§IX. Logic-related Problems (from space & time complexity's perspective)

* 关于 logic formula (representations) 的定义:

Definitions:

1. Proposition letters & Propositional Formula:

$\mathbf{P} = \{p_1, p_2, \ldots\}$ — proposition letters.

The set of formulas of propositional logic:

- every element of $\mathbf{P}$ is a formula
- if φ_1 and φ_2 are formulas, then so are

$$(\neg\varphi_1),\quad (\varphi_1 \vee \varphi_2),\quad (\varphi_1 \wedge \varphi_2),\quad (\varphi_1 \to \varphi_2)$$

2. Assignment:

对任意谓词公式的赋值

- An *assignment* is a function $\theta : \mathbf{P} \to \{T, F\}$.
- we extend θ to formulas by setting

$$\begin{aligned}
\theta(\neg\varphi_1) &= T \text{ iff } \varphi_1 = F \\
\theta(\varphi_1 \vee \varphi_2) &= T \text{ iff } \theta(\varphi_1) = T \text{ or } \theta(\varphi_2) = T \\
\theta(\varphi_1 \wedge \varphi_2) &= T \text{ iff } \theta(\varphi_1) = T \text{ and } \theta(\varphi_2) = T \\
\theta(\varphi_1 \to \varphi_2) &= T \text{ iff } \theta(\varphi_1) = F \text{ or } \theta(\varphi_2) = T
\end{aligned}$$

3. Satisfiability:

- A formula φ is *satisfiable* if there exists an assignment θ such that $\theta(\varphi) = T$.

* 关于 propositional logic formula 的主要问题: Satisfiability Check!

即: 检查给定谓词公式的可满足性.

PROPOSITIONAL SAT

Given: a propositional logic formula φ;

Return: Yes if φ is satisfiable, and No otherwise.

关于 Prop-SAT: 主结论:

Theorem:

PROPOSITIONAL-SAT is in NPTime.

证明:

The following non-deterministic algorithm recognizes the satisfiable formulas

```
begin NdPropSatTest(φ)
    guess an assignment θ for the proposition letters in φ
    if θ(φ) = ⊤ then return Yes
    return No
```

if it can guess out a satisfying assignment then it's running time is polynomial.

* 引入关于文字 (literal) 和子句 (clause) 的概念:

Definitions:

1. literal:

A literal is an expression p or $\neg p$, where p is an proposition letter

2. clause:

A clause is an expression $\ell_1 \vee \cdots \vee \ell_k$, where the ℓ_i are literals. (The *empty* disjunction is written $\bot$.)

注: If ℓ is a literal, denote the opposite literal by $\bar{\ell}$.

3. Satisfiable for a set of clauses:

A set of clauses is *satisfiable* if there exists an assignment θ such that $\theta(\gamma) = T$ for all $\gamma \in \Gamma$.

故引出了下面的问题: (set of clauses 的 satisfiability 问题)

SAT

Given: a set of clauses Γ;

Return: Yes if Γ is satisfiable, and No otherwise.

k-SAT

Given: a set of clauses Γ, each with at most k literals;

Return: Yes if Γ is satisfiable, and No otherwise.

对 k-SAT 问题的解决：DPLL算法.

The Davis-Putnam (-Logemann-Loveland) algorithm:

```
begin resolve(Γ, ℓ)
    for each γ ∈ Γ
        if γ contains ℓ, remove γ from Γ
        if γ contains ℓ̄, remove ℓ̄ from γ

begin DPLL(Γ)
    if Γ is empty then return Yes
    if Γ contains the empty clause then return No
    while Γ contains any unit clause ℓ
        remove ℓ from Γ
        Γ = resolve(Γ, ℓ)
        if Γ is empty then return Yes
        if Γ contains the empty clause then return No
    let ℓ be the first literal of the first clause of Γ
    if DPLL(Γ ∪ {ℓ}) then return Yes
    if DPLL(Γ ∪ {ℓ̄}) then return Yes
    return No
```

* 下面关注两种特殊的SAT问题：Horn-SAT 和 Krom-SAT (2-SAT).

1. Horn-SAT:

首先定义 Horn Clause:

对每个 clause，其中最多一个 literal 为 negative，即：总可写成 Implication 的形式.

