$\boxed{\text{NL} \subseteq \text{P}}$

Dit tekstje vervangt Hoofdstuk 8.5 uit het boek.

Stelling. *Voor elke taal* $A \in \text{NL}$ *geldt dat* $A \leq_{\text{P}} PATH$.

Bewijs. Neem een taal $A \in \text{NL}$. Dit wil zeggen dat er een niet-deterministische Turing machine M is, met 1 read-only inputtape en 1 werktape, die een input aanvaard als en slechts als die input in A zit, en die op elke input hoogstens $\log n$ space gebruikt op de werktape (n is de lengte van de input).

We moet inzien dat er een efficiënte reductie is van A naar $PATH$. We moeten dus een willekeurige input w voor A op efficiënte wijze kunnen omvormen naar een tripel $\langle G, s, t\rangle$, waar G een graaf is, en s en t twee van z'n knopen, zodat

$$w \in A \quad \Leftrightarrow \quad \langle G, s, t\rangle \in PATH.$$

Als knopen van G nemen we alle 4-tupels $\langle q, s, i, j\rangle$ zodat:

- q is een toestand van M;
- s is een willekeurige string van lengte hoogstens $\log n$, met n de lengte van w;
- i is een getal tussen 0 en n; en
- j is een getal tussen 0 en $\log n$.

Elke knoop van G stelt een mogelijke configuratie voor van M werkend op input w, waarbij q de toestand is, s de inhoud van de werktape, i de positie van de kop op de inputtape, en j de positie van de kop op de werktape.

We trekken een pijl van knoop K_1 naar knoop K_2 als M van configuratie K_1 kan overgaan in configuratie K_2 in 1 stap.

Als s nemen we de knoop die de startconfiguratie voorstelt; dit is de knoop $\langle q_0, \sqcup \ldots \sqcup, 0, 0\rangle$, waar q_0 de begintoestand is van M, en $\sqcup \ldots \sqcup$ de string is bestaande uit precies $\log n$ blanco's. Als t nemen we de knoop $\langle q_{\text{yes}}, \sqcup \ldots \sqcup, 0, 0\rangle$, waar q_{yes} de eindtoestand is van M. We onderstellen hierbij voor het gemak dat M maar 1 eindtoestand heeft, en dat áls M haar input aanvaardt, dat ze dan eerst de werktape volledig wist (= volschrijft met blanco's) en de twee koppen terug helemaal links zet. Als M niet deze eigenschappen heeft, kunnen we haar altijd gemakkelijk herprogrammeren zodat ze ze krijgt.

We hebben nu duidelijk dat er een pad is in G van s naar t als en slechts als er een berekening is van M die w aanvaardt, d.w.z., als en slechts als $w \in A$.

Is G efficiënt te construeren? Laat ons eerst naar de knopen kijken. Stel dat het alfabet bestaat uit ℓ letters.

- Het aantal mogelijke strings s van $\log n$ letters is

$$\ell^{\log n} = (2^{\log \ell})^{\log n} = (2^{\log n})^{\log \ell} = n^{\log \ell}.$$

- Het aantal mogelijkheden voor q is het aantal toestanden van M, dit is een constante.

- Het aantal mogelijkheden voor i is n, en dat voor j is $\log n$.

We besluiten hieruit dat het aantal knopen van G gelijk is aan

$$n^{\log \ell} \times \text{constante} \times n \times \log n,$$

wat duidelijk polynomiaal is in n (ℓ is ook een constante). We kunnen ze eenvoudig allemaal genereren, en omdat er dus maar polynomiaal veel zijn, kunnen we ze dus genereren in polynomiale tijd.

Om de pijlen van G te genereren doen we het volgende. Voor elke knoop K_1 kijken we naar de toestand, de i-de letter van w, en de j-de letter van s. Op deze informatie passen we de transitiefunctie van M toe, die een eindig aantal mogelijke acties geeft (letter op werktape veranderen, twee koppen bewegen naar links of rechts, en van toestand veranderen). Elk van deze mogelijke acties geeft een mogelijke volgende configuratie. We lopen nu opnieuw (geneste loop) door alle knopen, en voor elke knoop K_2 die tot de lijst van mogelijkheden behoort, genereren we een pijl (K_1, K_2). Omdat het aantal knopen polynomiaal is, en de transitiefunctie van M vastligt, werkt dit in polynomiale tijd. ■

Gevolg. *Elk probleem in NL zit in P.*