Lecture 1

WHILE-Syntax: simple statement: -$x_i := x_j + x_k$, -$x_i := x_j - x_k$, -$x_i := c$, $i, j, k, c \in \mathbb{N}$
WHILE program: 1: simple statement or, 2: $P_1; P_2$ or, 3: **while** $x_i \neq 0$ **do** P_1 **od**, for WHILE programs P_1, P_2 and $i \in \mathbb{N}$
$\mathcal{W}_0 = \{\text{simple statements}\}$
$\mathcal{W}_n = \mathcal{W}_{n-1} \cup \{P | \exists P_1 \in \mathcal{W}_{n-1}, i \in \mathbb{N} : P = \text{while } x_i \neq 0 \text{ do } P_1 \text{ od}\}$
$\cup \{P | \exists P_1 \in \mathcal{W}_j, P_2 \in \mathcal{W}_k : j + k \leq n - 1 : P = P_1; P_2\}$
$\mathcal{W}_n = \{\text{WHILE programs built by applying rules 2 or 3 at most } n \text{ times}\}$
$\mathcal{W} = \cup_{n \in \mathbb{N}} \mathcal{W}_n$
WHILE-Semantics: -$\ell = \ell(P)$: Largest index of a variable in P, -state: vector from $\mathbb{N}^{\ell+1}$, -pad with 0 to match lengths
- Φ_P maps state vectors $S = (\sigma_0, \ldots, \sigma_\ell)$ to state vectors
$P = x_i := x_j - x_k : \Phi_P(S) := (\sigma_o, \ldots, \sigma_{i-1}, \max\{\sigma_j - \sigma_k, 0\}, \sigma_{i+1}, \ldots, \sigma_\ell)$
$P = \text{while } x_i \neq 0 \text{ do } P_1 \text{ od} : \Phi_P(S) = \begin{cases} \Phi_{P_1}^{(r)}(S) \text{ if } r \text{ exists and } \Phi_{P_1}^{(r)}(S) \text{ is defined,} \\ \text{undefined otherwise} \end{cases}$
for every WHILE program P, the function Φ_P is well-defined, non-termination possible if P does not halt, Φ_P is a partial function, can be undefined for some arguments
$\varphi_P : \mathbb{N}^s \to \mathbb{N}$ computed by P:
$\varphi_P(\alpha_1, \ldots, \alpha_s) = \begin{cases} = \text{first entry of } \Phi_P(\alpha_1, \ldots, \alpha_s, 0 \ldots, 0) \text{ if defined} \\ \text{undefined } (\bot) \text{ otherwise} \end{cases}$
a partial function $f : \mathbb{N}^s \to \mathbb{N}$ is WHILE-computable if there is a WHILE program P with $\varphi_P = f$, R $= \{f | f \text{ is WHILE-computable}\}$
partial function $f : A \to B$: -dom$(f) = \{x \in A | f(x) \text{ is defined}\} \subseteq A$ is the domain of f, -im$(f) = \{f(x) | x \in \text{dom}(f)\} \subseteq B$ is the image of f. total: dom$(f) = A$, partial: even if f is total, surjective: if im$(f) = B$, injective: if $f(x) = f(y)$ implies $x = y$ for all $x, y \in \text{dom}(f)$, bijective: if f is both injective and surjective
pairing functions $\langle \cdot, \cdot \rangle$ (bijective/injective functions $\mathbb{N}^2 \to \mathbb{N}$) allow us to restrict to functions $\mathbb{N} \to \mathbb{N}$
Theorem: $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ given by $(x, y) \mapsto \langle x, y \rangle = \frac{1}{2}(x+y)(x+y+1) + y$ is bijective. **Proof sketch:** let $p \in \mathbb{N}$ be given. choose $z = \max\{z' \in \mathbb{N} | \frac{1}{2} z'(z'+1) \leq p\}$. $y = \pi_2(p) = p - \frac{1}{2}z(z+1)$ and $x = \pi_1(p) = z - y$ result in $\langle x, y \rangle = p$ with $x, y \in \mathbb{N}$.
for WHILE-computable functions $\langle \cdot, \cdot \rangle, \pi_1, \pi_2$, $\langle \pi_1(z), \pi_2(z) \rangle = z$ for all $z \in \mathbb{N}$, $\pi_1(\langle x, y \rangle) = x$ and $\pi_2(\langle x, y \rangle) = y$ for all $x, y \in \mathbb{N}$.
Let $A = \langle a_1, \langle a_2, \ldots \langle a_{k-1}, a_k \rangle \ldots \rangle \rangle$. Acces a_i : $\pi_1 \circ \pi_2^{(i-1)}(A)$ if $i \leq k-1$ and $\pi_2^{(k-1)}(A)$ if $i = k$.
sets $L \subseteq \mathbb{N}$ (decision problem, language), characteristic function $\chi_L : \mathbb{N} \to \{0, 1\}$ with $\chi_L(x) = \begin{cases} 1 \text{ if } x \in L \text{ and} \\ 0 \text{ if } x \notin L. \end{cases}$ L is recursive/decidable/computable if $\chi_L \in$ R.
REC $= \{L \subseteq \mathbb{N} | \chi_L \in \text{R}\}$ - set of decidable/computable sets/languages. $L \in$ REC : there is a program that always terminates and, for all numbers x, answers correctly whether $x \in L$.
göd: $\mathcal{W} \to \mathbb{N}$ maps WHILE programs to numbers. göd is well-defined and injective. $\varphi_P = \varphi_{\text{göd}(P)}$ for short or $\varphi_g = \varphi_{\text{göd}^{-1}(g)}$ for $g \in \mathcal{G}$. $\mathcal{G} = \text{im(göd)} = \{\text{göd}(P) | P \in \mathcal{W}\} \subseteq \mathbb{N}$.
a set S is countable if there exists an injective function $f : S \to \mathbb{N}$, equivalently there is a surjective function $f : \mathbb{N} \to S$. S is countably infinite if there exists a bijective function $f : S \to \mathbb{N}$. By gödelization, the number of WHILE programs is countable, but the sets $L \subseteq \mathbb{N}$ is uncountable. Thus, there must be an $L \subseteq \mathbb{N}$ with $\chi_L \notin$ R and $L \notin$ REC. Then L is not decidable/computable/recursive.
Theorem: $\mathcal{P}(\mathbb{N})$ is uncountable. **Proof sketch:** assume to the contrary that there exists a surjective function $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$. Let $S = \{i \in \mathbb{N} | i \notin f(i)\}$. We have $S \neq f(i)$ for all $i \in \mathbb{N}$ by construction. Contradiction - f is not surjective.
Halting problem: $H = \{\langle g, x \rangle | g \in \mathcal{G} \text{ and göd}^{-1}(g) \text{ halts on input } x\}$.
Special halting problem: $H_0 = \{g | g \in \mathcal{G} \text{ and göd}^{-1}(g) \text{ halts on input } g\}$.
Theorem: $H, H_0 \notin$ REC. **Proof sketch:** assume to the contrary that $H_0 \in$ REC; then $\chi_{H_0} \in$ R. then $f \in$ R with $f(x) = \begin{cases} 1 \text{ if } \chi_{H_0}(x) = 0 \text{ and} \\ \text{undefined if } \chi_{H_0}(x) = 1. \end{cases}$ since $f \in$ R, there is a g with $\varphi_g = f$. we obtain $f(g) = 1 \iff f(g)$ is undefined - a contradiction. Thus, $H_0 \notin$ REC. $\chi_{H_0}(x) = \chi_H(\langle x, x \rangle)$ for all x yields $H \notin$ REC.(reduction from H_0 to H).

Lecture 2

U : universal WHILE program. $H = \text{dom}(\varphi_U) \notin$ REC.
$\langle g, \langle x, t \rangle \rangle \mapsto \begin{cases} 1 \text{ if god}^{-1}(g) \text{ halts on } x \text{ after } \leq t \text{ steps} \\ 0 \text{ otherwise} \end{cases}$ is computable.
$L \subseteq \mathbb{N}$ is recursively enumerable (semi-decidable) if there is a WHILE program P with $\varphi_P(x) = \begin{cases} 1 \text{ if } x \in L \text{ and} \\ 0 \text{ or undefined if } x \notin L \end{cases}$ for all $x \in \mathbb{N}$.
RE $= \{L \subseteq \mathbb{N} | L \text{ is recursively enumerable}\}$. if $x \in L$, then we will eventually know. $H, H_0 \in$ RE.
co-RE $= \{\overline{L} | L \in \text{RE}\}$, where $\overline{L} = \mathbb{N} \setminus L$. if $x \notin L$, then we will eventually know. co-RE $\neq \mathcal{P}(\mathbb{N}) \setminus$ RE. REC = RE $\cap$ co-RE, consequence: $H, H_0 \notin$ co-RE. If $A, B \in$ REC, then $\overline{A}, A \cap B, A \cup B \in$ REC.
Theorem: REC = RE $\cap$ co-RE **Proof:** $\subseteq$ follows from REC $\subseteq$ RE and REC $\subseteq$ co-RE. $\supseteq$: Let $L \in$ RE $\cap$ co-RE. there are programs P_{RE} and $P_{\text{CO-RE}}$ that show this. to check if $x \in L$: run $P_{\text{RE}}(x)$ and $P_{\text{CO-RE}}(x)$ alternatingly for 1,2,3,... steps, until one output an answer. output 1 or 0 accordingly.
Theorem: if $A, B \in$ RE, then $A \cap B, A \cup B \in$ RE. **Proof sketch:** $A \cap B$: simulate P_A on x, then simulate P_B on x. if both output 1, then output 1. non-termination is no problem. $A \cup B$: simulate P_A and P_B on x alternately. output 1 if one of them outputs 1.

```
program Q_42
input: g ∈ ℕ
1: if g ∉ G then
2:     loop forever
3: else
4:     for z = 0,1,2,... do
5:         x := π1(z); t := π2(z)
6:         simulate P = göd^-1(g) on input x for t steps
7:         if P halts within t steps and outputs 42 then
8:             return 1
```

dom$(\varphi_{Q_{42}}) = L_{42}$

$L_{42} = \{g \in \mathcal{G} | 42 \in \text{im}(\varphi_g)\} \in$ RE.
Proof sketch: $L = \{g \in \mathbb{N} | Q_{42}(g) = 1\}$ - we have to show $L = L_{42}$. $L \subseteq L_{42}$: Let $g \in L$. Then $Q_{42} = 1$. Thus there exists $x, t \in \mathbb{N}$ s.t. göd$^{-1}(g)$ halts on x after $\leq t$ steps and $\varphi_g(x) = 42$. Hence, $g \in L_{42}$. $L \supseteq L_{42}$: Let $g \in L_{42}$. Then there exists an $x \in \mathbb{N}$ with $\varphi_g(x) = 42$. This means that there is a $t \in \mathbb{N}$ s.t. göd$^{-1}(g)$ halts on x after $\leq t$ steps. Since lines 5-9 of Q_{42} always terminate, Q_{42} halts and outputs 1 for $z = \langle x, t \rangle$ or earlier. Thus, $g \in L$.

