Het bewijs van de stelling van Cook–Levin zoals gegeven in het boek van Sipser gebruikt niet-deterministische turing machines. Het is inderdaad mogelijk de klasse NP op een alternatieve wijze te definiëren met behulp van polynomiale niet-deterministische turing machines, i.p.v. met behulp van verifiers. Omdat we in onze cursus niet met niet-deterministische turing machines willen werken, geven we hier een aangepast bewijs gebaseerd op de ons bekende definitie van NP met behulp van verifiers.

Stelling. *SAT is NP-compleet.*

Idee van het bewijs: Aantonen dat SAT in NP zit is gemakkelijk. Het moeilijke aan het bewijs is aan te tonen dat elke taal in NP polynomiaal kan gereduceerd worden naar SAT.

Om dit te doen construeren we voor elke taal A in NP een polynomiale reductie naar SAT. Zij V een polynomiale verifier voor A. De reductie voor A neemt een string w als input, en produceert een booleaanse formule ϕ als output. Deze formule ϕ zal variabelen bevatten die tesamen een certificaat c voorstellen. De formule zal een zodanige structuur hebben dat ze satisfiable is, als en slechts als er een certificaat c bestaat zodat V de input w, c aanvaardt.

De formule ϕ zal dus de werking van de verifier moeten simuleren. Op zich is het niet verwonderlijk dat dit mogelijk is; tenslotte bestaat een echte computer ook niet uit veel anders dan massa's logische schakelingen zoals AND, OR, en NOT. De gedetailleerde beschrijving van ϕ vergt echter wel serieus wat werk.

Bewijs. Eerst tonen we aan dat SAT in NP zit. Dit is gemakkelijk; we kunnen een verifier construeren die als certificaat voor lidmaatschap van een formule ϕ in SAT, een toekenning verwacht van de variabelen in ϕ waaronder de formule inderdaad tot 1 evalueert. De verifier hoeft enkel de formule uit te rekenen gegeven de toekenning, dit kan efficiënt gebeuren en is dus polynomiaal.

Nu nemen we een willekeurige taal A in NP en tonen aan dat A polynomiaal kan gereduceerd worden naar SAT. Zij V een polynomiale verifier voor A. We kunnen hem implementeren als een single-tape turing machine die op een input w, c loopt in tijd n^k voor een of andere vaste k, waarbij n de lengte is van w. We voeren nu het volgende begrip in:

De ***berekeningstabel*** voor V op een input w, c is een $n^k \times n^k$ tabel, waar elke rij een configuratie[1] voorstelt van V op input w, c. Zie de illustratie in Figure 7.8 van het boek, *waarbij je in de bovenste rij echter ook nog het certificaat c moet toevoegen aan de input:*

#	q_0	w_1	w_2	$\cdots$	w_n	$\sqcup$	c_1	$\cdots$	c_m	$\sqcup$	$\cdots$	$\sqcup$	#

Voor het gemak onderstellen we dat elke configuratie begint en eindigt met de letter # zodat de eerste en laatste kolommen van de tabel volledig gevuld zijn met deze letter. De eerste rij van de tabel is de startconfiguratie van V op input w, c. Elke volgende rij is de configuratie bekomen uit de vorige door 1 stap van V uit te voeren. We noemen de tabel ***aanvaardend*** als minstens een van van z'n rijen een aanvaardende configuratie is. Dus, of we nu zeggen *"V aanvaardt input w, c"* of we zeggen *"de berekeningstabel van V op input w, c is aanvaardend"* dan zeggen we twee keer precies hetzelfde.

We kunnen nu beginnen aan onze polynomiale reductie f van A naar SAT. Op input w zal $f(w)$ een booleaanse formule ϕ zijn. We beginnen met te beschrijven wat de variabelen in ϕ zijn. Noem de verzameling toestanden en het tape-alfabet van V respectievelijk Q en Γ. Stel $C := Q \cup \Gamma \cup \{\#\}$. Zij n de lengte van w. Voor elke i en j tussen 1 en n^k en voor elke s in C hebben we een variabele $x_{i,j,s}$. De intuïtieve betekenis van variabele $x_{i,j,s}$ is dat in de berekeningstabel, de cel in rij i en kolom j het symbool s bevat.

Nu ontwerpen we ϕ zodanig dat de toekenningen van de variabelen waaronder ϕ waar wordt, precies overeenkomen met certificaten c die samen met w aanvaard worden door V. De formule ϕ is de AND van vier delen:

$$\phi = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}.$$

We beschrijven nu elk deel.

Formule ϕ_{cell} drukt uit dat elke cel van de tabel precies 1 symbool bevat:

$$\phi_{\text{cell}} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

Voor elke cel zegt deze formule dat minstens 1 van de variabelen waar moet zijn (elke cel moet ingevuld zijn), en zegt ze ook dat geen 2 variabelen tegelijk mogen waar zijn (er kunnen geen 2 verschillende symbolen in 1 cel staan).

[1] Als je niet meer weet wat een configuratie is van een turing machine, is het nu de moment dit op te frissen, zie hoofdstuk 3 uit het boek.

Elke toekenning die ϕ waar maakt, maakt dus zeker ϕ_{cell} waar, en specificeert dus een willekeurige invulling van een $n^k \times n^k$ tabel. De overige formules ϕ_{start}, ϕ_{move} en ϕ_{accept} moeten er nu voor zorgen dat deze invulling wel degelijk de berekeningstabel van V op een input w, c voorstelt, waar w de gegeven input van onze reductie is, en c een willekeurig certificaat is.