则称这样的 clause 为 Horn Clause.

形式化定义关于 Horn-clause 的 SAT 问题：

Horn-SAT
Given: a set of Horn-clauses Γ;
Return: Yes if Γ is satisfiable, and No otherwise.

并且有以下结论：

Theorem.
Horn-SAT is in PTime.

证明：可使用修改版的 DPLL 在多项式时间内判定.

The following modification of DPLL decides Horn-SAT:

```
begin Horn-DPLL(Γ)
    if Γ contains the empty clause then return No
    while Γ contains any unit clause ℓ
        remove ℓ from Γ
        Γ = resolve(Γ, ℓ)
        if Γ contains the empty clause then return No
    return Yes
end Horn-DPLL
```

2. Krom-SAT

首先定义 Krom clause:

Krom clause 就是 2-clause：contains at most 2 literals.

注：non-unit Krom clause may be regarded as implications. (形如 $\neg p \vee q$).

形式化定义关于 Krom-clause 的 SAT 问题：

Krom-SAT (= 2-SAT)
Given: a set of Krom-clauses Γ;
Return: Yes if Γ is satisfiable, and No otherwise.

KROM-UNSAT
Given: A set Γ of Krom clauses.
Return: Yes if Γ is unsatisfiable, and No otherwise.

并且有以下结论：

Theorem.

KROM-UNSAT is in NLogSpace.

证明：不妨假设给定的 Γ：a set of Krom clauses 中无 unit clause，且 $\perp \notin \Gamma$，$\Gamma \neq \emptyset$.
故所有 Γ 中的 clause 均可表为 implication，即：$l \to m$ 的形式.

并且称：若 $\exists\ l_0, l_1, \cdots, l_k$, s.t. $l_0 \to l_1 \to \cdots \to l_m \to m$，则记 $l \succeq m$;
若 $l \succeq m$ 且 $m \succeq l$，则记 $l \sim m$.

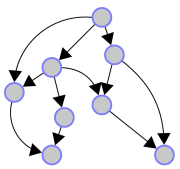
Define $\ell \succeq m$ iff there is a sequence of literals $\ell = \ell_0, \ldots, \ell_k = m$ ($k \geq 1$) such that $\ell_i \to \ell_{i+1}$ or $\bar{\ell}_{i+1} \to \bar{\ell}_i$ is in Γ for each i ($0 \leq i < k$). Thus, $\succeq$ is a pre-order (reflexive and transitive). Write $\ell \sim m$ if $\ell \succeq m$ and $m \succeq \ell$. 先根次序

知：Γ is satisfiable $\Leftrightarrow$ 存在 l, s.t. $l \not\sim \bar{l}$. 因为：

It is obvious that Γ is unsatisfable if there exists a literal ℓ such that $\ell \sim \bar{\ell}$.

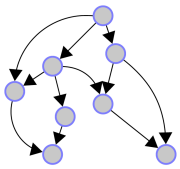
(so for every literal l in Γ, just need to check if there is "$l \sim \bar{l}$". if so, then it's unsatisfiable)

To prove the converse, consider the partial order (reflexive, transitive and anti-symmetric) induced by $\succeq$ on the equivalence classes of $\sim$



Note that if ℓ and m are in the same equivalence class, then so are $\bar{\ell}$ and $\bar{m}$. So equivalence classes come in 'opposite pairs'.
Suppose that ℓ is never equivalent to $\bar{\ell}$.

Start with some (undecided) equivalence class C lowest in the partial order, and make all its literals true (no contraditions). Make all its literals in the opposite equivalence class, say $\bar{C}$, false. (no contraditions). 也即是 "所有不在这个等价类中的东西"



We can show that $\bar{C}$ will not yet have been made false, and that all literals in classes higher up in the partial order will already have been made false.

下面引入 "co-C" 的定义：

If $\mathcal{C}$ is a class of languages (over some alphabet Σ), then co-$\mathcal{C}$ is the class of languages

$$\{L \subseteq \Sigma^* \mid (\Sigma^* \setminus L) \in \mathcal{C}\}.$$ class of complement

由于已证：Krom-UNSAT $\subseteq$ NLogSpace : KROM-SAT is in co-NLogSpace.

