två steg.

Kapitel 7.1 – Reguljära språk och maskiner

Sats: *Alla reguljära språk kan avgöras av en tillståndsmaskin.*

Vi gör ett induktionsbevis. Det består av två steg.

Basfall Språken $\emptyset$ och σ avgörs av tillståndsmaskiner.

Kapitel 7.1 – Reguljära språk och maskiner

Sats: *Alla reguljära språk kan avgöras av en tillståndsmaskin.*

Vi gör ett induktionsbevis. Det består av två steg.

Basfall Språken $\emptyset$ och σ avgörs av tillståndsmaskiner.

Induktionssteg Om L_1 och L_2 avgörs av tillståndsmaskiner, så avgörs $L_1 L_2$, $L_1 \cup L_2$ och L_1^*.

(whiteboard)

Kapitel 7.1 – Reguljära språk och maskiner

Beviset ger en algoritm för att konstruera en ITM som avgör ett reguljärt uttryck.

Kapitel 7.1 – Reguljära språk och maskiner

Beviset ger en algoritm för att konstruera en ITM som avgör ett reguljärt uttryck.

Exempel: $ab^*a \cup b$

Kapitel 7.2 – Kleenes algoritm

Alla reguljära språk kan avgöras av en tillståndsmaskin. Men finns det tillståndsmaskiner som avgör språk som inte är reguljära?

Kapitel 7.2 – Kleenes algoritm

Alla reguljära språk kan avgöras av en tillståndsmaskin. Men finns det tillståndsmaskiner som avgör språk som inte är reguljära?

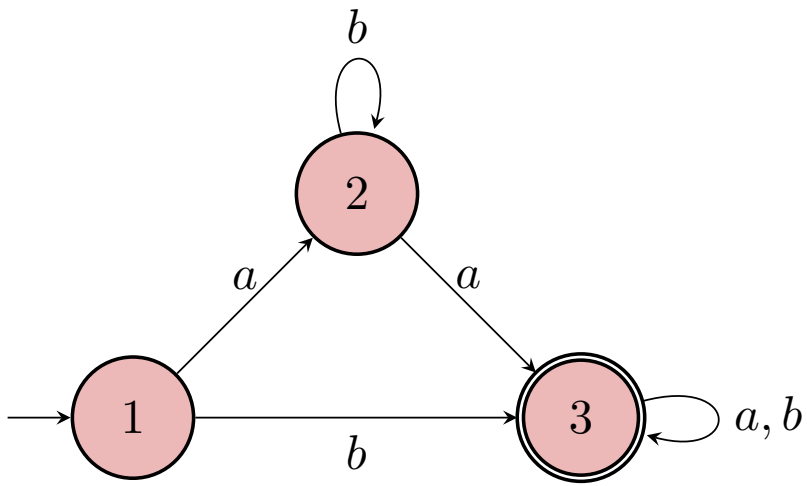
Svaret är **nej**. Om M är en tillståndsmaskin, så finns det ett reguljärt uttryck som motsvarar språket som M avgör.

Kapitel 7.2 – Kleenes algoritm

En maskins språk består av alla ord som driver maskinen från ett starttillstånd till ett accepterande tillstånd.

