DOUBLE-SAT We toonden op 17 april reeds aan dat DOUBLE-SAT in NP zit. Om de NP-compleetheid aan te tonen, reduceren we SAT naar DOUBLE-SAT in polynomiale tijd.

Beschouw daartoe volgende functie f:

1. Input: Een willekeurige input ϕ voor SAT.
2. Neem een variabele die niet voorkomt in ϕ, noem hem u.
3. Output: De formule $\phi \wedge (u \vee \neg u)$, dus $f(\phi) := (\phi \wedge (u \vee \neg u))$.

Deze f is duidelijk efficiënt berekenbaar. We moeten aantonen:

$$\phi \in \text{SAT} \quad \Leftrightarrow \quad f(\phi) \in \text{DOUBLE-SAT}.$$

$\boxed{\Rightarrow:}$ Gegeven: $\phi \in$ SAT, d.w.z., er bestaat een waarheidstoekenning op de variabelen in ϕ waaronder ϕ waar wordt. Noem deze toekenning α. Gevraagd: $f(\phi) \in$ DOUBLE-SAT, d.w.z, er bestaan twee verschillende waarheidstoekenningen waaronder $\phi \wedge (u \vee \neg u)$ waar wordt. Neem als eerste waarheidstoekenning α die we uitbreiden naar u door u op 1 te zetten. Neem als tweede toekenning α die we uitbreiden naar u door u op 0 te zetten.

$\boxed{\Leftarrow:}$ Gegeven: $f(\phi) \in$ DOUBLE-SAT, d.w.z., er bestaan twee verschillende waarheidstoekenningen waaronder $\phi \wedge (u \vee \neg u)$ waar wordt. Noem deze α_1 en α_2. Gevraagd: $\phi \in$ SAT, d.w.z., er bestaat een waarheidstoekenning waaronder ϕ waar wordt. Neem hiervoor b.v. α_1 (je kan ook α_2 nemen) waar je de waarde van u gewoon negeert (u komt toch niet voor in ϕ). Als $\phi \wedge (u \vee \neg u)$ waar is onder α_1, is zeker ϕ waar onder α_1 (want het is een AND).

HALF-CLIQUE We toonden op 17 april reeds aan dat HALF-CLIQUE in NP zit. Om de NP-compleetheid aan te tonen, reduceren we CLIQUE naar HALF-CLIQUE in polynomiale tijd.

Beschouw daartoe volgende functie f:

1. Input: Een willekeurige input $\langle G, k\rangle$ voor CLIQUE.
2. Zij n het aantal knopen in G. We construeren een graaf H als volgt:
   - Als $k = n/2$ dan is H gewoon G.
   - Als $k < n/2$ dan is H bekomen uit G door een clique van $n - 2k$ nieuwe knopen toe te voegen aan G, en alle knopen uit deze clique ook nog eens te verbinden met alle knopen uit G.
   - Als $k > n/2$ dan is H bekomen uit G door $2k - n$ nieuwe knopen toe te voegen, zonder enige verbindingen.
3. Output: H, dus $f(G, k) := H$.

Deze f is duidelijk efficiënt berekenbaar. We moeten aantonen:

$$\langle G, k\rangle \in \text{CLIQUE} \quad \Leftrightarrow \quad f(G, k) \in \text{HALF-CLIQUE}.$$

$\boxed{\Rightarrow:}$ Stel $\langle G, k\rangle \in$ CLIQUE. Dus G bezit een clique van k knopen. Noem deze clique C. We moeten aantonen dat H een clique bezit van de helft van z'n knopen. We beschouwen de verschillende gevallen.

- Als $k = n/2$ dan is H gelijk aan G en heeft H dus een $n/2$-clique zoals gewenst.
- Als $k < n/2$ dan heeft H $n + (n - 2k) = 2(n - k)$ knopen. De helft hiervan is $n - k$. We zoeken dus een clique van $n - k$ knopen. Deze vinden we ook: we hebben reeds C van k knopen, en voegen daarbij de extra clique van $n - 2k$ toegevoegde knopen uit H. Totaal $k + (n - 2k) = n - k$ knopen zoals gewenst.
- Als $k > n/2$ dan heeft H $n + (2k - n) = 2k$ knopen. De helft hiervan is k. We zoeken dus een clique van k knopen. Deze hebben we, namelijk C.

$\boxed{\Leftarrow:}$ Stel $H \in$ HALF-CLIQUE. Dus H bezit een clique bestaande uit minstens de helft van z'n knopen. We moeten aantonen dat G dan een clique van k knopen bezit. We beschouwen weer de verschillende gevallen.

- Als $k = n/2$ dan is G gelijk aan H. H, en dus G, heeft een clique bestaande uit minstens $n/2$ knopen, en dus heeft G zeker een clique van k knopen want $k = n/2$.
- Als $k < n/2$ dan heeft H $2(n - k)$ knopen (zie hierboven). H heeft dus een clique van minstens $n - k$ knopen. Hoogstens $n - 2k$ daarvan komen uit het stuk dat we toevoegden, omdat dit stuk slechts zo groot was. Dus minstens k moeten komen uit de originele G. Dus heeft G minstens een k-clique.
- Als $k > n/2$ dan heeft H $2k$ knopen (zie hierboven). H heeft dus een clique van minstens k knopen. Deze kunnen echter niet komen uit het stuk dat we toevoegden, dit waren immers knopen zonder verbindingen. De clique moet dus volledig uit G komen.

Oefening 7.25 We tonen eerst aan dat U in NP zit. Een niet-deterministisch polynomiaal algoritme voor U is het volgende: "Op input $\langle M, x, 1^t\rangle$, run M op x (niet-deterministich) voor t stappen. Aanvaard indien de gekozen run aanvaardt."

Merk op dat het hier belangrijk is dat t gegeven is als t eentjes, immers, als t b.v. in decimale notatie zou gegeven zijn, dan zou de lengte daarvan slechts $\log t$ zijn, en we doen t stappen, dit is gelijk aan $2^{\log t}$ wat dus exponentiëel is.

Nu tonen we aan dat elk probleem in NP kan gereduceerd worden naar U in polynomiale tijd. Neem een willekeurig probleem A in NP. Dan is er een polynomiaal niet-deterministisch algoritme voor A. Schrijf dit algoritme als een Turing machine M. Deze machine werkt op een input x hoogstens n^k stappen, waar n de lengte is van x.

Beschouw nu volgende functie $f(x) := \langle M, x, 1^{n^k} \rangle$, waarbij n de lengte van x is." Deze functie is duidelijk efficiënt berekenbaar. Immers, M is een constante string, namelijk de source code van M; x is gewoon de input kopiëren; en n^k eentjes schrijven is duidelijk polynomiaal. We tonen nu aan dat deze f een correcte reductie is.

- Als $x \in A$, dan wordt x dus aanvaard door M, en aangezien M hoogstens n^k stappen maakt, is dus $f(x) \in U$.

- Als $f(x) \in U$, dan wordt x dus aanvaard door M in hoogstens n^k stappen. Aangezien M nooit meer stappen doet dan dat, wordt x dus aanvaard door M tout court, en is dus $x \in A$.