3. QBF (Quantified Boolean formula)

首先定义QBF：

A quantified Boolean formula is an expression

$$Q_1 p_1 \cdots Q_m p_m \psi,$$

where ψ is a formula of propositional logic and, for all i ($1 \leq i \leq m$), $p_i \in \mathbf{P}$ and $Q_i \in \{\forall, \exists\}$.

即：形如 "$\exists p_1 \forall p_2 (p_1 \vee p_2)$. $\forall p_1 \exists p_2 (p_1 \to p_2)$. ..."

并且定义QBF的语义：

1. QBF 的 assignment：

a function $\theta: P \to \{T, F\}$.

2. 并且记关于某个 literal 有相同 interpretation 即只关于 p 解释不同的两个 assignment θ 和 θ' : $\theta \approx_p \theta'$.

3. 并将 assignment θ 扩充到 QBF 上:

$$\theta(\neg\varphi_1) = T \text{ iff } \varphi_1 = F$$
$$\theta(\varphi_1 \vee \varphi_2) = T \text{ iff } \theta(\varphi_1) = T \text{ or } \theta(\varphi_2) = T$$
$$\cdots$$
$$\theta(\exists p.\varphi_1) = T \text{ iff } \theta'(\varphi_1) = T \text{ for some } \theta' \text{ s.t. } \theta' \approx_p \theta$$
$$\theta(\forall p.\varphi_1) = T \text{ iff } \theta'(\varphi_1) = T \text{ for all } \theta' \text{ s.t. } \theta' \approx_p \theta.$$

4. 称不含自由变量的 QBF 为 quantified Boolean Sentence.

故对于 QBF 有以下的问题:

QBF
Given: A quantified Boolean sentence φ.
Return: Yes if φ is true, and No otherwise.

(关心 QBF 的值).

并且有结论:

Theorem:
QBF is in PSpace.

证明: 下列的算法: QBF-Test is in PSpace.

this is a partial assignment, will be expanded

- For $0 \le k \le m$, we take $\alpha = (v_1, \ldots, v_k)$ to be an assignment of v_i (either T or F) to p_i $(1 \le i \le k)$.

```
begin QBF-test(Q_1 p_1 ... Q_m p_m.ψ)
    return QBF-rec(Q_1 p_1 ... Q_m p_m.ψ, ( ))

begin QBF-rec(Q_i p_i ... Q_m p_m.ψ, α)
    if i = m + 1
        return α(ψ)
    if Q_i = ∀ and QBF-rec(Q_{i+1} p_{i+1} ... |Q_m p_m.ψ, (α, F)) = F
        return F
    if Q_i = ∃ and QBF-rec(Q_{i+1} p_{i+1} ... Q_m p_m.ψ, (α, F)) = T
        return T
    return QBF-rec(Q_{i+1} p_{i+1} ... Q_m p_m.ψ, (α, T))
```

多x: Hardness, Reduction, Cook-Levin Thm, NLogSpace & PSpace's completeness

问题: 在多xx中: 我们对不同的时空复杂度,确定了位于某中的问题 => 这些复杂度类均有上界. 问:
对某个给定的时空复杂度类,找出该类中 lowest complexity bound.

定义: many-one logspace reducible
记 P_1, P_2 分别为定义在字母表 Σ_1 和 Σ_2 上的 languages. 若 $\exists$ function $f: \Sigma_1^* \longrightarrow \Sigma_2^*$, 使得:

1) 转换函数可由一个空间复杂度为 $\log n$ 的 deterministic TM 所计算. (注: 回顾"calculates"的概念.)
2) 对 $\forall x \in \Sigma_1^*$, $x \in P_1$ iff $f(x) \in P_2$.

则称 P_1 is (many-one logspace) reducible to P_2, 并且记: $P_1 \leq_m^{\log n} P_2$.
- P_1 can be reduced to P_2
- P_2 can be reduced from P_1.

并且已知 $P_1 \leq_m^{\log n} P_2$, 显然有:
1) P_2 is at least as hard as P_1. (even if P_1 is solved, still needs to convert solution in P_1 to P_2 forms)
2) P_1 is no harder than P_2. (to solve P_1, just convert P_1 to P_2, then use the new method to solve it)
3) iff $\exists$ an easy way to solve P_2, then have an easy way to solve P_1.