1. $L \in$ RE $\iff$ 2. there exists a WHILE program P with dom$(\varphi_P) = L$ $\iff$ 3. there exists a WHILE program P with im$(\varphi_P) = L$ $\iff$ 4. Either $L = \emptyset$ or there exists a WHILE program P that always terminates and satisfies im$(\varphi_P) = L \iff$ 5. $L \leq H_0$.

```
program Q'_42
input: z = ⟨g, x⟩ ∈ ℕ
1: if g = π1(z) ∉ G then
2:     loop forever
3: else
4:     simulate P = göd^-1(g) on input x = π2(z)
5:     if P outputs 42 then
6:         return g
7:     else
8:         loop forever
```

im$(\varphi_{Q'_{42}}) = L_{42}$

```
program Q''_42
input: z = ⟨g, ⟨x, t⟩⟩ ∈ ℕ
1: if g = π1(z) ∉ G then
2:     return g0
3: else
4:     x = π1(π2(z)); t = π2(π2(z))
5:     simulate P = göd^-1(g) on input x for t steps
6:     if P halts within t steps and outputs 42 then
7:         return g
8:     else
9:         return g0
```

im$(\varphi_{Q''_{42}}) = L_{42}$
Q''_{42} always terminates

Let $A, B \subseteq \mathbb{N}$, $f : \mathbb{N} \to \mathbb{N}$ is called a many-one reduction from A to B if f is total and computable and $x \in A \iff f(x) \in B$ for all $x \in \mathbb{N}$. If f exists, then A is ((recursively) many-one) reducible to B and write $A \leq B$. Then if A is difficult, then B is difficult. If B is easy, then A is easy.
$A \leq B$, then $\overline{A} \leq \overline{B}$. If $A \leq B$ and $B \leq C$, then $A \leq C$ (transitivity). **Theorem:** For CLASS $\in \{\text{REC}, \text{RE}, \text{CO-RE}\}$: if $A \leq B$ and $B \in$ CLASS, then $A \in$ CLASS. if $A \leq B$ and $A \notin$ CLASS, then $B \notin$ CLASS. **Proof sketch:** Let f be a reduction form A to B. 1. $B \in \text{REC} \implies \chi_B \in \text{R} \implies \chi_A = \chi_B \circ f \in \text{R} \implies A \in \text{REC}$, 2. $B \in \text{RE} \implies$ there exists a function $h \in$ R with dom$(h) = B \implies h \circ f \in$ R and dom$(h \circ f) = A \implies A \in$ RE.
$H_0 \leq H : x \mapsto f(x) = \langle x, x \rangle$ shows this, since $x \in H_0 \iff f(x) \in H$ for all $x \in \mathbb{N}$: ($\implies$): if $x \in H_0$, then $x \in \mathcal{G}$ and $x \in$dom(φ_x). This implies $\langle x, x \rangle = f(x) \in H$. ($\impliedby$) : if $f(x) = \langle x, x \rangle \in H$, then $x \in$dom(φ_x). Hence, $x \in H_0$.
SURJ $= \{g \in \mathcal{G} | \text{im}(\varphi_g) = \mathbb{N}\}$. Let $f : \mathbb{N} \to \mathbb{N}$ be given by $(\star) := g \mapsto f(g) = \begin{cases} g \text{ if } g \notin \mathcal{G} \text{ and} \\ \tilde{g} \text{ if } g \in \mathcal{G}, \end{cases}$ where $\tilde{g}$ is the Gödel number of the program:
input: x
1: run göd$^{-1}(g)$ on input g
2: output x
f is clearly WHILE-computable. $g \in H_0 \iff f(g) \in$ SURJ: ($\implies$): $g \in H_0 \implies g \in \mathcal{G}$ and $f(g) = \tilde{g} \implies \varphi_{\tilde{g}}(x) = x$ for all $x \in \mathbb{N} \implies f(g) \in$ SURJ. ($\impliedby$): $g \notin H_0$ - two cases: $g \notin \mathcal{G} : \implies f(g) = g \notin \mathcal{G} \supseteq \text{SURJ} \implies f(g) \notin$ SURJ. $g \in \mathcal{G} \setminus H_0 \implies \text{im}(\varphi_{\tilde{g}}) = \emptyset \implies f(g) \notin$ SURJ.
$H_0 \leq \overline{\text{SURJ}} : \overline{\text{SURJ}} = \mathbb{N} \setminus \text{SURJ} = \{g \in \mathcal{G} | \text{im}(\varphi_g) \subsetneq \mathbb{N}\} \cup (\mathbb{N} \setminus \mathcal{G})$.
Let $g_{\text{SURJ}} \in$ SURJ be any fixed element of SURJ $\neq \emptyset$. Let f be given by $f(g) = \begin{cases} g_{\text{SURJ}} \text{ if } g \notin \mathcal{G} \text{ and} \\ \tilde{g} \text{ if } g \in \mathcal{G}, \end{cases}$ where $\tilde{g}$ is the Gödel number of the following program:
input: x
1: if göd$^{-1}(g)$ does not stop within x steps on input g then
2: output x
3: else
4: loop forever
f is clearly WHILE-computable. $g \in H_0 \iff f(g) \in \overline{\text{SURJ}}$: ($\implies$): $g \in H_0 \implies$ göd$^{-1}(g)$ halts on g after t steps for some $t \in \mathbb{N} \implies \text{im}(\varphi_{\tilde{g}}) = \{0, 1, 2, \ldots, t-1\} \neq \mathbb{N} \implies f(g) \notin$ SURJ. ($\impliedby$): $g \notin H_0$ - two cases: $g \notin \mathcal{G} : \implies f(g) = g_{\text{SURJ}} \in$ SURJ. $g \in \mathcal{G} \setminus H_0 : \implies \text{im}(\varphi_{\tilde{g}}) = \mathbb{N} \implies f(g) \in$ SURJ.
$H_0 \leq$ SURJ: SURJ $\notin$ REC and SURJ $\notin$ co-RE. $H_0 \leq \overline{\text{SURJ}} : \overline{\text{SURJ}} \notin$ REC (known) and $\overline{\text{SURJ}} \notin$ co-RE, SURJ $\notin$ RE. So SURJ is even more difficult than H_0 and H.

Lecture 3

$L \subseteq \mathcal{G}$ is an index set if $i \in L$ and $\varphi_i = \varphi_j$ implies $j \in L$ for all $i, j \in \mathcal{G}$. L is a non-trivial index set if L is an index set and $\emptyset \subsetneq L \subsetneq \mathcal{G}$. So L is an index set if there is a set $F \subseteq$ R of functions with $L = \{i \in \mathcal{G} | \varphi_i \in F\}$.
Lemma: Let $U \subseteq \mathcal{G}$ and $L \subseteq \mathbb{N}$. Then $U \cap L \in$ REC if and only if $U \setminus L \in$ REC. (This implies $L \in$ REC $\iff \mathcal{G} \setminus L \in$ REC for all index sets L) **Proof sketch:** $\chi_{U \setminus L}(x) + \chi_{U \cap L}(x) = \chi_U(x)$ for all $x \in \mathbb{N}$.
Lemma: If L is (non-trivial) index set, then $\mathcal{G} \setminus L$ is a (non-trivial) index set. (note: if $L \subseteq \mathcal{G}$, then $\mathcal{G} \setminus L \neq \overline{L} = \mathbb{N} \setminus L$) **Proof sketch:** Let $i \in \mathcal{G} \setminus L$ and $j \in \mathcal{G}$ with $\varphi_i = \varphi_j$. If $j \in L$, then $i \in L$ since L is an index set. Hence, $j \in \mathcal{G} \setminus L$. We conclude that $\mathcal{G} \setminus L$ is an index set and is non-trivial if and only if L is non-trivial.
Rice's Theorem: Every non-trivial index set is undecidable. **Proof:** Let $L \subseteq \mathcal{G}$ be an arbitrary, non-trivial index set. NEVER $= \{g \in \mathcal{G} | \text{dom}(\varphi_g) = \emptyset\}$. L is index set $\implies$ NEVER $\subseteq L$ or NEVER $\subseteq \mathcal{G} \setminus L$. We assume w.l.o.g. NEVER $\subseteq \overline{L}$ (either both $L, \mathcal{G} \setminus L \in$ REC or none). L is non-trivial $\implies$ there exists a $g_0 \in L$ with dom$(\varphi_{g_0}) \neq \emptyset$. We will show $H_0 \leq L$. Let $f : \mathbb{N} \to \mathbb{N}$ be given by $(\star)$, where $\tilde{g}$ is the Gödel number of the following program:
input: x
1: run göd$^{-1}(g)$ on input g
2: compute and output $\varphi_{g_0}(x)$
$g \notin H_0 \implies f(g) \notin L$: Either $g \notin \mathcal{G}$ (clear) or $g \in \mathcal{G} \setminus H_0$. Then line 1 of göd$^{-1}(\tilde{g})$ does not terminate. Hence, dom$(\varphi_{\tilde{g}}) = \emptyset$.
$g \in H_0 \implies f(g) \in L$: göd$^{-1}(\tilde{g})$ computes φ_{g_0} and $g_0 \in L$. Since L is an index set, this implies $\tilde{g} \in L$.
f is WHILE-computable and total.
s-m-n Theorem: For every $m, n \geq 1$, there is a computable function $S_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ s.t. for all $g \in \mathcal{G}, y \in \mathbb{N}^m$, and $z \in \mathbb{N}^m$, we have $\varphi_g^{m+1}(y, z) = \varphi_{S_n^m(g,y)}^n(z)$.
example: if we have $(x, y) \mapsto x + y$, then we get $y \mapsto 5 + y$ in a systematic way.
Recursion Theorem: For every computable function $f : \mathbb{N}^{n+1} \to \mathbb{N}$, there is a $g \in \mathcal{G}$ s.t. $\varphi_g^n(z) = f(g, z)$ for all $z \in \mathbb{N}^n$.
Hence, there exists a program that outputs its own Gödel number and H_0 is no index set.
Fixed Point Theorem: For all computable, total functions $f : \mathbb{N} \to \mathbb{N}$ with im$(f) \subseteq \mathcal{G}$ and for all $n \in \mathbb{N} \setminus \{0\}$, there is an $e \in \mathcal{G}$ with $\varphi_{f(e)}^n = \varphi_e^n$.
Hence, there is no program that modifies all programs.
WHILE-computable = GOTO-computable = Turing-computable = ...
Church-Turing Thesis: Something can be computed by a sufficient powerful computing device if and only if it can be computed by a Turing machine.
asymptotic growth of functions $f, g : \mathbb{N} \to \mathbb{R}_0^+$:
$f = O(g) : \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 : f(n) \leq cg(n)$
$f = o(g) : \forall c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 : f(n) \leq cg(n)$
$f = \Omega(g) : \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 : f(n) \geq cg(n)$
$f = \omega(g) : \forall c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 : f(n) \geq cg(n)$
$f = \Theta(g) : f = O(g)$ and $f = \Omega(g)$.

