

1. P, NP, NPC

- **Beslutningsproblemer, P, NP**
- **Reduktioner, NPH, NPC**
- **Cook's sætning**

I dette kursus beskæftiger vi os med **beslutningsproblemer**, hvor outputtet altid er ja eller nej. Input er probleminstanser kodet binært. Disse kan beskrives som sprog i $\{0,1\}^*$, hvor vi beslutter om input er med (svarer ja) eller ikke (svarer nej).

P er en kompleksitetsklasse, der indeholder de beslutningsproblemer, der kan afgøres af en TM i maks. $\rho(|x|)$ skridt på input x . Dvs. det kan afgøres i polynomisk tid om et input x er i et sprog.

Polynomial Church-Turing sætningen fortæller, at et beslutningsproblem kan løses i polynomisk tid af en passende sekventiel beregningsmodel \Leftrightarrow det kan løses i polynomisk tid af en TM. Dette betyder, at vi i praksis laver en effektiv algoritme, som løser problemet, og så henviser til **P C-T**.

NP er en anden kompleksitetsklasse, som er formelt defineret ved:

$$\forall x \in \{0,1\}^*: x \in L \Leftrightarrow [\exists y \in \{0,1\}^*: |y| \leq p(|x|) \wedge \langle x, y \rangle \in L'] \quad , \quad L' \in \mathbf{P} \quad , \quad p \text{ er poly.}$$

Dvs. vi vil kunne tjekke om en given løsning er en løsning i polynomiel tid – ydermere må denne løsning maks. have længden polynomiel af probleminstansen.

Dermed kunne vi afgøre om $x \in L$ for $L \in \mathbf{NP}$ ved at lave en exhaustive search, hvor der for alle mulige $2^{p(|x|)+1} - 1$ værdier af y tjekkes om $\langle x, y \rangle \in L'$.

Reduktion: $\exists r: L_1 \leq L_2: \forall x \in \{0,1\}^*: x \in L_1 \Leftrightarrow r(x) \in L_2$ hvor $r: \{0,1\}^* \rightarrow \{0,1\}^*$ er en polynomiel tids beregnelig model. Vi siger at L_2 er et mere generelt sprog end L_1 .

Transitivitet gælder for reduktioner: $L_1 \leq L_2 \wedge L_2 \leq L_3 \Rightarrow L_1 \leq L_3$.

NP-hårde sprog er de sprog, der har ret lille sandsynlighed for at ligge i P. Formelt kan vi sige

$$\forall L' \in \mathbf{NP}: L' \leq L \Rightarrow L \in \mathbf{NPH}$$

NP-hårde sprog ligger ikke nødvendigvis i NP, men de der gør er ret interessante, idet der er en del naturlige problemer, vi ønsker at løse, hvor vi ved, vi ikke kan finde en effektiv løsning (hvis $P \neq NP$). Problemer i både NP og NPH kaldes **NP-complete**.

Lemma 7: Hvis L_1 er NPH og $L_1 \leq L_2$ så er L_2 NPH.

Bevis: Transitivitet gælder, og idet alle problemer i NP reducerer til L_1 reducerer disse også til L_2 , og def. for NPH er opfyldt.

Lemma 7 giver os en anledning til at finde et problem, som er NPC, hvor vi vha. lemma 7 kan reducere andre mulige NPC-problemer til, således vi ikke behøver føre et langt, akavet bevis, hver gang vi vil vise, at et problem er NPC. Cook's sætning gør nøjagtigt dette. Først etablerer vi nogle lemmaer til senere brug:

Lemma 8: For enhver boolsk funktion $f: \{0,1\}^n \rightarrow \{0,1\}^m$ er der et kredsløb C således at $\forall x \in \{0,1\}^n: C(x) = f(x)$.

Lemma 9: Vi er givet en TM M , som kører i tid højst $p(n) > n$ på input af længde n , hvor p er et pol.

Givet ethvert fixed input længde n er der et kredsløb C_n af størrelse (antal gates) højst $O(p(n)^2)$ således at $\forall x \in \{0,1\}^n: C_n(x) = 1 \Leftrightarrow M \text{ acc. } x$.

Mapning $1^n \rightarrow C_n$ er polynomisk tid beregnelig.

Bevis: Vi kigger på positioner $i: -p(x)$ til $p(x)$ og tider $t \in \{0, \dots, p(n)\}$

Cell-state-vektor $c_{t,i}$ indeholder information om symbolet på plads i til tid t , om hovedet peger på plads i og hvis der peges på i noterer vi tilstanden for TM til tiden t .

Vi kan bestemme $c_{t,i} = h(c_{t-1,i-1}, c_{t-1,i}, c_{t-1,i+1})$ (ud fra tidligere tid).