下面以一个 Thm 为例子:

★ Theorem: "$SAT \leq_m^{\log n} 3\text{-SAT}$":

证明:
* 首先定义 SAT 和 k-SAT:

Definitions:

SAT
Given: A set of clauses Γ
Return: Y if Γ is satisfiable, and N otherwise

k-SAT
Given: A set of clauses Γ each of which has at most k literals.
Return: Y if Γ is satisfiable, and N otherwise.

* 下面说明,对于任何 SAT 可被转换为 3-SAT:

Let Γ be a set of (propositional logic) clauses.
Suppose there exists $\gamma = (\ell_1 \vee \cdots \vee \ell_m) \in \Gamma$ with $m \geq 4$. ← 取一般情形的 SAT
Let p be a new proposition letter, and let Γ' be the result of replacing γ in Γ with the pair of clauses:

$p \vee \ell_3 \vee \cdots \vee \ell_m$ ← 实质上和 $\ell_1 \vee \cdots \vee \ell_m$ 的 satisfiability 相同.
$\neg p \vee \ell_1 \vee \ell_2$ ← 即: $p \longrightarrow (\ell_1 \vee \ell_2)$

故知: $\Gamma' \longrightarrow \Gamma$:
若 Γ 有一个 satisfying assignment θ, 则它亦 satisfies $\gamma = \ell_1 \vee \cdots \vee \ell_m$.
只需将 $\theta(p) := \theta(\ell_1 \vee \ell_2)$, 则可使 θ 亦满足 $p \vee \ell_3 \vee \cdots \vee \ell_m$ 和 $\neg p \vee \ell_1 \vee \ell_2$
=> 这样的 θ 亦 satisfies Γ'. => Γ' 和 Γ 的可满足性是一样的 (equisatisfiable)//

反复做对 Γ 中子式的替换,就可最终得到一个和 Γ equisatisfiable 的 set of clauses: Γ^*.
其中, Γ^* 是一个 3-SAT. (all clauses in Γ^* has at most 3 literals).

由于构造 Γ^* 只需 polynomial time 和 $\log n$ space: 可知: $SAT \leq_m^{\log n} 3\text{-SAT}$. //

注: 对常见的 complexity classes, 如 LogSpace, NLogSpace, PTime, NPTime 等,
they are closed under many-one logspace reduction, 即:
对 P_1, P_2: 若 $P_1 \leq_m^{\log n} P_2$ => P_1 与 P_2, 则称 P_2 is closed under many-one logspace reduction.

但是: Time(n), Time(n^2)等: not closed under many-one logspace reduction.

下面介绍 many-one logspace reducibility 的性质:

Theorem:

many-one logspace reducibility is a transitive relation. 即:

若 $f_1: \Sigma_1^* \longrightarrow \Sigma_2^*$ 和 $f_2: \Sigma_2^* \longrightarrow \Sigma_3^*$ 均在 logarithmic space 中 computable, 则 $f_2 \circ f_1: \Sigma_1^* \longrightarrow \Sigma_3^*$ 亦然.

证明: 可以构造出空间复杂度为 $\log n$ 的如下 Turing Machine:

f_1 first, then f_2

Here is a Turing machine that will compute $f_2 \circ f_1$ in logarithmic space:

calculate the first bit of $f_1(x)$
keep a counter to say which bit this is—initially 1
start a simulation of $f_2(f_1(x))$, using the calculated bit
if the simulation of f_2 asks to move the read head to the right
    calculate next bit of $f_1(x)$
    write it on top of the current bit
    update the output bit counter
if the simulation of f_2 asks to move the read head to the left
    restart the calculation of $f_1(x)$
    continue until the required output bit is calculated
    write it on top of the current bit
    update the output bit counter

注意: 证明某个问题是 many-one logspace reducible 在实践上很困难.