Lecture 4

time:
$\text{TIME}_M(x) = $ #steps that DTM M takes on input x (can be infinite). $\text{TIME}_M(n) = \max\{\text{TIME}_M(x) : |x| = n\}$. TM M is t time bounded if $\text{TIME}_M(n) \leq t(n)$ for all $n \in \mathbb{N}$.
space
$\text{SPACE}_M(x) = $ #cells used by M on input x. $\text{SPACE}_M(x) = \max\{\text{SPACE}_M(x) : |x| = n\}$. TM M is s space bounded if $\text{SPACE}_M(n) \leq s(n)$ for all $n \in \mathbb{N}$.
$t, s : \mathbb{N} \to \mathbb{R}_{\geq 0}$ are functions.
Nondeterministic Turing Machines: replace $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, S, R\}^k$ by $\delta : Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^k \times \{L, S, R\}^k)$. Replace computation path by computation tree. $L(M) = \{x \in \Sigma^* :$ there is at least on path from the starting configuration to some accepting configura
$\text{TIME}_M(x)$ =shortest accepting computation path. $\text{TIME}_M(n) = \max\{\text{TIME}_M(x) : |x| = n, x \in L(M)\}$(0 if no such x exist). NTM M is weakly t time bounded if $\text{TIME}_M(n) \leq t(n)$ for all n. NTM M is strongly t time bounded if every computation path on every input x of length n is bounded by $t(n)$ for all n.
(N/)DTIME$(t) = \{L : L = L(M)$ for some t time bounded (N/)DTM $M\}$.
(N/)DTIME$_k(t) = \{L : L = L(M)$ for some t time bounded $k-$tape (N/)DTM $M\}$.
(N/)DSPACE$(s) = \{L : L = L(M)$ for some s space bounded (N/)DTM $M\}$.
(N/)DTIME$(t) = \{L : L = L(M)$ for some s space bounded $k-$tape (N/)DTM $M\}$.
DTIME$(T) = \cup_{t \in T}$DTIME$(t), \ldots$.
Tape reduction Theorem: For all $s, t : \mathbb{N} \to \mathbb{N}$: DTIMESPACE$(t, s) \subseteq$ DTIMESPACE$_1(O(ts), O(s))$ and NTIMESPACE$(t, s) \subseteq$ NTIMESPACE$_1(O(ts), O(s))$
Quadratic simulation by 1-tape TMs Corollary: For all $t : \mathbb{N} \to \mathbb{N}$: DTIME$(t) \subseteq$ DTIME$_1(O(t^2))$ and $t : \mathbb{N} \to \mathbb{N}$: NTIME$(t) \subseteq$ NTIME$_1(O(t^2))$.
Tape compression Theorem: For all $0 < \epsilon \leq 1$ and $s : \mathbb{N} \to \mathbb{N}$: DSPACE$(s(n)) \subseteq$ DSPACE$_{1,E}(\lceil \epsilon \cdot s(n) \rceil)$ and NSPACE$(s(n)) \subseteq$ NSPACE$_{1,E}(\lceil \epsilon \cdot s(n) \rceil)$
Accelaration Theorem: For all $k \geq 2$, all $t : \mathbb{N} \to \mathbb{N}$, and $0 < \epsilon \leq 1$: DTIME$_k(t(n)) \subseteq$ DTIME$_k(n + \epsilon(n + t(n)))$ and NTIME$_k(t(n)) \subseteq$ NTIME$_k(n + \epsilon(n + t(n)))$
$t : \mathbb{N} \to \mathbb{N}$ is time constructible if there is an $O(t)$ time bounded DTM that computes the function $1^n \mapsto \text{bin}(t(n))$.
$s : \mathbb{N} \to \mathbb{N}$ is space constructible if there is an $O(s)$ space bounded DTM that computes $1^n \mapsto \text{bin}(s(n))$.
Consequence: on input x, writes $1^{t(|x|)}$ (or $1^{s(|x|)}$) on one tape.
Lemma: Let t be time constructible, and let s be space constructible. If $L \in$ NTIME(t), then there exists a strongly $O(t)$ time bounded NTM with $L(M) = L$ and if $L \in$ NSPACE(s), then there exists a strongly $O(s)$ space bounded NTM with $L(M) = L$.
Configuration graph: fll description of current situation: current state, content of all tapes, head positions.
Number of configuratons: $|Q| \cdot (|\Gamma|^s)^k \cdot s^k$ or c^s for some constant c for $s \geq \log n$.
Configuration graph: directed graph, nodes = configurations, edge (C, C') if TM goes from C to C' in a single step. DTM implies outdegree ≤ 1. NTM accepts x if there exists a path from $SC(x)$ to some accepting configuration.
Corollary: Let $s(n) \geq \log n$. If an s space bounded DTM halts on input x, then it performs $\leq c^{s(|x|)}$ steps on input x.
Deterministic space is closed under complement Corollary: Let $s(n) \geq \log n$. If $L \in$ DSPACE(s), then $\overline{L} \in$ DSPACE(s).(Deterministic time classes are trivally closed under complement.**Theorem:** Let $(n) \geq \log n$ be space constructible. Then DSPACE$(s) \subseteq$ NSPACE$(s) \subseteq$ DTIME$(2^{O(s)})$. **Theorem:** Let t be time constructible. Then DTIME$(t) \subseteq$ NTIME$(t) \subseteq$ NSPACE$(t) \subseteq$ DTIME$(2^{O(t)})$.

Lecture 5

Savitch's Theorem: Let s be space constructible with $s(n) \geq \log n$. Then $\text{NSPACE}(s) \subseteq \text{DSPACE}(O(s^2))$. **Proof sketch:** In the configuration graph: $L \in \text{NSPACE}(s) \implies$ configuration graph of NTM has $\leq c^s$ nodes $\implies$ reachability (starting configuration to accepting configuration) can be decided in space $O((\log c^s)^2) = O(s^2) \implies L \in \text{DSPACE}(s^2)$.

Deterministic space hierarchy Theorem: Let $s_2(n) \geq \log n$ be space constructible, and let $s_1 = o(s_2)$. Then $\text{DSPACE}(s_1) \subsetneq \text{DSPACE}(s_2)$. (more space, more power)(proof idea via diagonalization)(same theorem holds for NSPACE)

Proof:

```
DIAG (x = [g, y])
mark s_2(|x|) cells
run M = göd^{-1}_TM(g) on x
if M runs for more than 2^{s_2(|x|)} steps then accept
if M goes out of bounds then reject
if M accepts then reject
if M rejects then accept
```

TM DIAG is s_2 space bounded.

$L(M) \neq L(\text{DIAG})$ if M is s_1 space bounded: - $g = \text{göd}^{-1}_{\text{TM}}(M)$; choose y sufficiently long; $x = [g, y]$. - $x \in L(\text{DIAG})$: M needs $\geq 2^{s_2(|x|)} \gg c^{s_1(|x|)}$ steps, hence does not terminate or - M rejects, hence $x \notin L(M)$. $x \notin L(\text{DIAG})$: - M runs out of bounds, hence M not s_1 space bounded or - M accepts, hence $x \in L(M)$.

$\text{DTIME}(o(t)) \subseteq \text{DSPACE}(o(t)) \subsetneq \text{DSPACE}(t) \subseteq \text{DTIME}(2^{O(t)})$.

$\text{NTIME}(o(t)) \subseteq \text{NSPACE}(o(t)) \subsetneq \text{DSPACE}(o(t^2))$ (Savitch)

$\text{NTIME}(o(t)) \subsetneq \text{DSPACE}(t^2)$ (space hierarchy)

$\text{NTIME}(o(t)) \subseteq \text{DTIME}(2^{O(t^2)})$.

Deterministic time hierarchy Theorem: Let t_2 be time constructible, and let $t_1^2 = o(t_2)$. Then $\text{DTIME}(t_1) \subsetneq \text{DTIME}(t_2)$.

Stronger deterministic time hierarchy Theorem: same as above, but weaker condition $t_1 \cdot \log t_1 = o(t_2)$.

Borodin's Gap Theorem: Let f be recursive with $f(n) \geq n$ for all n. Then there are total, recursive functions $t, s : \mathbb{N} \to \mathbb{N}$ with $s(n) \geq n, t(n) \geq n$ s.t. $\text{DTIME}(f(t(n))) = \text{DTIME}(t(n))$ and $\text{DSPACE}(f(s(n))) = \text{DSPACE}(s(n))$. (Theorem implies that constructability is necessary for hierarchies).

$\text{L} = \text{DSPACE}(O(\log n))$. $\text{NL} = \text{NSPACE}(O(\log n))$. $\text{P} = \cup_c \text{DTIME}(O(n^c))$.

$\text{NP} = \cup_c \text{NTIME}(O(n^c))$. $\text{co-NP} = \{L : \overline{L} \in \text{NP}\}$. $\text{PSPACE} = \cup_c \text{DSPACE}(O(n^c))$.

$\text{EXP} = \cup_c \text{DTIME}(2^{O(n^c)})$.

$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP}$ and $\text{co-NP} \subseteq \text{PSPACE} \subseteq \text{EXP}$.

$\text{NL} \subsetneq \text{PSPACE}$ and $\text{P} \subsetneq \text{EXP}$.

Polynomial-time verifier M for L : there exists a polynomial p s.t.
- for all $x \in L$, there exists a $c \in \{0,1\}^\star$ with $|c| \leq p(|x|)$ s.t. M accepts $[x, c]$; (c is called a certificate/witness/proof)
- for all $x \notin L$ and all $c \in \{0,1\}^\star$ with $|c| \leq p(|x|)$, M rejects $[x, c]$;
- M runs in polynomial time.

Theorem: $L \in \text{NP}$ if and only if there is a polynomial-time verifier for L.

Proof: ($\implies$): certificate: string that describes the branches of the NTM, vertifier: simulate branch of NTM using the certificate. ($\impliedby$): guess a certificate using non-deterministic branching and check validility using verifier.

$f : \Sigma^\star \to \Sigma^\star$ is called a polynomial-time many-one reduction from $A \subseteq \Sigma^\star$ to $B \subseteq \Sigma^\star$ if f is polynomial-time computable and $x \in A \iff f(x) \in B$ for all $x \in \Sigma^\star$. Then A is polynomial-time reducible to B if there exists such a function and $A \leq_{\text{P}} B$.

Transitivity of polynomial-time many-one reductions Theorem: If $A \leq_{\text{P}} B$ and $B \leq_{\text{P}} C$, then $A \leq_{\text{P}} C$.

Proof: $f : A \leq_{\text{P}} B$, time $O(n^a)$ and $g : B \leq_{\text{P}} C$, time $O(n^b)$, then $g \circ f$ shows $A \leq_{\text{P}} C$ with time $O(n^{ab})$.

Theorem: For CLASS $\in \{\text{P}, \text{NP}, \text{co-NP}, \text{PSPACE}, \text{EXP}, \ldots\}$: If $A \leq_{\text{P}} B$ and $B \in$ CLASS, then $A \in$ CLASS. (We say that the classes are closed under $\leq_{\text{P}}$). (Note that this doesn't work for NL or L, since there is not enough space.)

L is NP-hard if $A \leq_{\text{P}} L$ for all $A \in \text{NP}$.

L is NP-complete if L is NP-hard and $L \in \text{NP}$.

$\text{SAT} = \{\Phi : \Phi \text{ is satisfiable Boolean formula}\}$.

Lemma: if there is one NP-complete problem L with $L \in \text{P}$, then $\text{P} = \text{NP}$.

Lemma: if A is NP-hard and $A \leq_P B$, then B is NP-hard.

Cook, Karp, Levin Theorem: SAT is NP-complete.

Lecture 6

Boolean formula Φ is in kCNF if $\Phi = C_1 \wedge \ldots \wedge C_m$ for clause $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ldots \vee \ell_{i,k}$ for literals $\ell_{i,j} \in \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$.

$k\text{SAT} = \{\Phi : \Phi \text{ is in } k\text{CNF and satisfiable}\}$.

Theorem: kSAT is NP-complete for $k \geq 3$.

Theorem: $2\text{SAT} \in \text{P}$.

$\text{CSAT} = \{C : C \text{ is a satisfiable Boolean circuit}\}$

$L \leq_{\text{P}} \text{CSAT} \leq_{\text{P}} 3\text{SAT}$ for arbitrary $L \in \text{NP}$.

Proof sketch: encode polynomial-time verifier M for L as circuit. M accepts if and only if circuit is satisfiable. $\text{CSAT} \leq_{\text{P}} 3\text{SAT}$: direct transformation impossible.

$L \leq_{\text{P}}$ CSAT: polynomial-time TMs can be simulated by circuits of polynomial size. circuits: one circuit C_n for each input size n. $1^n \mapsto C_n$ can be computed in polynomial time (otherwise $C_i = \begin{cases} 1 \text{ if } i \in H_0, \\ 0 \text{ if } i \notin H_0 \end{cases}$) transition function: $D : \{0,1\}^q \times \{0,1\}^g \times \ldots \to \{0,1\}^q \times \ldots$

CSAT $\leq_{\text{P}}$ 3SAT: circuit $C \mapsto$ formula Φ. C has input gates g_j: g_j is input gate: variable, $g_j = \neg g_i$: $(a_i \vee a_j), (\neg a_i \vee \neg a_j)$. $g_j = g_i \wedge g_h$: $(\neg a_j \vee a_i), (\neg a_j \vee a_h), (a_j \vee \neg a_i \vee \neg a_h)$. $g_j = g_i \vee g_h$: can be expressed using $\neg$ and $\wedge$. output gate $g_j : a_j$.

CNF-SAT $\leq_{\text{P}}$ 3SAT:

$\text{CNF-SAT} = \{F : F \text{ is in CNF} \wedge F \in \text{SAT}\}$. reduction to 3SAT clause by clause: $C_i = \ell_1 \vee \ell_2 \vee \ldots \vee \ell_k$ is transformed to $(\ell_1 \vee \ell_2 \vee y_{i,1}), (\overline{y_{i,1}} \vee \ell_3 \vee y_{i,2}), \ldots, (\overline{y_{i,k-4}} \vee \ell_{k-2} \vee y_{i,k-3}), (\overline{y_{i,k-3}} \vee \ell_{k-1} \vee \ell_k)$.

clique of an undirected graph = complete subgraph

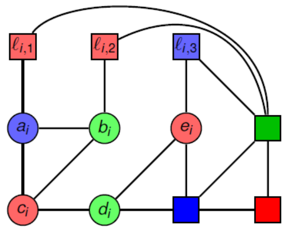
$\text{CLIQUE} = \{(G,k) : \text{undirected graph } G \text{ contains a clique of size } k\}$.

