Disposition:

1. Introduktion
2. Turing Maskiner
3. Sprog
4. SA ≤ Acc
5. Acc ≤ Halts
6. Rice

1 Introduktion

Vi vil gerne kunne specificere hvilke problemer det kan lade sig gøre at løse og hvilke problemer vi måske kan løse og hvilke vi i hvert fald ikke kan løse, for at undgå at bruge kræfter på at løse uløselige problemer.

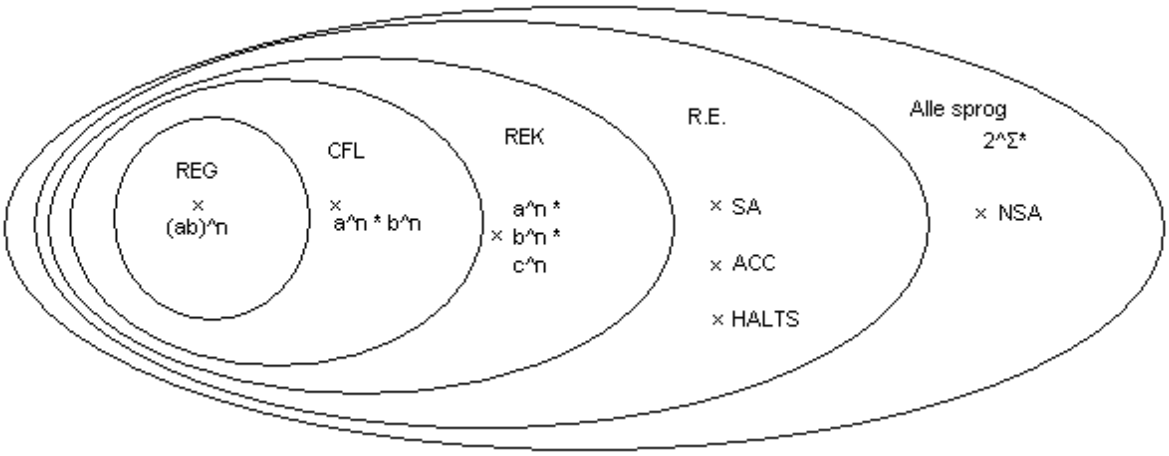
Det vil vise sig, at de fleste problemer er uløselige. Dog kan nogle problemer løses.

2 Turing Maskiner

TM kan udtrykke REG, CFL, Rekursive og Rekursiv Nummerable sprog.

5-tupel: $T = (Q, \Sigma, \Gamma, \mathrm{q}_0, \delta)$, $\delta(q, X) = (r, Y, D) | q, r \in Q$, $X, Y \in \Gamma$, $\mathrm{D} \in \{\mathrm{R}, \mathrm{L}, \mathrm{S}\}$

3 Sprog



Rekursive og rekursivt nummerable sprogklasser i forb. med TM.

Accepteret: svarer ja, hvis en streng ligger i sproget, ellers kan vi ikke sige hvad der sker (evigt loop). Dette svarer til de rekursivt nummerable sprog (semi-afgørbare). TM kan ikke afgøre, da strengene er nummereret tilfældigt, og vi dermed potentielt kan komme i et uendeligt loop.

Afgørlig: svarer ja til alle strenge i sproget, og nej til alle andre. Dette svarer til de rekursive sprog (afgørbare). Nummerer alle strenge alfabetisk, ved om strengen er indeholdt, hvis den når den, eller springer over, hvor den burde være.

3.1 Definitioner af SA

- **Definition 11.1**
  $NSA = \{w \in \{0,1\}^* \mid w = e(T) \text{ for some TM T, and } w \notin L(T)\}$
  $SA = \{w \in \{0,1\}^* \mid w = e(T) \text{ for some TM T, and } w \in L(T)\}$

SA er self-accepting: Der findes en TM der accepterer TM'en indkodet som input.

Bevis at SA er rekursivt enumerabelt men ikke rekursivt:

1. Sproget E indeholder alle strenge der er indkodede TM's.
2. Hvis sproget er self-accepting, så må det være R.E.
3. Da strengen w i SA er en indkodet TM og w er sproget for netop denne TM, så kan det bevises at SA er R.E.
4. Vi laver en TM der accepterer SA. Den vil læse noget input x og beslutte om det er i E, for at se om x = e(T) for en eller anden TM T.
5. Hvis x ikke er, vil vores TM rejecte.
6. Hvis x er i E vil den indkode e(x) = e(e(T)) og sætte den sammen med den indkodede TM T.
7. Derefter sender den e(T)e(e(T)) videre til en universel TM.
8. Vores TM accepterer hvis og kun hvis T accepterer e(T), og dermed er de accepterede strenge præcis dem i SA.

Hvorfor er den rekursivt enumerabel og ikke rekursiv? Fordi vi ikke ved om den terminerer på tilfældigt input, men kun at den ender i ha hvis den modtager sproget for TMen indkodet som input.