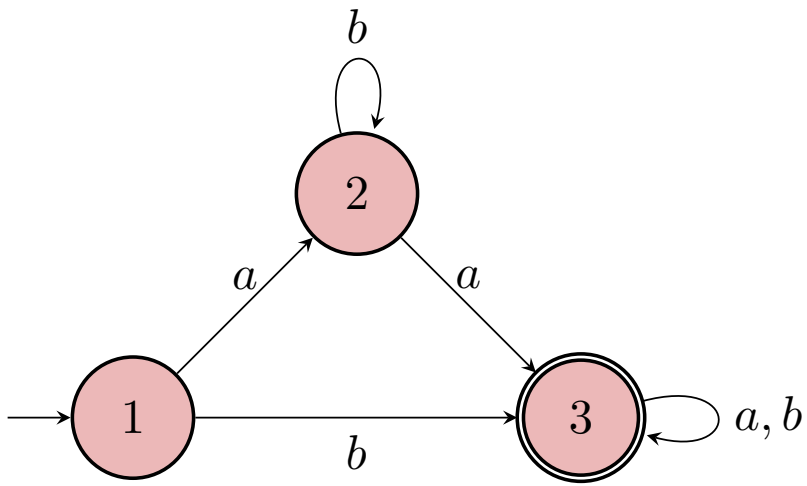
Kapitel 7.2 – Kleenes algoritm

En maskins språk består av alla ord som driver maskinen från ett starttillstånd till ett accepterande tillstånd. **Exempel**:



Kapitel 7.2 – Kleenes algoritm

En maskins språk består av alla ord som driver maskinen från ett starttillstånd till ett accepterande tillstånd. **Exempel**:



Maskinens språk är alla ord som driver maskinen från 1 till 3.

Kapitel 7.2 – Kleenes algoritm

Ett ord driver maskinen från 1 till 3 på två sätt: antingen direkt från 1 till 3, eller via 2.

Kapitel 7.2 – Kleenes algoritm

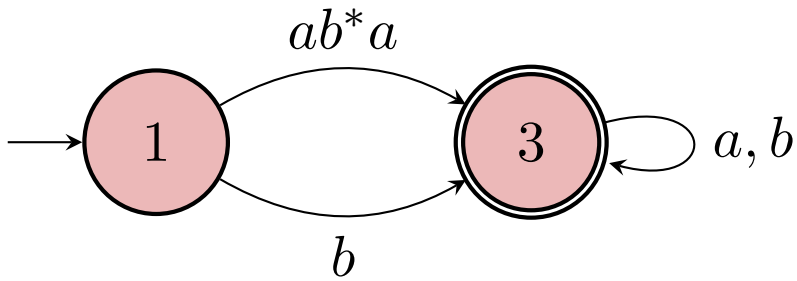
Ett ord driver maskinen från 1 till 3 på två sätt: antingen direkt från 1 till 3, eller via 2.

Man kan illustrera detta genom att plocka bort tillstånd 2, och ersätta med en ab^*a-övergång.

Kapitel 7.2 – Kleenes algoritm

Ett ord driver maskinen från 1 till 3 på två sätt: antingen direkt från 1 till 3, eller via 2.

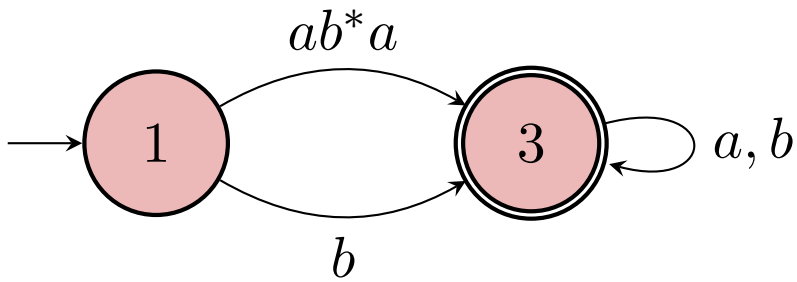
Man kan illustrera detta genom att plocka bort tillstånd 2, och ersätta med en ab^*a-övergång.



Kapitel 7.2 – Kleenes algoritm

Ett ord driver maskinen från 1 till 3 på två sätt: antingen direkt från 1 till 3, eller via 2.

Man kan illustrera detta genom att plocka bort tillstånd 2, och ersätta med en ab^*a-övergång.



Hur kan man göra detta generellt?

Kapitel 7.2 – Kleenes algoritm

Numrera tillstånden 1 till n och välj två tillstånd i och j. Hur kan ett ord driva maskinen från i till j?