$\text{CLIQUE} \in \text{NP}$: input: (G,k), $G = (V,E)$. certificate: set $U \subseteq V$ of nodes, check if $|U| \geq k$ and if $\{u,v\} \in E$ for all distinct $u, v \in U$.

$3\text{SAT} \leq_{\text{P}}$ CLIQUE: instance for 3SAT : $\Phi = C_1 \wedge \ldots \wedge C_m$ with $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ and $\ell_{i,s} \in \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$. goal: polynomial-time computable function $\Phi \mapsto (G,k)$ with $\Phi \in 3\text{SAT} \iff (G,k) \in \text{CLIQUE}$. construction: $V = \{(i,s) : i \in \{1, \ldots, m\}, s \in \{1,2,3\}\}$, $E = \{\{(i,s),(j,t)\} : i \neq j \text{ and } \ell_{i,s} \neq \neg \ell_{j,t}\}$, $k = m$. clear: $\Phi \mapsto (G,k)$ is polynomial-time computable.

$\Phi \in 3\text{SAT} \implies (G,k) \in \text{CLIQUE}$: Φ satisfiable $\implies$ there exists an assignment that assigns 1 to at least one literal per clause. Let $\ell_{1,s_1}, \ell_{2,s_2}, \ldots, \ell_{m,s_m}$ be such literals. $\{(1,s_1),(2,s_2),\ldots,(m,s_m)\}$ is a k-clique of G because $1 = \ell_{i,s_i} \neq \neg \ell_{j,s_j} = 0$. Hence, $(G,k) \in \text{CLIQUE}$.

$(G,k) \in \text{CLIQUE} \implies \Phi \in 3\text{SAT}$: Let $U \subseteq V$ be a k-clique=m-clique of G, then $|U \cap \{(i,1),(i,2),(i,3)\}| \leq 1$ by construction and $|U \cap \{(i,1),(i,2),(i,3)\}| \geq 1$ since $|U| = m$. for $(i, s_i) \in U$, set $\ell_{i,s_i} = 1$: every clause has at least one literal, and no conflicts by construction. assign arbitrary values to remaining variables $\implies \Phi \in 3\text{SAT}$.



3SAT $\leq_{\text{P}}$ 3COLORING: gadget for clause $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$. rectangular nodes are reused, circular nodes are fresh for each clause. n variables, m clause: $2n + 5m + 3$ nodes. $\ell_{i,j} = x_k$ or $\ell_{i,j} = \overline{x_k}$ for some $k \in \{1, \ldots, n\}$. coding: true, false, third color.

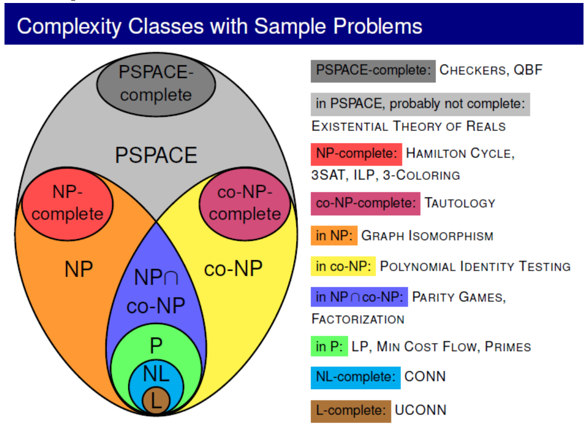
Williams' Theorem: Let $\phi \approx 1.618$ be the golden ratio. For every $\epsilon > 0$ and t, s with $t(n) \cdot s(n) = O(n^{\phi - \epsilon})$, we have $\text{SAT} \notin \text{DTIMESPACE}(t, s)$.

Exponential time hypothesis, ETH Conjecture: $\text{SAT} \notin \text{DTIME}(2^{o(n)})$.

Strong exponential time hypothesis, SETH Conjecture: For every $\epsilon > 0$, $\text{SAT} \notin \text{DTIME}((2-\epsilon)^n)$.

Lecture 7

Ladner's Theorem: $\text{P} \neq \text{NP} \implies$ there are problems in NP $\setminus$ P that are not NP-complete.



To show P=NP: choose your favorite NP-complete problem Π, prove that $\Pi \in \text{P}$. (hardly anybody believes that P=NP)

To show $\text{P} \neq \text{NP}$: choose your favorite NP-complete problem Π, prove that $\Pi \notin \text{P}$. (lower bounds are difficult to prove)

Encoding matters: 3COLORING is NP-complete, only for sane encodings: $L = \{x : x \in \{0,1\}^{n^2}$ is adjacency matrix of 3-colorable graph$\}$,

$\{x\#1^{2^n} : n \in \mathbb{N} \text{ and } x \in \{0,1\}^{n^2} \cap L\} \in \text{P}$ and $\{x : x$ encodes a circuit that encodes a 3-colorable graph$\}$ is NEXP-complete.

$f : \Sigma^\star \to \Sigma^\star$ is called a logarithmic-space many-one reduction from $A \subseteq \Sigma^\star$ to $B \subseteq \Sigma^\star$ if f is logarithmic-space computable and $x \in A \iff f(x) \in B$ for all $x \in \Sigma^\star$. Then A is logarithmic-space reducible to B is there exists such a function and we write $A \leq_{\log} B$.

Theorem: If $A \leq_{\log} B$ and $B \leq_{\log} C$, then $A \leq_{\log} C$.

Theorem: For CLASS $\in \{\text{L}, \text{NL}, \text{P}, \text{NP}, \text{co-NP}, \text{PSPACE}, \text{EXP}, \ldots\}$: If $A \leq_{\log} B$ and $B \in$ CLASS, then $A \in$ CLASS.

Theorem: Let $f, g : \Sigma^\star \to \Sigma^\star$ be computable in logarithmic space. Then $g \circ f$ is computable in logarithmic space.

Proof sketch: M_f computes f, M_g computes g: read-only input tape and write-only output tape. simulate M_g. if M_g wants to read symbol i from $f(x)$, then simulate M_f on x, ignore output, except for symbol i (counting is possible) and return. space usage: $O(\log|x|)$: for M_f as subroutine: $O(\log|x|)$ (reuse space) and for M_g: $O(\log|f(x)|)$ and $|f(x)| = O(n^c)$ for some c.

Generic NL-complete problem $\text{GENNL} = \{e\#x\#1^s : \text{göd}^{-1}_{\text{TM}}(e)$ is an NTM and accepts

Theorem: GENNL is NL-complete.

Proof: $\text{GENNL} \in \text{NL}$: input: $e\#x\#1^s$, $M = \text{göd}^{-1}_{\text{TM}}(e)$, every state and symbol of M needs at most space $|e|$, hence, M can be simulated in space $\log(s)/|e| \cdot |e| \leq \log n/$

GENNL is NL-hard: $A \in \text{NL}$, $M = \text{göd}^{-1}_{\text{TM}}(e)$ log-space NTM for A, M needs $c \cdot \log n$ space for some constant c, $x \mapsto f(x) = e\#x\#1^{(2^{c \cdot |e|})^{\log|x|}}$ (log-space computable). $x \in A \implies M$ accepts in space $c \cdot \log|x| = \log((2^{c \cdot |e|})^{\log|x|})/|e| \implies f(x) \in \text{GENNL}$. $x \notin A \implies M$ rejects $\implies f(x) \notin \text{GENNL}$.

Theorem: $\text{CONN} = \{(G,s,t) : G = (V,E) \text{ directed; contains } s-t \text{ path}\}$ is NL-complete.

Proof sketch: $\text{CONN} \in \text{NL}$: guess path, keep track of length. CONN is NL-hard: $\text{GENNL} \leq_{\log} \text{CONN}$, $x \mapsto$ instance for CONN: configuration graph of NLmachine for GENNL on x, start node = starting configuration, target node = unique accepting configuration (log-space computable)

Undirected graph connectvity: $\text{SL} = \{L : L \leq_{\log} \text{UCONN}\}$ and SL can be defined via symmetric log-space NTMs.

Reingold's Theorem: $\text{UCONN} \in \text{L}$. **Corollary:** SL=L.

Theorem: $2\text{SAT} \in \text{co-NL}$.

Proof sketch: $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, $C_i = \ell_{i,1} \vee \ell_{i,2}$, $\ell_{i,k} \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$, C_i can be written as $\overline{\ell_{i,1}} \to \ell_{i,2}$ or as $\overline{\ell_{i,2}} \to \ell_{i,1}$, $G = (V,E)$ with $V = \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$, $E = \{(\overline{\ell_{i,1}}, \ell_{i,2}), (\overline{\ell_{i,2}}, \ell_{i,1}) : 1 \leq i \leq m\}$,

Lemma: F is not satisfiable if and only if there is an i with x_i and $\overline{x_i}$ in the same strongly connected component (SCC) of G. **Proof:** ($\impliedby$): clear. ($\implies$): sort graph of G topologically. literal in the same SCC must get the same value. if there is a path from u to v, then there is a path from $\overline{v}$ to $\overline{u}$. consequences: if u is in a sink, then $\overline{u}$ is in a source and if C is a SCC, then $\overline{C} = \{\overline{u} : u \in C\}$ is a SCC. do iteratively: set literals in onse sink to 1, remove corresponding sink and source, no conflicts by choice of SCCs.

Lecture 8

Lemma: For all space-constructible s with $s(n) \geq \log n$: $\text{NL} = \text{co-NL} \implies \text{NSPACE}(s) = \text{co-NSPACE}(s)$.

Lemma: $\overline{\text{CONN}} \in \text{NL}$.

Lemma: NL = co-NL.

Immerman & Szelepcsényi's Theorem: For all space-constructible s with $s(n) \geq \log n$: $\text{NSPACE}(s) = \text{co-NSPACE}(s)$.

Inductive counting preparation: input: directed graph $G = (V,E)$ and vertices $s, t \in V$, notation: $N_d = N_d(s) = \{v \in V : G \text{ contains an } s-v \text{ path with at most } d \text{ edges}\}$, $n_d = n_d(s) = |N_d|$, $\text{DIST} = \{\langle G,s,v,d\rangle : v \in N_d\}$, $\langle G,s,v,d,n_d\rangle \in \text{NEGDIST} \iff v \notin N_d$. observations: NEGDIST is not really a set and NEGDIST, DIST $\in$ NL.

Inductive counting:

input: $\langle G, s, d\rangle$

output: $n_d = |N_d|$

```
n_0 := 1
for i := 0 to d − 1 do
    c := 0
    for each v ∈ V do
        guess if v ∈ N_{i+1}
        if v ∈ N_{i+1} was guessed then
            if ⟨G, s, v, i + 1⟩ ∈ DIST then c := c + 1
            else rejects
        else
            for all u ∈ V with u = v or (u, v) ∈ E do
                if ⟨G, s, u, i, n_i⟩ ∉ NEGDIST then reject
    n_{i+1} := c
```

Corollary: co-NL = NL.

Corollary: co-NPSPACE=NPSPACE(=PSPACE)

Second LBA problem, Corollary: $L \in \text{CSL} \implies \overline{L} \in \text{CSL}$

Time-constructable Translation Theorem: For $M, N \in \{\text{DTIME}, \text{NTIME}, \text{co-NTIME}, \text{DSPACE}, \text{NSPACE}, \text{co-NSPACE}\}$, if t_1, t_2, f time constructable with $t_1(n), t_2(n) \geq (1+\epsilon) \cdot n$ and $f(n) \geq n$, then $M(t_1) \subseteq N(t_2) \implies M(t_1 \circ f) \subseteq N(t_2 \circ f)$

Proof: $L \in M(t_1 \circ f) \implies \tilde{L} = \{x\#1^{f(|x|)-1-|x|} : x \in L\} \in M(t_1)$: linear-time counting (syntax check) and check if $x \in L$ can be done in $M(t_1 \circ f(|x|)) = M(t_1(\text{“input length”}))$. by assumption: $\tilde{L} \in N(t_2)$. on input x : generate $x\#1^{f(|x|)-1-|x|}$ and simulate machine for $\tilde{L} \in N(t_2)$. conclusion: $L \in N(t_2 \circ f)/$

Space-constructable Translation Theorem: For $M, N \in \{\text{DSPACE}, \text{NSPACE}, \text{co-NSPACE}\}$, if s_1, s_2, f space constructable with $s_1(n), s_2(n) \geq \log n$ and $f(n) \geq n$, then $M(s_1) \subseteq N(s_2) \implies M(s_1 \circ f) \subseteq N(s_2 \circ f)$

Proof sketch: similar to the Time-constructable Translation Theorem via $\tilde{L} = \{x\#1^{f(|x|)-1-|x|} : x \in L\}$. issue: not enough space to store $x\#1^{f(|x|)-1-|x|}$. solution: pretend that input is $x\#1^{f(|x|)-1-|x|}$ using counter and run machine for $\tilde{L}$.

