Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

Satz D: (Sprachkomplement)

DTIME(f(n)) = co-DTIME(f(n)) DSPACE(f(n)) = co-DSPACE(f(n))

NSPACE(f(n)) = co-NSPACE(f(n))

Dabei ist für Sprachklasse X die Sprachklasse co-X definiert als

co-X = { Sprache L | $\overline{L} \in$ X }

Lemma D: $L \in$ NSPACE(f(n)) $\Rightarrow \overline{L} \in$ NSPACE(f(n))

Beweis auch dieses Lemmas reduziert sich auf "Berechnen" von Erreichbarkeit in einem sehr großen, implizit gegebenen Graphen. Es muss die Frage beantwortet werden

$\exists$ gerichteter Pfad von init(w) zu einer Endkonfiguration im Konfigurationsgraphen $KG_M(w,N)$

n=|w| N=f(n)

Abstraktes Problem:

Gerichteter Graph G = (V,E) ist implizit gegeben, d.h. man kann
- (i) die Knoten in V aufzählen
- (ii) für zwei Knoten u,v testen, ob u=v
- (iii) für zwei Knoten u,v testen, ob $\langle u,v \rangle \in E$

und zwar jeweils mit Speicherverbrauch O(log |V|) (Knotendarstellungsgröße)

Berechne

erreichbar(s , v, λ) ... gibt es in G einen gerichteten Pfad von s nach v der Länge höchstens λ ?

Lemma D: (Immerman, Szelepcsényi)

$L \in$ NSPACE(f(n)) $\Rightarrow \overline{L} \in$ NSPACE(f(n))

Für Beweis von Lemma D, nicht-deterministische Lösung mit O(log|V|) Platzverbrauch

Achtung: Das Schwierige ist, nicht-deterministisch zu zeigen, dass erreichbar(s , v, λ) nicht gilt.

$R_s(\lambda) = \{ x \in V \mid$ erreichbar(s , x , λ) $\}$

$N_s(\lambda) = |R_s(\lambda)|$

Beh 0: Gegeben $x \in R_s(\lambda)$, kann man nicht-det. mit O(log|V|) Platz zeigen, dass tatsächlich $x \in R_s(\lambda)$.

Bew: rate und verifiziere Pfad von s nach x

$R_s(\lambda) = \{ x \in V \mid \text{erreichbar}(s , x , \lambda) \}$

$N_s(\lambda) = |R_s(\lambda)|$

Beh 0: Gegeben $x \in R_s(\lambda)$, kann man nicht-det. mit $O(\log|V|)$ Platz zeigen, dass tatsächlich $x \in R_s(\lambda)$.

Bew: rate und verifiziere Pfad von s nach x

Beh 1: Wenn $N_s(\lambda)$ bekannt ist, kann man, gegeben $x \notin R_s(\lambda)$, nicht-det. mit $O(\log|V|)$ Platz zeigen, dass tatsächlich $x \notin R_s(\lambda)$.

Bew: rate $N_s(\lambda)$ viele verschiedene $u \in V$, und verifiziere, dass $u \neq x$ und $u \in R_s(\lambda)$ (Beh 0!).

Beh 2: Wenn $N_s(\lambda -1)$ bekannt ist, kann man nicht-det. mit $O(\log|V|)$ Platz $N_s(\lambda)$ berechnen.

Bew:
```
Z=0
for each y∈V do
    rate ob y∈R_s(λ)
    wenn ja, dann verifiziere dies (Beh. 0) und Z=Z+1
    wenn nein, dann verifiziere dies folgendermaßen:
            verifiziere y∉R_s(λ-1) (Beh. 1)
            for each x∈V do
                verifiziere x∉R_s(λ-1) (Beh. 1)
                    oder [x,y⟩∉E
```


Für Beweis von Lemma D, nicht-deterministische Lösung mit $O(\log|V|)$ Platzverbrauch

```
erreichbar( s , v, λ ) = rate richtig ob ja oder nein
                         wenn ja, dann verifiziere dies (Beh. 0)
                         wenn nein:
                             N_s(0) = 0
                             for i from 1 to λ do
                                 berechne N_s(i) aus N_s(i-1) (Beh. 2)
                             mit N_s(λ) verifiziere, dass v∉R_s(λ)
                                                          (Beh. 1)
```


Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

Satz E: (Hierarchie) $f \in O(g)$

$\text{DTIME}(f(n)) \subseteq \text{DTIME}(g(n))$ $\quad \text{DSPACE}(f(n)) \subseteq \text{DSPACE}(g(n))$

$\text{NTIME}(f(n)) \subseteq \text{NTIME}(g(n))$ $\quad \text{NSPACE}(f(n)) \subseteq \text{NSPACE}(g(n))$

Satz F: (strikte Hierarchie)

$f(n) \in o(g(n)) \Rightarrow \text{DSPACE}(f(n)) \subsetneq \text{DSPACE}(g(n))$

$\text{NSPACE}(f(n)) \subsetneq \text{NSPACE}(g(n))$

$\text{NTIME}(f(n)) \subsetneq \text{NTIME}(g(n))$

$f(n) \in o(g(n)/\log g(n)) \Rightarrow \text{DTIME}(f(n)) \subsetneq \text{DTIME}(g(n))$

Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

Diese "constructibility"-Bedingung ist wesentlich! Ohne sie können recht eigenartige Phänomene auftreten:

Satz Z: (Borodin's Lückensatz)

Sei $g()$ eine überall berechenbare rekursive Funktion.
Dann existiert eine überall berechenbare Funktion $T()$ mit

$\text{DTIME}(S(n)) = \text{DTIME}(g(S(n)))$

z.B. für $g(n)=2^n$ beweist dieser Satz die Existenz einer Funktion $S()$ mit

$\text{DTIME}(S(n)) = \text{DTIME}(2^{(S(n))})$,

also exponentiell mehr Zeit "bringt nichts".

Analoges gilt für NTIME, NSPACE, DSPACE.

Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

Diese "constructibility"-Bedingung ist wesentlich! Ohne sie können recht eigenartige Phänomene auftreten:

Satz Y: (Blum's Beschleunigungssatz)

Sei $g()$ eine überall berechenbare rekursive Funktion.
Dann existiert eine TM-Sprache L sodass

$\forall$ TM M mit $L(M)=L$ $\quad \exists$ TM M' mit $L(M')=L$ sodass $\quad g(T_M(n)) \leq_{fa} T_M(n)$

$T_M(n)$ ist dabei die worst-case Laufzeit von Turingmaschine M

z.B. für $g(n)=2^n$ beweist dieser Satz die Existenz einer Sprache L, für die jede sie akzeptierende Turingmaschine durch eine exponentiell schnellere ersetzt werden kann.

Analoges gilt für NTIME, NSPACE, DSPACE.