由于已知 $\text{LogSpace} \subseteq \text{NLogSpace} \subseteq \text{PSpace}$: 进一步引入了更弱但更方便的 many-one polynomial time reducible:

定义: many-one polytime reducible

记 P_1, P_2 分别为定义在字母表 Σ_1 和 Σ_2 上的 languages. 若 $\exists$ function $f: \Sigma_1^* \longrightarrow \Sigma_2^*$, 使得:

1) 转换函数可由一个时间复杂度为 polynomial 的 deterministic TM 所计算. (注: 回顾 "calculates" 的概念.)
2) 对 $\forall x \in \Sigma_1^*$, $x \in P_1$ iff $f(x) \in P_2$.

则称 P_1 is (many-one polytime) reducible to P_2, 并且记: $P_1 \leq_m^p P_2$.

注: 1) many-one polytime reducible $\xrightarrow{\quad}$ many-one logspace reducible.
故 many-one logspace reducible 强于 many-one polytime reducible.

2) 和 many-one logspace reducible 一样, many-one polytime reducible 亦满足传递性.

下面讨论复杂度类的 upper-bound:

定义: (C-hard 和 C-complete)

记 'C' 为某个复杂度类, P 为某个 problem. 若对 $\forall P' \in C$, $P' \leq_m^{\log} P$, (回顾: 复杂度类指由一系列 ..... 的 languages 组成的...)
则称 P is C-hard.

若 P is C-hard 且 $P \in C$, 则称 P is C-complete. (显然它必为 C 的子集)

若某个 class of all problems, 其每一个 problem 均为 C-hard, 则亦称这个 complexity class 为 C-complete.

在讨论了关于某个 complexity class 的 hardness 与 completeness 的形式化定义后, 下面看一个具体例子:

Theorem: (Cook-Levin Theorem)

SAT is NPTime-Complete.

证明:

由此前知识已知: SAT $\in$ NPTime.

下面要证: $\forall p \in \text{NPTime}$, $P \leq_m^{\log} \text{SAT}$.

设 P 为复杂度类 NPTime 中的任一个问题. 记 M 为一个确定的, 以 $p(n)$ 为运行时间上界的 Turing Machine.

并假设 M 为单纸带的. 则可记 M 为:

$$M := \langle \Sigma, Q, q_0, T \rangle$$

注: M is non-deterministic!

其中: Σ ~ alphabet
Q ~ set of states
q_0 ~ initial state
Γ ~ set of transitions.

并且, $\forall \tau \in \Gamma$, τ 可表示为: $\langle s, a, t, b, \delta \rangle$.
其中: s ~ 当前状态,
a ~ head 当前读入的 char
t ~ 将转换到的状态,
b ~ 将在 tape 上覆写的 char
δ ~ head 的移动方向:

$\delta \in \{-1, 0, 1\}$. (left, stay, right).
$a, b \in \Sigma \cup \{\triangleright, \sqcup\}$.
$s, t \in Q$.

目标: 对任意给定的 P, 与之确定了 M
=> 对任何 P 的 instance x, 说明: x can be mapped to an instance of SAT. 且:
x is a "positive instance" of P iff corresponding SAT is satisfiable!

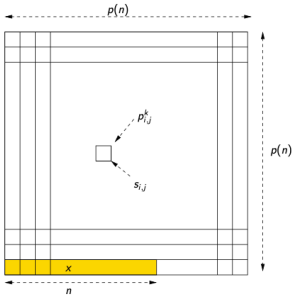
将 M 的所有可能 moves 的可能状态表示为一个 grid:

We picture the operation of M as

p(n): polynomial bounding M's running time

竖向的行数表示最多有多少个moves,
因此also bounded by p(n)!

而横向的squares同样受运行步数限制:
in worst case, 每一个move占用一个square,
因此最多占用p(n) 个square!



然后每一个 move 的状态 (or 局性) 即可用 propositional letters 描述:

$p^a_{i,j}$: tape square i contains symbol a at time j
$h_{i,j}$: the head is over tape square i at time j
q^s_j: the state is s at time j.
$t^\tau_{i,j}$: transition τ is executed at time j with head on tape square i.

注: Total # of prop. letters are bounded by $p(n)^2$
therefore also polynomial.

进而可以用这些 propositional letters 对任何 runs 编码.

下面对任一次 run 编码. 无需注意细节. 只用知道:
we can encode the operation of a given turing machine using propositional clauses with polynomial-bounded numbers.