Corollary: for space-constructable $s \geq \log n$: $\text{NL} = \text{co-NL} \implies \text{NSPACE}(s) =$

CO-NSPACE(s).

Proof: choose $M = \text{NSPACE}$, $N = \text{CO-NSPACE}$, $s_1 = s_2 = \log n$ and $f = 2^s$.

Consequences of translation: - CONN $\in$ DSPACE$((\log n)^2)$ implies Savitch's theorem. - CONN $\in$ co-NL implies theorem of Immerman and Szelepcsényi. - P = NP implies EXP = NEXP and EEXP = NEEXP and . . . - EXP $\subsetneq$ NEXP implies P $\subsetneq$ NP. - L = NP implies PSPACE = EXP.

Preparation Ladner's Theorem (Lecture 7): $M_1, M_2, \ldots$: enumeration of TMs M_i, running in time n^i (equip M_i with a counter - diagonalize for $\notin$ P). $f_1, f_2, \ldots$: enumeration of functions f_i computable in time n^i (equip TMs with counters - diagonalize against reductions). computable in time n^i, then infinitely many TMs witness this-thus $L \in$ P if and only if $L = L(M_i)$ for some i and f is polynomial-time computable if and only if $f = f_i$ for some i.

Proof by Padding: $B = \{x\#1^{f(|x|)-|x|-1} : x \in A\}$. increasing f makes B simpler. $f(n)$ super-polynomial in n forbids $A \leq_{\text{P}} B$. $f(n)$ computable in time polynomial in $f(n)$ allows $B \leq_{\text{P}} A$. diagonalization forbids $b \in$ P.

```
i ← 1
for all y with |y| ≤ log log n do
    if y ∈ L(M_i)ΔB then i ← i + 1
f(n) ← n^i
```

f is computable in time polynomial in $f(n)$ (not in n): $y \in B$ by brute force and $y \in L(M_i)$ using counter, use previous values of f to check syntax for y.

$B \leq_{\text{P}} A : y \mapsto h(y) = \begin{cases} x \text{ if } y = x\#1^{f(|x|)-|x|-1} \\ z_0 \notin A \text{ otherwise} \end{cases}$ is computable in time polynomial in $f(|x|) = |y|$. Hence, $y \in B \iff h(y) \in A$ by construction.

$B \notin$ P: if $B \in$ P, then $B = L(M_i)$ for some i; choose smallest such i. then $f(n) = n^i$ for all sufficiently large n, then $x \mapsto x\#1^{f(|x|)-|x|-1}$ shows that $A \leq_{\text{P}} B$ (also because f is then computable in polynomial time), but then $A \in$ P - a contradiction.

$A \not\leq_{\text{P}} B$: if $A \leq_{\text{P}} B$, then there is a reduction g_j; we have $g_j(x) \leq |x|^j$. $B \neq L(M_i)$ for all i (otherwise $B \in$ P); thus, there is an n_j with $g(n) \geq n^{i+1}$ for all $n \geq n_j$.

```
if |x| < n_j then decide x ∈ A by table look-up
else if |g_j(x)| ∉ im(f) then g_j(x) ∉ B (testable since f monotone)
otherwise g_j(x) = y#1^{f(|y|)-|y|-1} and
```

$f(|y|) = |g_j(x)| \leq |x|^j$ and $f(|y|) \geq |y|^{j+1}$;

this implies $|y| \leq |x|^{\frac{j}{j+1}}$ and you can solve $x \in A$ recursively.

Proof by Holes: $A = \{x : x \in \text{3SAT and } h(|x|) \text{ is even}\}$. if h is polynomial-time computable, then $A \in$ NP. events: $F_i : A \neq L(M_i)$ and $R_i : x \in$ 3SAT x-or $f_i(x) \in A$. idea: monotone f, $h(n) = 2i$ until F_i (if F_i is never satisfied, then $A \in$ P and $|A\Delta\text{3SAT}|$ is finite). $h(n) = 2i + 1$ until R_i (if R_i is never satisfied, then A is finite and 3SAT$\leq_{\text{P}} A$ via f_i). $h(0) = h(1) = 2$; if $(\log n)^{h(n)} \geq n$, then $h(n+1) = h(n)$. $h(n) = 2i$: check if there is an x with $|x| \leq \log n$ with $M_i(x)$ accepts and $[h(|x|)$ is odd or $x \notin$ 3SAT$]$ or $M_i(x)$ rejects and $[h(|x|)$ is even and $x \in$ 3SAT$]$. $h(n) = 2i + 1$: check if there is an x with $|x| \leq \log n$ with $x \in$ 3SAT and $[h(|f_i(x)|)$ is odd or $f_i(x) \notin$ 3SAT$]$ or $x \notin$ 3SAT and $[h(|f_i(x)|)$ is even and $f_i(x) \in$ 3SAT$]$. if yes, then $h(n+1) = h(n) + 1$; if no, then $h(n+1) = h(n)$. to compute h, we only need x with $|x| \leq \log n$ and $|x|^i \leq (\log n)^i \leq (\log n)^{h(n)} < n$. hence, h is polynomial-time computable. h does not increase until the corresponding F_i or R_i are satisfied. if h remains constant, then P = NP.

Exercise Sheet 1:

1 WHILE Programs

For convenience, we would like to add other, more powerful commands to the WHILE language without increasing its power. Give short WHILE programs for the following extensions: (a)"`xi := xj^xk`", where "a^b" means "a raised to the power of b". (b)"`if xi = 0 then P1 else P2 endif`" for WHILE programs P1 and P2. (c)"`xi := xj div xk`", where $a \div b = \lfloor a/b \rfloor$. For a real number x, "$\lfloor x \rfloor$" denotes x rounded down to the nearest integer, i.e., we have $\lfloor x \rfloor = \max\{y \in \mathbb{Z} \mid y \leq x\}$. (d)"`xi := xj mod xk`", where mod denotes modulo (we have $x_j = x_k \cdot (x_j \div x_k) + x_j \bmod x_k$).

2 Collatz Conjecture

The Collatz conjecture is a conjecture for a certain class of sequences $(a_n)_{n\in\mathbb{N}}$. For a certain start value $a_0 \in \mathbb{N} \setminus \{0\}$, we have $a_{n+1} = \begin{cases} a_n/2 & \text{if } a_n \text{ is even,} \\ 3a_n + 1 & \text{if } a_n \text{ is odd.} \end{cases}$ For instance, for $a_0 = 6$, we obtain the sequence 6, 3, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, . . . The Collatz conjecture states that for all $a_0 \in \mathbb{N} \setminus \{0\}$ there is an index s with $a_s = 1$ (from then on, the sequence will consist solely of repetitions of (4, 2, 1)). Now assume that you have a program Z that takes as input a WHILE program Q with one input variable. If Q halts and outputs 0 for all inputs, i.e., $\varphi_Q(y) = 0$ for all $y \in \mathbb{N}$, then Z outputs 0 on input Q. Otherwise, Z outputs 1. Describe how to use this program Z to prove or disprove the Collatz conjecture. **Note:** Do not try to give a program Z. Design a certain program P and show that it suffices to run Z on input P.

3 Reduction Primer

Assume you have a program Z that does the following: Z gets as input a WHILE program P with one input variable as input. If P computes the square function, i.e., $\varphi_P(x) = x^2$ for all $x \in \mathbb{N}$, then Z outputs 1. If P does not compute the square function (because it does not halt on some inputs or because it outputs a different value), then Z outputs 0. Now you are given the following problem: Given a WHILE program P with one input variable and a value $x \in \mathbb{N}$, does P halt on input x? Describe a method to solve this problem using the program Z described above as a black box.

4 Pairing Functions

For completeness, let us recall that $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$, $\mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\}$, and $\mathbb{Q} = \left\{ \frac{a}{b} \,\middle|\, a \in \mathbb{Z}, b \in \mathbb{N} \setminus \{0\} \right\}$.

(a) Give a bijective function $f : \mathbb{Z} \to \mathbb{N}$.
(b) Let $g : \mathbb{N}^2 \to \mathbb{N}$ be given by $g(a, b) = \max(a, b)^2 + \max(a, b) + a - b$. Prove that g is a bijective function.
(c) Give an injective function $h : \mathbb{Q} \to \mathbb{N}$. Is your function also bijective?

5 Cardinalities of Sets

(a) Prove that there is no surjective function $s : \mathbb{N} \to \mathbb{R}$.
(b) Prove the following statement: Let S be an arbitrary set, and let $\mathcal{P}(S)$ be the power set of S. Then there does not exist a surjective function $S \to \mathcal{P}(S)$.
Hint: Assume to the contrary that a surjective function $f : S \to \mathcal{P}(S)$ exists. Consider the set $M = \{x \in S \mid x \notin f(x)\}$. **Remark:** The non-existence of a surjective function $A \to B$ shows that B contains "more" elements than A. For the special case that A is a finite set, this is equivalent to $|B| \geq |A| + 1$. If S is an arbitrary finite set, then $|\mathcal{P}(S)| = 2^{|S|} > |S|$. Simply counting the number of elements, however, fails for infinite sets. In this case, the existence of injective and surjective functions between sets is used to compare their cardinalities.

Solutions for Exercise Sheet 1

(1)
(a) Multiplication can be done as described in Section 3.4. We can do exponentiation as follows (here, y is a new variable):

```
y := xk
xi := 1
while y! = 0 do
    xi := xi · xj
    y := y − 1
```

(b) Here, y and z are new variables.

```
y := 1 − xi (strictly speaking, this is subtracting a variable from a constant; rephraseable as a WHILE program)
z := 1 − y (if xi ¿= 1, then y = 0, z = 1; if xi = 0, then y = 1, z = 0)
while z! = 0 do
    P1
    z := 0
while y! = 0 do
    P2
y := 0
```

(c) There are many ways to do this. One is to find the largest number a such that $a \cdot x_k \leq x_j$. Here, y, z are new variables.

```
xi := 0
z := 1
while z != 0 do
    y := xi * xk - xj (this is two arithmetic operations in one; rephraseable)
    if y = 0 then
        xi := xi + 1
    else (in this case, xi * xk ¿ xj for the first time;)
        xi := xi - 1 (and we have to decrease xi by 1;)
        z := 0 (and we set the flag to leave the loop)
```

(d)

```
y := xj div xk
xi := xj - y * xk
```

(2) We use the following program P that takes x_1 as input and uses it also as the output (caution: the program uses a few statements that do not exist in pure WHILE):

```
z := 1
while z != 0 do
    y := x1 mod 2 (modulo with a constant)
    if y = 0 then
        x1 := x1 div 2
    else
        x1 := 3 * x1 + 1
    z := x1 - 1 (only if 1 is reached, we get z = 0 and leave the loop)
x1 := 0
```

Now we simply use Z to test P. If Z outputs 0, then the Collatz conjecture is true. Otherwise, the Collatz conjecture is false.

(3) Given P and x, we create the following program Q, which takes y as input:

```
run P on input x
compute y^2
```

If P halts on x, then Q computes the square function. If P does not halt on x, then also Q does not halt. In particular, this means that Q does not compute the square function. We can use Z to check if Q computes the square function, which gives us the desired output.