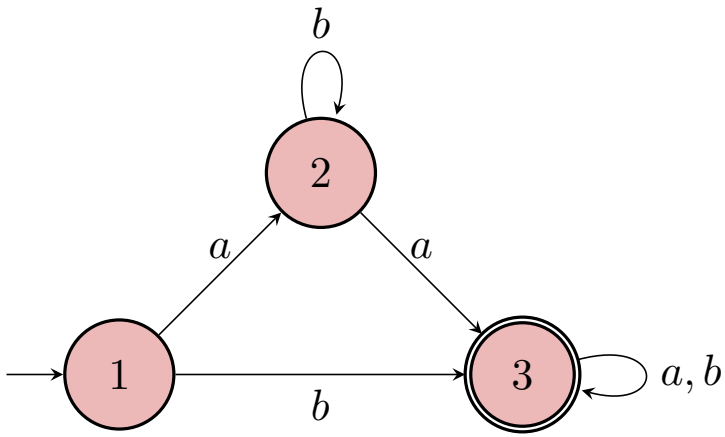
Kapitel 7.2 – Kleenes algoritm

Numrera tillstånden 1 till n och välj två tillstånd i och j. Hur kan ett ord driva maskinen från i till j? Idén är att titta på tillstånden **mellan** i och j.

Kapitel 7.2 – Kleenes algoritm

Numrera tillstånden 1 till n och välj två tillstånd i och j. Hur kan ett ord driva maskinen från i till j? Idén är att titta på tillstånden **mellan** i och j.

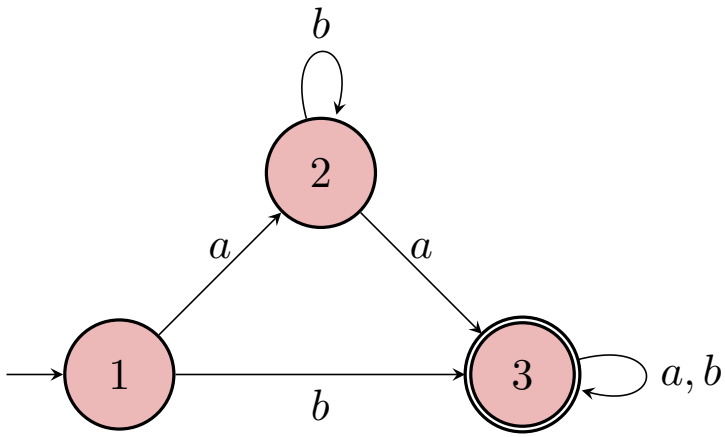
Exempel:



Kapitel 7.2 – Kleenes algoritm

Numrera tillstånden 1 till n och välj två tillstånd i och j. Hur kan ett ord driva maskinen från i till j? Idén är att titta på tillstånden **mellan** i och j.

Exempel:



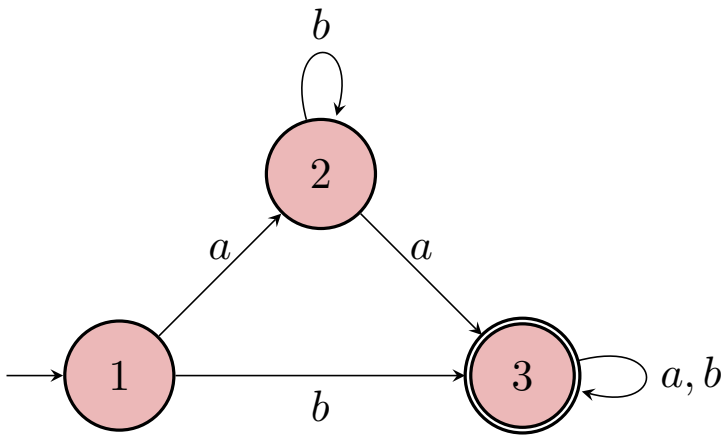
Ordet *abbab* driver maskin enligt $1 \to 2 \to 2 \to 2 \to 3 \to 3$, och passerar tillstånden 2 och 3.