We write clauses saying that, at each time, the head is somewhere

$$\{h_{1,j} \vee \cdots \vee h_{p(n),j} \mid 1 \le j \le p(n)\}$$

and is not in two places at once

$$\{\neg h_{i,j} \vee \neg h_{i',j} \mid 1 \le i < i' \le p(n), 1 \le j \le p(n)\}$$

and so on. We write clauses saying that the input is $x[1], \ldots, x[n]$ (remember $\sqcup$ is the blank symbol):

$$\{p^{x[i]}_{i,1} \mid 1 \le i \le n\}$$
$$\{p^{\sqcup}_{i,1} \mid n+1 \le i \le p(n)\}$$

and so on. (proof TBC ...) TBC: to be continued

Further, we write clauses specifying when a transition of M may be executed. For all i, j $(1 \le i, j \le p(n))$, and for all $a \in \Sigma \cup \{\sqcup, \triangleleft\}$, we take Γ_x to contain the (big) clause

deterministic turing machine => these clauses must be HORN
there's only one positive clause on the right

$$\neg q_j^s \vee \neg h_{i,j} \vee \neg p_{i,j}^a \vee \bigvee \{t_{i,j}^\tau \mid \tau = \langle s, a, t, b, \delta \rangle \in T\}$$

listing the allowed transitions M may make. Note that M is a non-deterministic TM!

And we write clauses specifying the effects of transitions:

$$\{\neg t_{i,j}^\tau \vee p_{i,j+1}^b \mid 1 \le i,j \le p(n),\ \tau = \langle s, a, t, b, \delta \rangle\}$$
$$\{\neg t_{i,j}^\tau \vee q_{j+1}^t \mid 1 \le i,j \le p(n),\ \tau = \langle s, a, t, b, \delta \rangle\}$$
$$\{\neg t_{i,j}^\tau \vee h_{i+\delta,j+1} \mid 1 \le i,j \le p(n),\ \tau = \langle s, a, t, b, \delta \rangle\}.$$

最后，在用 propositional letters 编码 M 后，可用 clauses 编码 / 表明：

M accepts the input $[q_{p(n)}^{s^*}, h_{1,p(n)}]$.

(at time $p(n)$, the state is s^*: halting state, while head is at location 1: leftmost pos, indicating succes/acceptance.)

⇒ Then denote the set of such clauses : T_x.

⇒ 则有：T_x is satisfiable iff M accepts x.

⇒ 即：T_x is satisfiable 当 $x \in P$, T_x 即为和 P 的 instance: x 对应的 SAT.

并且不难证明此结论：给定 x，计算 T_x 消耗的空间 $\le \log n$, $n = |x|$.

故有：function : $x \mapsto T_x$ indicates $P \le_m^{\log n} SAT$ //

基于 Cook-Levin Theorem，有下列结论：

1. Theorem.

3-SAT is NPTime-Complete.

证明：

由 { SAT $\le_m^{\log n}$ 3-SAT
SAT is NPTime-Complete
3-SAT $\in$ NPTime } ⇒ 3-SAT is NPTime-Complete.

2. Theorem.

Horn-SAT is PTime-Complete.

证明：由 DPLL，知 Horn-SAT $\in$ PTime.

并且：

Recall the clauses specifying when a transition can be executed for M in state s reading a:

$$\neg q_j^s \vee \neg h_{i,j} \vee \neg p_{i,j}^a \vee \bigvee \{t_{i,j}^\tau \mid \tau = \langle s, a, t, b, \delta \rangle \in T\}$$

If M is a deterministic, then these clauses have at most one positive literal, i.e. are Horn.

类似地利用 Cook-Levin Thm 即可证明：

for a deterministic Turing Machine M recognizing problem P: given instance x of problem P, obtained T_x is Horn! //

并且关于空间复杂度有以下结论：

1. Theorem:

st-CON is NLogspace-Complete.