Lemma 8 giver os et kredsløb D der kan beregne h . Kredsløb E , hvor $E(b)$ er en cell-state-vektor, hvor cellen indeholder 0 eller 1, og hovedet ikke peger på. Kredsløb F svarer $F(y) = 1$, hvis y er en cell-state-vektor, hvor hovedet er på, og TM er i en accept-tilstand.

Cook's sætning giver os et problem SAT, som er NP-hårdt (og ligger i NP). Dermed ligger det i NPC. Vi viser at CircuitSAT er NP-hårdt og reducerer til SAT, hvormed Cook's sætning er vist. CircuitSAT generaliserer SAT, idet CNF'er er formularer, som er kredsløb.

CircuitSAT: Givet et boolsk kredsløb C , er der så en vektor x , således at $C(x) = 1$?

Theorem 11: CIRCUIT SAT \in NPC

Bevis: CSAT er tydeligvis i NP, så vi skal vise, at det er NP-hårdt. Givet et tilfældigt sprog $L \in NP$, skal vi vise at $\forall x \in L \Leftrightarrow r(x) \in CSAT$.

$$\forall x \in \{0,1\}^*: x \in L \Leftrightarrow [\exists y \in \{0,1\}^*: |y| \leq p(|x|) \wedge \langle x, y \rangle \in L'] \quad , \quad L' \in \mathbf{P} \quad , \quad p \text{ er poly.}$$

Instanser af L skal mappes til instanser af CSAT, dvs. beskrivelser af kredsløb. Givet input x , så er $r(x)$ en beskrivelse af kredsløbet $C \equiv D_0 \vee D_1 \vee \dots \vee D_{p(|x|)}$.

Vi lader M være en TM, der afgør L' i polynomisk tid. Lemma 9 giver os et kredsløb C_n , som for alle z, y hvor $|\langle z, y \rangle| = n$ giver $C_n(\langle z, y \rangle) = 1 \Leftrightarrow M \text{ acc. } \langle z, y \rangle$. Vi lader $n = 2(|x| + |y| + 1)$.

På C_n hardwirer vi input x ved at udskifte ulige inputgates med x_i , lige med 0, og de to sidste med 1:

$$x_1 0 x_2 0 \dots 0 x_n 0 1 1$$

Dette kalder vi D_i , som kan afgøre om et input y af længde i er en løsning til x : Hver D_i tager i boolske input og evaluerer til 1 på input $y \in \{0,1\}^i \Leftrightarrow \langle x, y \rangle \in L'$.

Dvs. reduktionen skal konstruere $p(|x|)$ underkredsløb D_i , som kombineres med ORs og outputter C . Vha. Polynomial Church-Turing sætning kan vi se reduktionen er polynomisk tids beregnelig.

Nu har vi vist, at CSAT er NP-hårdt, men Cook's sætning siger, at SAT er NP-hårdt.

Theorem 12: CSAT \leq SAT

Givet et 1-output kredsløb C , definerer vi $f = r(C)$, således at f har en tilfredsstillende sandhedstilordning $\Leftrightarrow C$ har. r oversætter i polynomiel tid: $x \in CSAT \Leftrightarrow r(x) \in SAT$. Vi oversætter gates til klausuler:

AND: $(g \Leftrightarrow h_1 \wedge h_2)$

$$\begin{aligned} &\equiv (g \Rightarrow (h_1 \wedge h_2)) \wedge ((h_1 \wedge h_2) \Rightarrow g) \\ &\equiv (\neg g \vee (h_1 \wedge h_2)) \wedge (\neg(h_1 \wedge h_2) \vee g) \\ &\equiv (\neg g \vee h_1) \wedge (\neg g \vee h_2) \wedge (\neg h_1 \vee \neg h_2 \vee g) \end{aligned}$$

OR: $(g \Leftrightarrow h_1 \vee h_2) \equiv (\neg g \vee h_1 \vee h_2) \wedge (g \vee \neg h_1) \wedge (g \vee \neg h_2)$

NOT: $(g \Leftrightarrow \neg h) \equiv (\neg g \vee \neg h) \wedge (g \vee h)$ - klausulerne opfyldes kun, hvis g og h er forskellige.

COPY: $(g \Leftrightarrow h) \equiv (\neg g \vee h) \wedge (g \vee \neg h)$ - klausulerne opfyldes kun, hvis g og h er ens.

CONST-0: $0 \rightarrow (\neg g)$ CONST-1: $1 \rightarrow (g)$ OUTPUT: $g \rightarrow (g)$ er outputtet af C .

Ovenstående klausuler ANDes sammen til f . Output-klausulen sikrer, at SAT-formlen kun går op, når vores CSAT går op.