Kapitel 7.2 – Kleenes algoritm

Låt u_{ij}^k beteckna samlingen av ord som driver maskinen från i till j och bara passerar tillstånd som är mindre eller lika med k.

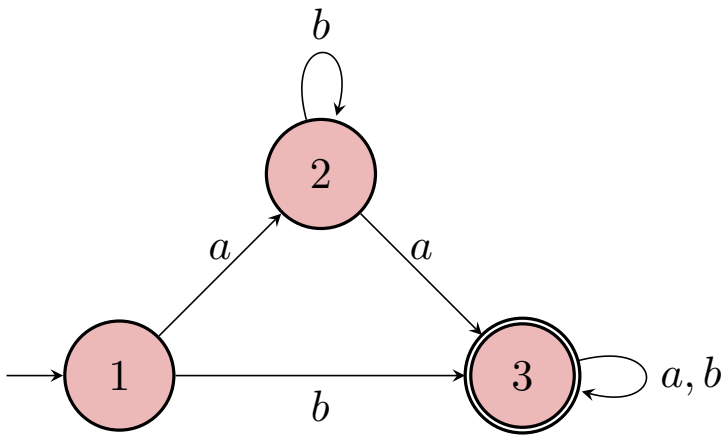
Kapitel 7.2 – Kleenes algoritm

Låt u_{ij}^k beteckna samlingen av ord som driver maskinen från i till j och bara passerar tillstånd som är mindre eller lika med k.



Kapitel 7.2 – Kleenes algoritm

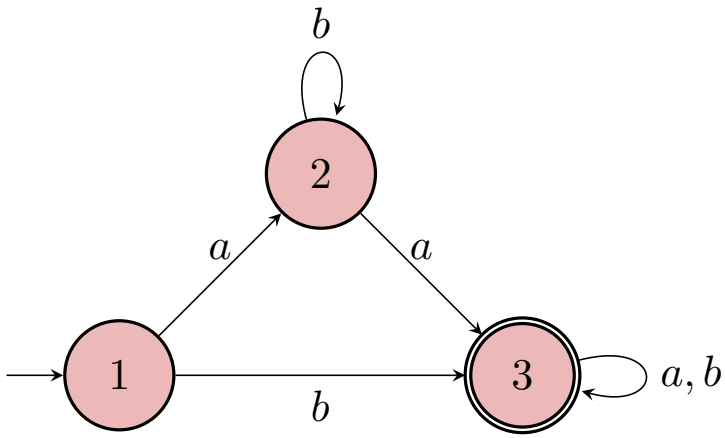
Låt u_{ij}^k beteckna samlingen av ord som driver maskinen från i till j och bara passerar tillstånd som är mindre eller lika med k.



Exempel: *abba* ligger i u_{13}^2, men inte i u_{13}^1. Ordet b ligger i u_{13}^0.

Kapitel 7.2 – Kleenes algoritm

Innan vi formulerar satsen, så ska vi titta på exemplet u_{13}^2 lite närmare. Vilka ord ingår i detta språk?



Kapitel 7.2 – Kleenes algoritm

Vi kan sammanfatta vår intuition i en sats.

Kapitel 7.2 – Kleenes algoritm

Vi kan sammanfatta vår intuition i en sats.

Sats: *Om i,j och k är tillstånd i en maskin, så är*

$$u_{ij}^{k+1} = u_{ij}^{k} \cup \left(u_{i(k+1)}^{k}\right)\left(u_{(k+1)(k+1)}^{k}\right)^{*}\left(u_{(k+1)j}^{k}\right).$$

Kapitel 7.2 – Kleenes algoritm

Vi kan sammanfatta vår intuition i en sats.