4 SA ≤ ACCEPTS

- **Theorem 11.5**

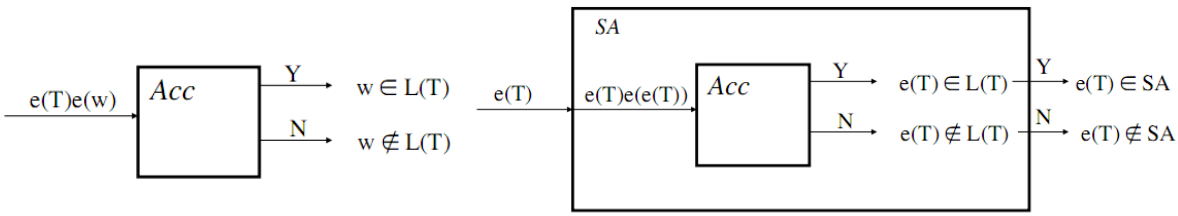
  $Acc = \{e(T)e(w) \mid w \in L(T)\}$ is not recursive

For at bevise at ACCEPTS ikke er rekursivt skal vi reducere det til et SA problem. Eftersom vi ved at SA ikke er rekursivt kan vi konkludere at det er ACCEPTS heller ikke.

ACCEPTS: Givet et input w og en TM T, accepterer T w?

Ideen er at indkode T som input w for ACC. Hvis den svarer ja så er indkodningen af T i sproget L(T) og derfor i SA. Hvis den svarer nej så er e(T) ikke i L(T) og derfor er den ikke i SA. Dette betyder at vi har reduceret ACCEPTS til et SA problem og eftersom SA er R.E. så er ACCEPTS ikke rekursivt.

- Assume you had TM accepting *Acc*
- Construct TM accepting *SA*



5 ACCEPTS ≤ HALTS

Givet en TM T og en streng w terminerer T (ha eller hr) på input w?

Vi vil vise at ACCEPTS kan reduceres til HALTS, og derved at HALTS ikke er rekursiv.

For at opnå ens opførsel mellem vores TM T, som sammen med strengen w er en instans af ACCEPTS, og vores TM T', som sammen med w' er en instans af HALTS, bliver T' nødt til at stoppe hvis og kun hvis T accepterer.

Lad w = w'. Så prøver vi at lave T' fra T.

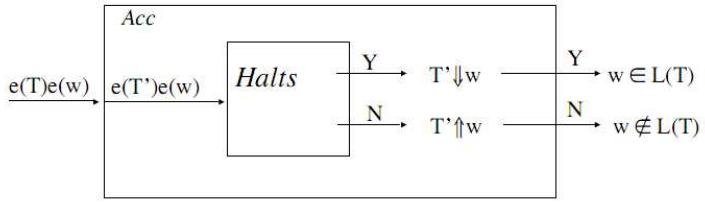
T stopper hvis den accepterer og den stopper ikke hvis den looper, så vi bliver nød til at modificere dens opførsel når den rejecter.

Vi ændrer alle transitioner der vil bringe T til hr, sådan at den i stedet bliver, hvor den er og looper der.

En sidste mulighed for T til at rejecte er, hvis den bevæger sig til venstre for felt 0 i TMen. Dette kan løses ved at flytte hele strengen en tak til højre og indsætte et tegn der ikke er i alfabetet på plads 0, og lave en transition, som lader T' loope, hvis den møder dette tegn.

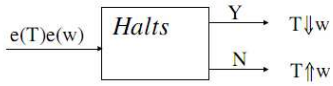
Nu er T blevet til T1.

- Construct TM accepting *Acc*



Reduction: $Acc \leq Halts$

- Assume you had TM accepting *Halts*



Where T' is T with all crashes and h_r replaced with divergence

Divergens betyder I denne henseende at der ikke kommer noget svar.

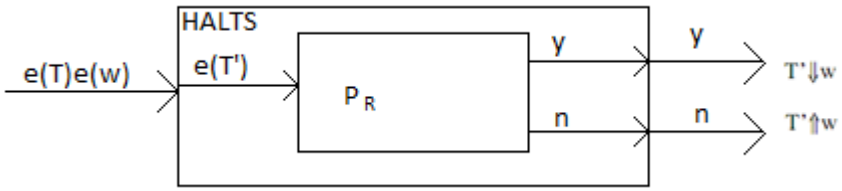
6. Rice's theorem

Rice's theorem siger, at beslutningsproblemet

P_R: Givet en TM t, har L(T) så egenskaben R?

er uløseligt. Hvor R vel og mærke er en ikke-triviel egenskab (dvs. den ikke kan svare ja for alle, eller nej for alle, men for nogen ja, og for nogen nej).

Vi laver igen en reduktion, denne gang: **Halts ≤ P_R**



Hvor T' er en TM, som først skal udføre inputtet til Halts, og derefter sørge for at egenskaben R gælder. Da T' afhænger af om inputtet nogensinde stopper, har vi vist, at dette problem er uløseligt, da vi netop tidligere har vist at Halting-problemet er uløseligt.

Mere håndgribeligt: Hvis vi på et bibliotek skal lave en bog, der indeholder alle de bøger, der ikke refererer til sig selv, så skal bogen selv også være med… Eller nej, eller jo…