证明：

We showed above that st-Con is in NLogSpace. (if there's a path, path length $\le$ tree height $= \log n$.)

now show it is NLogSpace-Hard:

Suppose L is a language recognized by a non-deterministic TM, M, running in space $O(\log n)$. Given an input x, let G be the

configuration graph for M with input x. Let $\mathfrak{c}_x$ be the node representing the initial configuration and $\mathfrak{d}$ the node representing the (cheating slightly) single accepting configuration. Now, $x \in L$ if and only if $(G, \mathfrak{c}_x, \mathfrak{d})$ is an instance of st-Con. The mapping $x \mapsto (G, \mathfrak{c}_x, \mathfrak{d})$ can easily be constructed in space bounded by $\log n$. □

2. Theorem:
QBF is PSpace-Complete.

证明:

We showed in a previous lecture that QBF is in PSPACE.

For PSPACE-completeness, suppose L is in PSPACE, and let M be a deterministic Turing Machine recognizing L and running with space bound $f(n)$, where f is a polynomial. Fix an input x for M of length n, and let G be the configuration graph of M with input x. The number of vertices in G is given by $c^{f(n)\log f(n)}$ where c is a constant depending only on M.

Let $\mathfrak{c}_x$ be the initial configuration with input x and $\mathfrak{d}$ the (cheating again) successful halting configuration. We want to know whether there is a path from $\mathfrak{c}_x$ to $\mathfrak{d}$ of length at most $c^{f'(n)}$. □

We encode each configuration with a sequence $\bar{p}$ of proposition letters of length $f''(n)$, (f'' a polynomial) and we can easily write a formula of propositional logic $\psi_0(\bar{p}, \bar{q})$ stating that the configuration encoded by $\bar{q}$ can be reached from that encoded by $\bar{q}$ in at most one step.

It suffices to write, for each i ($1 \leq i \leq \log(c^{f'(n)}) = f'(n) \cdot \log c$), a QB-formula $\psi_i(\bar{p}, \bar{q})$ true iff the configuration encoded by $\bar{q}$ is reachable from that encoded by $\bar{p}$ in at most 2^i steps. Note that ψ_0 has already been defined. □

Our first thought might be to to write

$$\psi_{i+1}(\bar{p}, \bar{q}) := \exists \bar{r}(\psi_i(\bar{p}, \bar{r}) \wedge \psi_i(\bar{r}, \bar{q})).$$

Too big! So, instead, we write

from \phi_{i} to \phi_{i+1}:
quantify states too!

$$\psi_{i+1}(\bar{p}, \bar{q}) := \exists \bar{r} \forall \bar{a} \forall \bar{b}(((\bar{a} = \bar{p} \wedge \bar{b} = \bar{r}) \vee (\bar{a} = \bar{r} \wedge \bar{b} = \bar{q})) \rightarrow \psi_i(\bar{a}, \bar{b})),$$

where $\bar{a} = \bar{p}$ abbreviates $(a_1 \leftrightarrow p_1) \wedge (a_2 \leftrightarrow p_2) \wedge \cdots$, etc.

It is then easy to map x to a QB-formula φ_x such that x is accepted by M (i.e. $x \in L$) if and only if φ_x is true. □

第X1. Two NP-Complete Problems

目标：介绍两个 NP-complete 的问题.

1. 3-Colorability. (3-染色问题)

首先引入对 3-Colorability 问题的定义：

> Let $G = (V, E)$ be a graph. A k-colouring of G is a function $f : V \to \{0, 1, \ldots, k-1\}$ such that, for any edge $(u, v) \in E$, $f(u) \neq f(v)$.

并且称：G is k-colorable 当 $\exists$ k-coloring for G.

问题的形式化表述：

> k-COLOURABILITY
> Given: A graph G.
> Return: Yes if G is k-colourable, and No otherwise.

并且显然：k-coloring $\in$ NPTime.

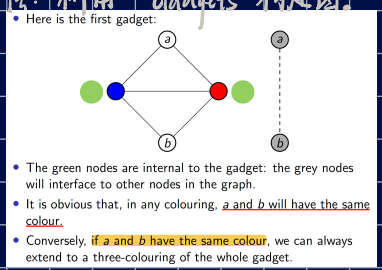
下面证明 3-COLORABILITY is NPTime-Hard：

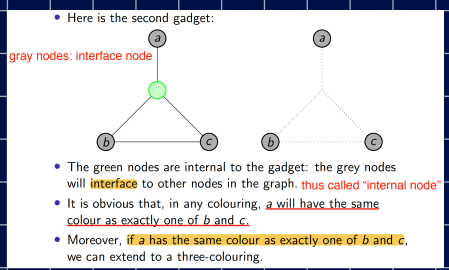
利用证明 3-SAT $\leq_{P}$ 3-COLORABILITY 实现.