Sats: *Om i,j och k är tillstånd i en maskin, så är*

$$u_{ij}^{k+1} = u_{ij}^{k} \cup \left(u_{i(k+1)}^{k}\right)\left(u_{(k+1)(k+1)}^{k}\right)^{*}\left(u_{(k+1)j}^{k}\right).$$

Bevis: (whiteboard)

Kapitel 7.2 – Kleenes algoritm

Sats: *Om i,j och k är tillstånd i en maskin, så är u_{ij}^k reguljärt.*

Kapitel 7.2 – Kleenes algoritm

Sats: *Om i,j och k är tillstånd i en maskin, så är u_{ij}^k reguljärt.*

Bevis:

Kapitel 7.2 – Kleenes algoritm

Sats: *Alla språk som kan avgöras av en tillståndsmaskin är reguljära.*

Kapitel 7.2 – Kleenes algoritm

Sats: *Alla språk som kan avgöras av en tillståndsmaskin är reguljära.*

Bevis: Om en maskin med n tillstånd har starttillstånd i och accepterande tillstånd $n_1, \ldots, n_m$, så är maskinens språk

$$u^n_{in_1} \cup \cdots \cup u^n_{in_m},$$

vilket är reguljärt.

Kapitel 7.3 – Särskiljning

Vi har bara studerat reguljära språk. Hur vet man att ett språk **inte** är reguljärt?

Kapitel 7.3 – Särskiljning

Vi har bara studerat reguljära språk. Hur vet man att ett språk **inte** är reguljärt?

Definition: Ett språk L *särskiljer* en mängd ord W om det för varje par u och w i W finns ett ord z så att uz ligger i L men inte wz, eller vice versa.

Kapitel 7.3 – Särskiljning

Sats: *Om ett reguljärt språk L särskiljer en mängd ord W, så måste en maskin som avgör L drivas till olika tillstånd av orden i W.*

Bevis:

Kapitel 7.3 – Särskiljning

En tillståndsmaskin har per definition ändligt många tillstånd, vilket leder till följande sats.

Kapitel 7.3 – Särskiljning

En tillståndsmaskin har per definition ändligt många tillstånd, vilket leder till följande sats.

Sats: *Om ett språk kan särskilja en oändlig mängd med ord är det inte reguljärt.*

Kapitel 7.3 – Särskiljning

En tillståndsmaskin har per definition ändligt många tillstånd, vilket leder till följande sats.

Sats: *Om ett språk kan särskilja en oändlig mängd med ord är det inte reguljärt.*

Faktum är även omvändningen gäller: ett språk som bara kan särskilja ändligt många ord är alltid reguljärt.

Kapitel 7.3 – Särskiljning

Exempel: Språket $L = \{a^n b^n \mid n = 0, 1, 2 \ldots\}$ är inte reguljärt.

Kapitel 7.3 – Särskiljning

Exempel: Språket $L = \{a^n b^n \mid n = 0, 1, 2 \ldots\}$ är inte reguljärt.

Bevis: Språket L särskiljer mängden $W = \{a^n \mid n = 0, 1, \ldots\}$: om a^n och a^m är olika ord i W så gäller att $a^n b^n \in L$ och $a^m b^n \notin L$. Eftersom W är oändlig kan inte L vara reguljärt.

Kapitel 7.3 – Särskiljning

Sats: *Om L är ett språk är följande påståenden ekvivalenta.*

1. *L är reguljärt.*

Kapitel 7.3 – Särskiljning

Sats: *Om L är ett språk är följande påståenden ekvivalenta.*

1. *L är reguljärt.*
2. *L kan avgöras av en tillståndsmaskin.*

Kapitel 7.3 – Särskiljning

Sats: *Om L är ett språk är följande påståenden ekvivalenta.*

1. *L är reguljärt.*
2. *L kan avgöras av en tillståndsmaskin.*
3. *L kan endast särskilja ändligt många ord.*

Nästa tillfälle

Nästa gång är om tre veckor (16/4). Det är ett övningstillfälle och kommer att ske på distans.

Se hemsidan för mer information.