Formule ϕ_{start} zegt dat de eerste rij van de tabel de startconfiguratie is van V op een input w, c, waar w onze gegeven input is, maar c volledig willekeurig blijft:

$$\begin{aligned}\phi_{\text{start}} = {} & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \cdots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

We onderstellen dat w en c in de input gescheiden van elkaar zijn door een blanko, vandaar dat $x_{1,n+3,\sqcup}$ waar moet zijn. De formule zegt niets over de cellen in de eerste rij in posities tussen $n+3$ en n^k, daar kan dus een willekeurig certificaat komen te staan.

Formule ϕ_{accept} zegt dat de accept-toestand van V, q_{accept}, in tenminste een rij voorkomt, en dus dat de berekeningstabel aanvaardend is:

$$\phi_{\text{accept}} = \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{\text{accept}}}$$

Tenslotte hebben we formule ϕ_{move}. Die moet garanderen dat elke rij (behalve de eerste) bekomen is uit de vorige door 1 stap van V uit te voeren. Hiervoor gaat het elk 2×3 venstertje in de tabel controleren. We zeggen dat zo'n venster ***legaal*** is als het overeenstemt met de transitiefunctie δ van V.

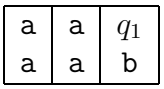
Bijvoorbeeld, stel dat `a`, `b`, en `c` tape-letters zijn en dat q_1 en q_2 toestanden zijn. Stel dat $\delta(q_1, \texttt{a}) = (q_1, \texttt{b}, \text{R})$ en $\delta(q_1, \texttt{b}) = (q_2, \texttt{c}, \text{L})$. We beschouwen dan een paar voorbeelden van legale vensters:

`a`	q_1	`b`
q_2	`a`	`c`

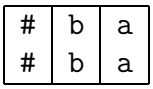
  Dit venster is legaal omdat er correct gebeurt wat $\delta(q_1, \texttt{b})$ voorschrijft.

`a`	q_1	`a`
`a`	`b`	q_1

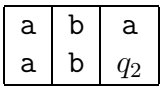
  Nu gebeurt correct wat $\delta(q_1, \texttt{a})$ voorschrijft.

- 
a	a	q_1
a	a	b

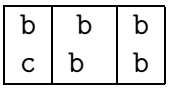
We zien hier in de bovenste rij van het venster niet wat de letter is die de machine ziet; we zien enkel dat ze in toestand q_1 is. De desbetreffende letter zou een a kunnen zijn, en in dat geval zou de machine ze volgens $\delta(q_1, \mathtt{a})$ in een b herschrijven en naar rechts gaan. Dit is precies wat de onderste rij van het venster beschrijft. Het venster is dus legaal.

- 
#	b	a
#	b	a

De twee rijen van het venster zijn gelijk. Dit kan gebeuren als de kop van de machine in de bovenste configuratie helemaal niet in de buurt is, veel verder naar rechts bijvoorbeeld, in dat geval zou er in het venster dus geen verandering te bespeuren zijn. Het venster is dus legaal.

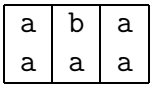
- 
a	b	a
a	b	q_2

Dit venster is legaal omdat in de bovenste configuratie de kop van de machine vlak rechts van de bovenste rij van het venster zou kunnen staan, in toestand q_1, en een b lezend. In dat geval zou hij inderdaad naar links gaan in toestand q_2 en zou dus q_2 verschijnen rechts in de onderste rij van het venster.

- 
b	b	b
c	b	b

Tenslotte is ook dit venster legaal: het zou kunnen zijn dat in de bovenste configuratie de kop van de machine staat op de meest linkse b van de bovenste rij van het venster, in toestand q_1. De q_1 is dan net uit het zicht, links van de eerste rij van het venster. In dat geval zou de machine de b herschrijven in een c, en nog een stap meer naar links gaan, zodat we enkel de c zien verschijnen in de onderste rij van het venster.

Hier zijn een paar voorbeelden van illegale vensters voor hetzelfde voorbeeld:

- 
a	b	a
a	a	a

Een letter kan enkel herschreven worden als de kop erop staat. Hier wordt de b herschreven, dus zou er links van b in de bovenste rij van het venster een toestand moeten staan, er staat echter een letter a. Het venster is dus illegaal.

a	q_1	b
q_1	a	a

  Hier gebeurt niet wat voorgeschreven is door $\delta(q_1, \texttt{b})$. De b moet herschreven worden in een c, niet een a.

b	q_1	b
q_2	b	q_2

  Dit venster is illegaal omdat er twee keer een toestand vermeld staat in de onderste rij. Dit kan niet, onze machine heeft maar 1 kop!

We noteren nu:

> *Als de bovenste rij van de tabel de startconfiguratie is, en elk* 2×3 *venster in de tabel is legaal met betrekking tot de transitiefunctie van* V*, dan is elke volgende rij van de tabel precies de configuratie bekomen uit de vorige door 1 stap uit te voeren van* V*.*

Dit is Claim 7.31 in het bewijs uit het boek. Vanaf nu verloopt het bewijs letterlijk verder zoals in het boek,[2] en verwijzen we dus gewoon verder naar het boek voor de rest van het bewijs.

[2]Met 1 uitzondering: in de zesdelaatste lijn van pagina 258 moet je 'N' vervangen door 'V'.