(4)

(a) Let f be given by $x \mapsto f(x) = \begin{cases} 2x & \text{if } x \geq 0, \\ -2x - 1 & \text{if } x < 0. \end{cases}$ We have to prove that f is surjective and injective. Let $y \in \mathbb{N}$. If y is even, then $f(y/2) = y$ since $y/2 \in \mathbb{Z}$ and $y/2 \geq 0$. If y is odd, then $f(-\frac{y+1}{2}) = y$ since $-\frac{y+1}{2} \in \mathbb{Z}$ and $-\frac{y+1}{2} < 0$ (because $y > 0$). This shows that f is surjective. Now consider $x, x' \in \mathbb{Z}$ with $f(x) = f(x')$. If x and x' have different sign, then $f(x)$ and $f(x')$ have different parity. Thus, $f(x) = f(x')$ implies that x and x' have the same sign. If both are non-negative, then $2x = 2x'$ implies $x = x'$. If both are negative, then $-2x - 1 = -2x' - 1$ implies $x = x'$. Thus, f is injective. Since f is both injective and surjective, the function f is bijective. Sometimes, the question arises if there is a closed-formula bijection between $\mathbb{N}$ and $\mathbb{Z}$. Here is a bijective function $g : \mathbb{N} \to \mathbb{Z}$ with this property: $g(n) = \frac{n}{2} \cdot \left(2 \cdot \left\lceil \frac{n+1}{2} \right\rceil - n - 1\right) - \frac{n+1}{2} \cdot \left(2 \cdot \left\lceil \frac{n}{2} \right\rceil - n\right)$.

(b) We have to prove that g is both surjective and injective. Two proofs are provided in the original solutions; we include the main steps here.

Proof 1 (surjectivity). Let $y \in \mathbb{N}$ be arbitrary. Set $h = \lfloor \sqrt{y} \rfloor$. If $y \geq h^2 + h$, let $a = h$ and choose $b \in \{0, 1, \ldots, h\}$ such that $\max(a, b)^2 + \max(a, b) + a - b = h^2 + 2h - b = y$. If $y < h^2 + h$, put $b = h$ and choose $a \in \{0, \ldots, h - 1\}$ such that $\max(a, b)^2 + \max(a, b) + a - b = h^2 + a = y$. This shows surjectivity.

Injectivity. Suppose $g(a, b) = g(a', b')$. Set $c = \max\{a, b\}$ and $c' = \max\{a', b'\}$. If $c \neq c'$ a contradiction arises from size comparisons. If $c = c'$ use the parity and small-case analysis to deduce $a = a'$ and $b = b'$.

(c) Let $c = \frac{a}{b} \in \mathbb{Q}$ with $\gcd(a, b) = 1$, $a \in \mathbb{Z}$, $b \in \mathbb{N} \setminus \{0\}$. Map $h(c) = g(f(a), b)$. By injectivity of f and g, also h is injective. The mapping h is not surjective for two reasons: - $b \neq 0$ (this can be fixed easily), and - images of fractions where numerator and denominator are not coprime are missing (more difficult to fix). **Remark:** There exist bijective mappings between $\mathbb{N}$ and $\mathbb{Q}$ (Cantor–Schröder–Bernstein theorem, etc.).

(5)
(a) Let $f : \mathbb{N} \to [0, 1)$ be arbitrary. For any $x \in \mathbb{N}$, consider the decimal representation $f(x) = \sum_{i=1}^{\infty} 10^{-i} a_{x,i}$ for digits $a_{x,i} \in \{0, 1, \ldots, 9\}$. Define $y = \sum_{i=1}^{\infty} 10^{-i} b_i$, $\quad b_i = 1 - a_{i,i} \in \{0, 1\}$. Then $y \in [0, 1)$ and by construction y differs from $f(x)$ in the x-th decimal place for every $x \in \mathbb{N}$. Hence f is not surjective. (Care: using binary representation would cause ambiguity for dyadic rationals.)
(b) Let S be arbitrary, and assume a surjective $f : S \to \mathcal{P}(S)$ exists. Consider $M = \{x \in S \mid x \notin f(x)\}$. By surjectivity there exists $y \in S$ with $f(y) = M$. Contradiction: $y \in M \iff y \notin f(y) = M$.

Exercise Sheet 2

1 Reductions

Let SomeSquare $= \{g \in G \mid \exists n \in \mathbb{N} : \varphi_g(n) = n^2\}$.
(a) Show $H_0 \leq_m$ SomeSquare.
(b) Show SomeSquare $\in$ RE.
(c) Is SomeSquare $\in$ REC?

2 Recursive Enumerability

Show that the following statements are equivalent for all sets $L \subseteq \mathbb{N}$:
(i) There exists a WHILE program P with $\text{dom}(\varphi_P) = L$.
(ii) There exists a WHILE program P with $\text{im}(\varphi_P) = L$.
(iii) Either $L = \varnothing$ or there exists a WHILE program P that always terminates and satisfies $\text{im}(\varphi_P) = L$.
(iv) $L \in$ RE.
(v) $L \leq_m H_0$. **Hint:** At some point, showing $L \leq_m H \leq_m H_0$ and exploiting transitivity might be easier than directly showing $L \leq_m H_0$. Proving $H \leq_m H_0$ is a useful exercise.

3 More on Reductions and Computable Functions

Prove the following statements.
(a) Let $f, g : \mathbb{N} \to \mathbb{N}$ be two WHILE-computable functions. Then also the function $h = f \circ g$ given by $h(x) = \begin{cases} f(g(x)) & \text{if } x \in \text{dom}(g) \text{ and } g(x) \in \text{dom}(f), \\ \text{undefined} & \text{otherwise} \end{cases}$ is WHILE-computable.
(b) If $A \leq_m B$ and $B \leq_m C$ for decision problems $A, B, C \subseteq \mathbb{N}$, then $A \leq_m C$.
(c) If $A \leq_m B$ for decision problems $A, B \subseteq \mathbb{N}$, then $\bar{A} \leq_m \bar{B}$. (trivial restatement)

4 Functions

Note: In this exercise, we deal with functions in the ordinary sense, i.e., all functions are total.
(a) Prove that the following two statements are equivalent for all sets A and B:
(i) There is an injective function $A \to B$.
(ii) There is a surjective function $B \to A$.
(There is a technical difficulty here; you may refer to the axiom of choice.)
(b) Show that the following three statements are equivalent for all finite sets A and all functions $f : A \to A$:
(iv) f is bijective. (v) f is injective. (vi) f is surjective.

Solutions for Exercise Sheet 2

(1)

(a) Consider the function $f : \mathbb{N} \to \mathbb{N}$ given by $g \mapsto f(g) = \begin{cases} g & \text{if } g \notin G, \\ \tilde{g} & \text{if } g \in G, \end{cases}$ where $\tilde{g}$ is the Gödel number of the program that on input x (1) simulates göd$^{-1}(g)$ on input g and (2) outputs x^2. Then f is total and computable and yields a many-one reduction from H_0 to SomeSquare.

(b) To show SomeSquare $\in$ RE, consider the WHILE program P:

```
input: z = ¡g,x¿ in N
if g not in G then
    loop forever
else
    simulate göd^−1(g) on input x using U
    if U terminates with output x^2 then
        output g
    else
        loop forever
```

This program enumerates those g for which some input x yields x^2; hence SomeSquare is recursively enumerable.

(c) If SomeSquare were recursive, it would contradict the fact that $H_0 \notin$ REC (via the reduction from (a)). Hence SomeSquare $\notin$ REC.

(2)
We sketch the equivalences: **(iv) $\Rightarrow$ (i):** If $L \in$ RE there exists a WHILE-computable partial function f with $\text{dom}(f) = L$. Modify output behavior (e.g., loop instead of outputting 0) to get a WHILE program with domain L.
(i) $\Rightarrow$ (ii): Given P with $\text{dom}(\varphi_P) = L$, modify P so that when it halts on x it outputs x. Then the image equals L.
(ii) $\Rightarrow$ (iii): If $L = \varnothing$ trivial. Otherwise pick $y_0 \in L$. Construct Q that on input $\langle x, t \rangle$ simulates P on x for t steps; if P halts output its result, otherwise output y_0. Then Q terminates on all inputs and $\text{im}(\varphi_Q) = L$.

(iii) ⇒ (iv): If such always-terminating P exists with $\text{im}(\varphi_P) = L$ then enumerate the image by running P on all inputs.
(i) ⇒ (v): Sketch: pick Gödel number g with $\text{dom}(\varphi_g) = L$. Then $x \mapsto \langle g, x\rangle$ reduces L to H. Show $H \leq_m H_0$ separately.
(v) ⇒ (iv): Because $H_0 \in$ RE and RE closed under many-one reductions, $L \in$ RE.
(3)
(a) Compose the WHILE programs for g and f, clearing temporary variables between them so partiality is handled correctly; this yields a WHILE program for $h = f \circ g$.
(b) If f reduces A to B and g reduces B to C, then $g \circ f$ reduces A to C. Use computability of composition and correctness of reductions.
(c) Trivial restatement: if f is a reduction from A to B it is such by definition.
(4)
(a) "⇒": If there is an injective $f : A \to B$, fix $a_0 \in A$ and define $g : B \to A$ by $g(b) = \begin{cases} a & \text{if } f(a) = b, \\ a_0 & \text{if } b \notin \text{im } f. \end{cases}$ Then g is surjective.
"⇐": If $g : B \to A$ is surjective, for every $a \in A$ let $M_a = \{b \in B \mid g(b) = a\}$. Choose $m_a \in M_a$ (requires axiom of choice). Then $f(a) = m_a$ is injective.
(b) For finite A, injectivity implies $|\text{im } f| = |A|$ hence $\text{im } f = A$ so surjective; similarly surjective implies injective. Thus all three properties are equivalent.

Exercise Sheet 3

1 Decidability

Check for the following nine decision problems which of the following properties hold: (i) They are index sets. (ii) They are decidable. (iii) They are recursively enumerable. (iv) Their complements are recursively enumerable.
(a) $L_1 = \{i \in G \mid 42 \in \text{dom}(\varphi_i)\}$.
(b) $L_2 = \{i \in G \mid \text{göd}^{-1}(i)$ terminates on input 42 after at most 2500 steps$\}$.
(c) $L_3 = \{i \in G \mid \text{dom}(\varphi_i) \in \text{RE}\}$.
(d) $L_4 = \{i \in G \mid \text{göd}^{-1}(i)$ contains at most three WHILE loops$\}$.
(e) $L_5 = \{i \in G \mid \text{dom}(\varphi_i)$ contains infinitely many elements$\}$.
(f) $L_6 = \{i \in G \mid \text{dom}(\varphi_i)$ is a finite set$\}$.
(g) $L_7 = L_1 \cup \{i \mid i \leq 10^{10^{1000}}\}$.
(h) $L_8 = L_1 \cap \{i \mid i \leq 10^{10^{1000}}\}$.
(i) $L_9 = L_1 \cap \{i \in G \mid \text{göd}^{-1}(i)$ contains at most 1000 WHILE loops$\}$.

2 Prove or Disprove

Prove or disprove the following statements. For the first three, let $h : \mathbb{N} \to \mathbb{N}$ be a total function and $A \subseteq \mathbb{N}$; define $h(A) = \{h(x) \mid x \in A\}$.
((a) For every total, WHILE-computable function h and every set A, if $A \in$ REC then $h(A) \in$ REC.
(b) For every total, WHILE-computable function h and every set A, if $h(A) \in$ REC then $A \in$ REC.
(c) For every total, WHILE-computable function h and every set A, if $A \in$ RE then $h(A) \in$ RE.
(d) For every $A \subseteq \mathbb{N}$, if $A \leq \bar{A}$, then $A \in$ REC or $A \notin$ RE.
(e) For every $A \subseteq \mathbb{N}$, if $A \leq \bar{A}$, then $A \in$ REC.

3 Index Sets Revisited

Consider the following statement: If $A \cap B = \emptyset$, then there exists a WHILE-computable function $f : \mathbb{N} \to \mathbb{N}$ such that $f(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \in B, \\ \in \{0,1\} \text{ or undefined} & \text{if } x \notin A \cup B. \end{cases}$ Which of the following cases holds:
(i) The statement holds for all non-trivial index sets $A, B \subseteq \mathbb{N}$.
(ii) There exist non-trivial index sets $A, B \subseteq \mathbb{N}$ such that it holds.
(iii) The statement is false for all non-trivial index sets $A, B \subseteq \mathbb{N}$.

4 Asymptotic Growth

(a) Sort the following functions according to their asymptotic growth (base 2 for log): (i) $n \log n$ (ii) n^2 (iii) $n!$ (iv) $(n+1)!$ (v) 2^n (vi) n^n (vii) $n^{\log n}$ (viii) $2^{(\log n)^2}$ (ix) $3^{3^{n^2}}$ (x) $2^{2^{\log\log n}}$ (xi) $(\log n)^{\log n}$ (xii) n
(c) Prove or disprove: if $f, g : \mathbb{N} \to \mathbb{N}$ are monotone and $f(n) = g(5n)$, then $f \in \Theta(g)$.
(d) Prove or disprove: for all $f, g : \mathbb{N} \to \mathbb{N}$, if $f \notin O(g)$, then $g \in O(f)$.

Solutions for Exercise Sheet 3

(1)
(a) L_1 is a non-trivial index set. If $i \in L_1$ and $\varphi_i = \varphi_j$, then $42 \in \text{dom}\,\varphi_j$, hence $j \in L_1$. By Rice's theorem, L_1 is undecidable ($L_1 \notin$ REC). A WHILE-computable $f(i) = \varphi_i(42)$ has $\text{dom}(f) = L_1$, hence $L_1 \in$ RE but not co-RE.
(b) L_2 can be decided by simulating $\text{göd}^{-1}(i)$ for 2500 steps, so $L_2 \in$ REC. It is not an index set, since two equivalent programs may differ in step count.
(c) Every WHILE-computable function has recursively enumerable domain. Hence $L_3 = G \in$ REC, a trivial index set.
(d) Checkable by parsing the Gödel number, so $L_4 \in$ REC, not an index set.
(e) L_5 is a non-trivial index set. By Rice's theorem, $L_5 \notin$ REC. Using reductions from H_0, one shows $L_5 \notin$ RE and $L_5 \notin$ co-RE.
(f) $L_6 = G \setminus L_5$ is also non-trivial; thus $L_6 \notin$ REC, RE, co-RE.
(g) L_7 differs from L_1 only finitely; all other properties coincide.
(h) L_8 is finite, hence in REC, RE, co-RE, but not an index set.
(i) L_9 is recursively enumerable but not recursive (shown via $L_1 \leq L_9$). It is not an index set.
(2)
(a) False. Let $A = \mathbb{N}$ and h total computable with $\text{im}(h) = H$ (the halting problem). Then $A \in$ REC but $h(A) = H \notin$ REC.
(b) False. Let $h(x) = 0$, $A = H$. Then $A \notin$ REC but $h(A) = \{0\} \in$ REC.
(c) True. If $A \in$ RE, then there exists total computable f with $\text{im}(f) = A$. Since h is total computable, $h \circ f$ is total computable and $\text{im}(h \circ f) = h(A) \in$ RE.
(d) True. If $A \leq \bar{A}$, then $A \in \text{RE} \Rightarrow \bar{A} \in \text{co-RE} \Rightarrow A \in$ REC.
(e) False. Counterexample: Let $A = \{2x \mid x \in H\} \cup \{2x+1 \mid x \notin H\}$, where H is the halting problem. Then $A \notin$ REC but $A \leq \bar{A}$ via $f(n) = n \pm 1$.
(3)
Version (ii) is true. Versions (i) and (iii) are false. If A is a non-trivial index set, $B = G \setminus A$ is also non-trivial. If f as described existed for all such pairs, A would be decidable — contradiction to Rice's theorem. However, there exist special A, B satisfying the property (constructed in the solution).
(4)
Ordering (increasing growth): (1) $2^{2^{\log\log n}}$, n (2) $n \log n$ (3) n^2 (4) $(\log n)^{\log n}$ (5) $n^{\log n}$, $2^{(\log n)^2}$ (6) 2^n (7) $n!$ (8) $(n+1)!$ (9) n^n (10) $3^{3^{n^2}}$
(c) False. Example: $g(n) = 2^n$, then $f(n) = 3^{2n} = \omega(g(n))$.
(d) False. Example: $f(n) = n \bmod 2$, $g(n) = (n+1) \bmod 2$.

Exercise Sheet 4

1 Counting on Turing Machines

Consider a Turing machine that, on input n in binary, repeatedly subtracts 1 until 0. Show that it needs only $O(n)$ steps (not $O(n \log n)$).

2 Nondeterminism

A graph $G = (V, E)$ is 3-colorable if there exists $\pi : V \to \{\text{red,green,blue}\}$ with $\pi(u) \neq \pi(v)$ for all $\{u, v\} \in E$. Let 3-Coloring $= \{G \mid G$ is 3-colorable$\}$. Show that 3-Coloring $\in \text{NTime}(N^{10})$, where N is the bit length of the encoding of G.

3 Palindromes – Revisited

We aim to show that Palindrome $\notin \text{DTime}_1(o(n^2))$. A sequence of subparts (a)–(i) define the crossing-sequence argument proving the quadratic lower bound for one-tape Turing machines.

4 Riemann Hypothesis and Computability

Prove or disprove: the function $r(n) = \begin{cases} 1, & \text{if RH is true,} \\ 0, & \text{if RH is false} \end{cases}$ is computable.

Solutions for Exercise Sheet 4

(1)
The running time equals the number of bit changes during counting. The i-th bit flips every 2^i iterations. If 2^k is the smallest power $\geq n$, then $n \leq 2^k \leq 2n$. Hence the total number of flips is bounded by $2^k \sum_{i=0}^{k} 2^{-i} i \leq 4n = O(n)$.
(2)
For each vertex, guess a color nondeterministically ($O(n)$ time). Verify edges ($O(n^2)$ time). Thus the total is polynomially bounded, e.g. $O(N^{10})$.
(3)
(a) If $x = yz$, $x' = y'z'$, $i = |y|$, $i' = |y'|$, and $CS(x, i) = CS(x', i')$, then substituting z' for z does not change acceptance, so $yz' \in L(M)$.
(b) Each step crossing i or $i+1$ contributes to $|CS(x, i)|$; summing gives total time: $\text{Time}_M(x) = \sum_i |CS(x, i)|$.
(c) For inputs $q_x = x1^m x^{\text{rev}}$ with $|x| = m$, different x yield distinct crossing sequences in the middle region.
(d) Averaging over all $x \in \{0,1\}^m$ gives $2^{-m} \sum_x \text{Time}_M(q_x) \geq \sum_{i=m}^{2m} \ell_i$, with $\ell_i = 2^{-m} \sum_x |CS(q_x, i)|$.
(e) At least half of the 2^m strings satisfy $|CS(q_x, i)| \leq 2\ell_i$.
(f) There are at most s^r crossing sequences of length r, so sequences of length $\leq 2\ell_i$ are bounded by $s^{2\ell_i + 1}$.
(g) Since distinct strings have distinct crossing sequences, $2^{m-1} \leq s^{2\ell_i+1}$.
(h) Rearranging: $\ell_i \geq \frac{m-1}{2\log s} - \frac{1}{2} \geq \frac{1}{4\log_2 s} m$ for large m.
(i) Averaging implies $2^{-m} \sum_x \text{Time}_M(q_x) \geq \frac{1}{4\log_2 s} m^2$, so some x has $\text{Time}_M(q_x) \geq cn^2$ with $n = 3m$. Hence one-tape Turing machines need $\Omega(n^2)$ time for Palindrome.
(4)
If the Riemann hypothesis is true, r is the constant-1 function (computable). If false, it is constant-0 (computable). Thus r is computable regardless of truth value.

Exercise Sheet 5

1 Palindromes – Last Time

Modify the proof that Palindrome $\notin \text{DTime}_1(o(n^2))$ to show that Palindrome $\notin \text{NTime}_1(o(n^2))$.

2 Nondeterminism

Prove or disprove that the following holds for all time-constructible functions t and all languages L: $L \in \text{NTime}_1(t) \Rightarrow \bar{L} \in \text{NTime}_1(t)$ **Hint:** Palindrome.

3 Polynomial-Time Many-One Reductions

(a) Prove 3-Coloring $\leq_P$ 3SAT.
(b) What can you conclude from this reduction, given that 3SAT is NP-complete?

4 Satisfiability

Prove that kSAT is NP-complete for all $k \geq 4$. You can use the fact that 3SAT is NP-complete. kSAT is defined in the same way as 3SAT, except that clauses consist of k literals.

5 Self-Reducibility

It might look weird that we restrict ourselves to "yes/no" variants of problems that are more naturally stated as search or optimization problems. Assume you have a polynomial-time algorithm `MagicClique` that decides Clique $= \{(G, k) \mid G$ contains a clique of size at least $k\}$. Give a polynomial-time algorithm that finds in polynomial time the largest clique of $G = (V, E)$ using `MagicClique`.

Solutions for Exercise Sheet 5

(1)
The proof is identical to the deterministic case. If an NTM accepts q_x and q_y for $|x| = |y| = m$ and $x \neq y$, and the same crossing sequence appears between m and $2m$, then the NTM also accepts $x1^n y^{\text{rev}} \notin$ Palindrome.
(2)
The statement is false. We have $\overline{\text{Palindrome}} \in \text{NTime}_1(n \log n)$, but Palindrome $\notin \text{NTime}_1(o(n^2))$.
(3a)
Let $G = (V, E)$ be an undirected graph. We map G to a formula Φ in 3CNF such that $G \in$ 3-Coloring iff $\Phi \in$ 3SAT. Variables: $x_{v,c}$ for $c \in \{\text{red, green, blue}\}$ and $v \in V$. Interpretation: $x_{v,c} = 1$ if and only if vertex v has color c. Clauses: (i) $x_{v,\text{red}} \vee x_{v,\text{green}} \vee x_{v,\text{blue}}$ for all $v \in V$. (ii) $\neg x_{u,c} \vee \neg x_{v,c}$ for all $\{u, v\} \in E$ and $c \in \{\text{red, green, blue}\}$. (Clauses with only two literals can be extended with dummy variables if needed.) It follows that Φ is satisfiable iff G is 3-colorable. The mapping $G \mapsto \Phi$ is polynomial-time computable, so 3-Coloring $\leq_P$ 3SAT.
(3b)
Since 3SAT is NP-complete, we conclude that 3-Coloring $\in$ NP.
(4)
We reduce kSAT to $(k+1)$SAT. Given Φ in kCNF with clauses $C_1, \ldots, C_m$, define $C_i' = C_i \vee y$, $C_i'' = C_i \vee \neg y$. Let $\Phi' = \bigwedge_i (C_i' \wedge C_i'')$. Clearly, $\Phi \in k\text{SAT} \Leftrightarrow \Phi' \in (k+1)\text{SAT}$, and the transformation is polynomial-time computable. Since 3SAT is NP-hard, all kSAT for $k \geq 3$ are NP-hard and also in NP, hence NP-complete.
(5) Algorithm `FindCliqueSize`
Input: undirected graph $G = (V, E)$
$n = |V|$
For $k = n$ down to 0:
If `MagicClique`(G, k) returns "yes", return k.
This finds the largest k for which G contains a clique of size k.
Algorithm `FindClique`
Input: graph $G = (V, E)$, $k_{\max}$ the largest clique size.
$U \leftarrow V$
For each $v_i \in V$:
If `MagicClique`$(G_{U \setminus \{v_i\}}, k_{\max})$ returns "yes", then $U \leftarrow U \setminus \{v_i\}$.
Return U
The algorithm outputs a clique C of size $k_{\max}$, as proven by induction in the solution text.

Exercise Sheet 6

1 RE-Completeness

Prove that the special halting problem H_0 is RE-complete.

2 Graph Reachability – Revisited

Show that the following can be solved by an NTM with $O(\log n)$ space: Input: directed graph $G = (V, E)$, vertices $s, t \in V$, and a number $\ell \in \mathbb{N}$. The NTM should: - Accept if there is no path from s to t and ℓ equals the number of vertices reachable from s. - Reject if there is a path from s to t and ℓ equals the number of vertices reachable from s. - Otherwise, behave arbitrarily.

3 NP-Completeness

(a) Show that IndependentSet $= \{(G, k) \mid G$ has an independent set of size $k\}$ is NP-complete.
(b) Show that VertexCover $= \{(G, k) \mid G$ has a vertex cover of size $k\}$ is NP-complete.

4 NP and co-NP

Let $U \in P$ and $L \in NP$ with $L \subsetneq U$. (a) Prove that $U \setminus L \in$ co-NP.
(b) Assume L is NP-complete. Prove that $U \setminus L$ is co-NP-complete.

5 co-NP-Completeness

Prove that NonClique $= \{(G, k) \mid G$ does not contain a clique of size $k\}$ is co-NP-complete.

Solutions for Exercise Sheet 6

(1)
$H_0 \in$ RE. Let $A \in$ RE. Then there exists a WHILE program P such that $\text{dom}(\varphi_P) = A$. Construct a Gödel number g_x encoding program Q:
input: n
run P on input x
output 0
Then $x \in A \Leftrightarrow g_x \in H_0$. Thus $A \leq H_0$, proving H_0 is RE-complete.
(2)
Vertices are numbered $1, \ldots, n$; assume $s = 1, t = n$. The NTM executes: 1: $c := 0$ 2: For $x = 1, \ldots, n-1$: 3: Nondeterministically guess if x is reachable. 4: If guessed and $c < \ell$, increment c. 5: Nondeterministically guess a path from s to x. If not found, reject. 6: If $c = \ell$, accept; else reject.
If ℓ equals the number of vertices reachable from s, the behavior matches the problem's definition.
(3a)
IndependentSet $\in$ NP (obvious verifier). We reduce Clique $\leq_P$ IndependentSet. Given $G = (V, E)$, define $G' = (V, E')$ where $E' = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$. Then $(G, k) \in$ Clique $\Leftrightarrow$ $(G', k) \in$ IndependentSet, so IndependentSet is NP-complete.
(3b)
VertexCover $\in$ NP. Use complementarity: U is an independent set $\Leftrightarrow V \setminus U$ is a vertex cover. Hence $(G, k) \mapsto (G, |V| - k)$ is a polynomial reduction from IndependentSet to VertexCover. Therefore, VertexCover is NP-complete.
(4a)
Since $U \in P$ and $L \in NP$, we can verify membership in $U \setminus L$ in polynomial time using L's certificate. Hence $U \setminus L \in$ co-NP.
(4b)
If L is NP-complete, construct f such that $x \in 3\text{SAT} \Leftrightarrow f(x) \in L$. Modify to $\tilde{f}(x) = \begin{cases} f(x) & f(x) \in U, \\ y & f(x) \notin U, \end{cases}$ for some fixed $y \in U \setminus L$. Then $\overline{3\text{SAT}} \leq_P U \setminus L$, proving co-NP-completeness.
(5)
Let U be all proper encodings of (G, k) and $L =$ Clique. Then $U \setminus L =$ NonClique, which is co-NP-complete.

Exercise Sheet 7

1 Certificates for RE and the Arithmetical Hierarchy

(a) Prove that the following two statements are equivalent for all $L \subseteq \mathbb{N}$: (i) $L \in \text{RE}$. (ii) There exists a set $R \in \text{REC}$ with: - For all $x \in L$, there exists a $c \in \mathbb{N}$ with $\langle x, c\rangle \in R$. - For all $x \notin L$ and $c \in \mathbb{N}$, we have $\langle x, c\rangle \notin R$.

(b)
Let $\text{Total} = \{g \in G \mid \text{dom}(\varphi_g) = \mathbb{N}\}$ be the set of Gödel numbers representing programs computing total functions. Prove that there exists a set $R_{\text{Total}} \in \text{REC}$ with: - For all $g \in \text{Total}$, for all $a \in \mathbb{N}$ there exists $b \in \mathbb{N}$ with $\langle g, \langle a, b\rangle\rangle \in R_{\text{Total}}$. - For all $g \notin \text{Total}$, there exists an $a \in \mathbb{N}$ such that for all $b \in \mathbb{N}$, $\langle g, \langle a, b\rangle\rangle \notin R_{\text{Total}}$.

(c) Define $\Pi_2^0 \subseteq P(\mathbb{N})$ as the set of decision problems L for which there exists $R_L \in \text{REC}$ such that: - For all $x \in L$: for all $a \in \mathbb{N}$ there exists $b \in \mathbb{N}$ with $\langle x, \langle a, b\rangle\rangle \in R_L$. - For all $x \notin L$: there exists an $a \in \mathbb{N}$ such that for all $b \in \mathbb{N}$, $\langle x, \langle a, b\rangle\rangle \notin R_L$. Prove: If $L \in \Pi_2^0$, then $L \leq \text{Total}$. **Remark:** Part (b) shows $\text{Total} \in \Pi_2^0$. Part (c) shows Total is Π_2^0-hard, hence Π_2^0-complete.

2 NP-Completeness 1

Prove that the following problem is NP-complete: $\text{01-ILP} = \{(A, b) \mid A \in \mathbb{Z}^{m\times n},\ b \in \mathbb{Z}^m,\ \exists x \in \{0,1\}^n : Ax \geq b\}$. Here $(Ax)_j \geq b_j$ for all $j \in \{1, \dots, m\}$.

3 NP-Completeness 2

Given an undirected graph $G = (V, E)$, define the density of the subgraph induced by $U \subseteq V$ as $d(U) = \frac{|\{e \in E \mid e \subseteq U\}|}{\binom{|U|}{2}}$. Prove that $\text{DenseSubgraph} = \{(G, \alpha, k) \mid \exists U \subseteq V, |U| \geq k,\ d(U) \geq \alpha\}$ is NP-complete.

4 Translation

Prove that if $L = P$, then PSPACE = EXP. **Hint:** For a language A, consider $\tilde{A} = \{x\#1^{f(|x|)} \mid x \in A\}$, with a cleverly chosen f.

Solutions for Exercise Sheet 7

(1a)
$\Rightarrow$: If $L \in \text{RE}$, there exists $f \in R$ with $\text{im}(f) = L$. Let $R = \{\langle x, c\rangle \mid f(c) = x\}$. Then $R \in \text{REC}$ and has the desired properties. Alternatively, let P be a WHILE program deciding L, and define $R = \{\langle x, c\rangle \mid P(x) \text{ halts in at most } c \text{ steps}\}$. Then $R \in \text{REC}$ and satisfies the condition. $\Leftarrow$: Given R as in the statement, define $f(\langle x, c\rangle) = \begin{cases} x, & \text{if } \langle x, c\rangle \in R, \\ \text{undefined}, & \text{otherwise.} \end{cases}$ Then $\text{im}(f) = L$, hence $L \in \text{RE}$.

(1b)
Let $R_{\text{Total}} = \{\langle g, \langle a, b\rangle\rangle \mid g \in G, \text{ and } \text{göd}^{-1}(g) \text{ halts on input } a \text{ after } \leq b \text{ steps}\}$. If $g \in \text{Total}$, for every a there exists b with $\langle g, \langle a, b\rangle\rangle \in R_{\text{Total}}$. If $g \notin \text{Total}$, there exists some a with $\langle g, \langle a, b\rangle\rangle \notin R_{\text{Total}}$ for all b.

(1c)
Let $L \in \Pi_2^0$ with R_L as above. Construct $f(x) = g_x$, where g_x is the Gödel number of:

```
input: a
for b = 0, 1, 2, ... do
    if ⟨x, ⟨a, b⟩⟩ ∈ R_L then output 1
```

Then $x \in L \Leftrightarrow g_x \in \text{Total}$, so $L \leq \text{Total}$.

(2)
A certificate for 01-ILP is $x \in \{0,1\}^n$ satisfying $Ax \geq b$. Verification is polynomial-time, hence 01-ILP $\in$ NP. To show NP-hardness, reduce 3SAT $\leq_P$ 01-ILP. Given $\Phi = C_1 \wedge \cdots \wedge C_m$ in 3CNF, each $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$, map to inequalities: $\ell_{i,1} + \ell_{i,2} + \ell_{i,3} \geq 1$, replacing $\ell_{i,j}$ by x_k if $\ell_{i,j} = x_k$, and by $1 - x_k$ if $\ell_{i,j} = \neg x_k$. This gives a polynomial reduction $\Phi \mapsto (A, b)$. Hence 01-ILP is NP-complete.

(3)
DenseSubgraph $\in$ NP since we can verify $|U| \geq k$ and $d(U) \geq \alpha$ in polynomial time. Reduction: Clique $\leq_P$ DenseSubgraph by $(G, k) \mapsto (G, 1, k)$. If G has a k-clique, then there exists U with $d(U) = 1$. Conversely, if some U with $|U| \geq k$ has $d(U) = 1$, U is a clique. Thus, DenseSubgraph is NP-complete.

(4)
Assume $L = P$. Then every $L \in \text{EXP}$ satisfies EXP $\subseteq$ PSPACE. Let $L \in \text{EXP}$, so $L \in \text{DTime}(2^{p(n)})$ for some polynomial p. Define $\tilde{L} = \{x\#1^{2^{p(|x|)}} \mid x \in L\}$. Checking $\tilde{L}$ can be done in linear time, so $\tilde{L} \in P = L$. Simulating $\tilde{L}$ requires $O(p(|x|))$ space, so $L \in \text{PSPACE}$.

Exercise Sheet 8

1 NL-Completeness

A directed graph $G = (V, E)$ is strongly connected if every pair $u, v \in V$ has a path from u to v. Let $\text{StrongConn} = \{G \mid G \text{ is strongly connected}\}$. Prove that StrongConn is NL-complete.

2 Sublogarithmic Space

For $m, d \in \mathbb{N}$ with $0 \leq m \leq 2^d - 1$, let $\text{bin}_d(m)$ be the d-digit binary encoding of m. Define $\text{Count} = \{\text{bin}_d(0)\#\text{bin}_d(1)\#\cdots\#\text{bin}_d(2^d - 1)\# \mid d \geq 1\}$. Show that Count $\in \text{DSpace}(\log\log n)$. **Hint:** The input length is $n = (d+1)2^d$, so $O(\log d) = O(\log\log n)$ space is available.

3 A Polynomial Algorithm for 3SAT

Assume $P = NP$. Describe an algorithm A for 3SAT such that: - There exists c where, for all $\Phi \in \text{3SAT}$, A outputs a satisfying assignment in $O(n^c)$ time. - If $\Phi \notin \text{3SAT}$, then A may not terminate.

Solutions for Exercise Sheet 8

(1)
We show StrongConn $\in$ NL and NL-hard. To decide StrongConn: For each pair $u, v \in V$, nondeterministically guess a path from u to v using $O(\log n)$ space. Accept iff paths exist for all pairs. NL-hardness: reduce CONN $\leq_{\log}$ StrongConn. Given (G, s, t), construct $G' = (V, E')$ where $E' = E \cup \{(v, s) \mid v \in V, v \neq s\} \cup \{(t, v) \mid v \in V, v \neq t\}$. $(G, s, t) \in \text{CONN} \Leftrightarrow G' \in \text{StrongConn}$.

(2)
We define maximal binary substrings (MBS) and design an iterative check: 1: Verify format using constant space (regular expression). 2: Check first and last blocks contain only 0s and 1s respectively. 3: For each phase $i = 1, 2, \dots$: - Check all MBS have length $\geq i$. - Check consecutive MBS represent consecutive numbers mod 2^i. If input $\in$ Count, all tests pass and we accept; otherwise, we reject in some phase. Each phase uses $O(\log i)$ space, and reaching phase i implies $n = \Omega(2^i)$, giving total $O(\log\log n)$ space. The extra question: this does *not* imply that $n \mapsto \log\log n$ is space-constructable.

(3)
Enumerate Turing machines $M_1, M_2, \dots$, simulate each for 2^i steps in round i. Since $P = NP$, there exists M_k running in $O(n^c)$ time that outputs a satisfying assignment for Φ. Hence, simulation halts in $O(2^k \cdot n^c) = O(n^c)$ steps when Φ is satisfiable.