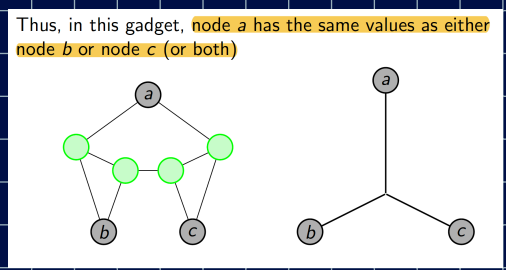
目标：given a set of 3-literal clauses Γ, compute in logspace a graph G_Γ

并说明：Γ is satisfiable $\Leftrightarrow$ G_Γ is 3-colorable.

方法：利用 "Gadgets" 构建图.

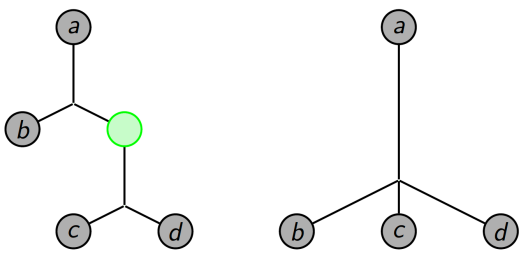






并且有：

> We can stick two of these gadgets together to make a fourth gadget, in which a has to have the same colour as any of b, c or d (with all such colourings possible)



然后就可基于一定的规则构造.

课上给了一个计算简单 3-SAT 的例子.

2. Hamiltonian Circuits 问题.

首先给出一系列基础定义：在连通图 G 中：

1. Cycle：
   a sequence of vertices: $v_0, \cdots, v_{n-1}$, s.t. $\forall i \in [0, n-1]$: $\langle v_i, v_{(i+1) \bmod n} \rangle \in E$.

2. Eulerian Circuit of G:
   a cycle in G, that each edge of G is traversed exactly once. (即每个 vertex 至少被 traverse 了一遍).

3. Hamiltonian Circuit of G:
   a cycle in G that each vertex of G is encountered exactly once. (即可以有的 edge 没被 traverse).

注：显然知：判断给定图是否有 Eulerian Circuit 是 easy 的：

只需判断给定图中有几个 vertex 的度是否为奇数：
若有任一个点为奇数，则必没有 Eulerian Circuit！

而判断 Hamilton Circuit 非常困难：实际上是 NP-Hard 问题（因此，且称 Hamilton-Circuit 问题为 NPTime-Complete）

形式化描述 Eulerian Circuit 和 Hamilton Circuit 问题：

EULERIAN-circuit	
Given:	A graph G
Return:	Yes if G has an Eulerian circuit, and No otherwise.

HAMILTONIAN-circuit	
Given:	A graph G
Return:	Yes if G has an Hamiltonian circuit, and No otherwise.

Theorem：(Euler)
a connected graph has an Euler Circuit iff every node is degree is even.

证明：

The only-if-direction is obvious. For the if-direction, let

$$v_0, \ldots, v_m$$

be the longest walk in G in which no edge is traversed more than once. Since every node has even degree, $v_m = v_0$. Suppose $e \in E$ is not in this walk. By connectedness of G let e be (v_i, u), where $0 \leq i < m$. Then

$$v_i, \ldots, v_m (= v_0) v_1, \ldots, v_i u$$

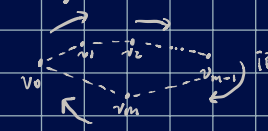
is a longer walk in G in which no edge is traversed more than once. Contradiction.

显然.

证明 "∃ Eulerian Circuit" ⇒ Every node's degree is even

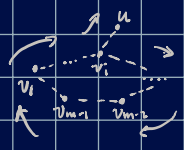
证明 "⇐"：

设有个 longest walk：$v_0, v_1, \cdots, v_{m-1}, v_m$. 则有：



证 $v_m = v_0$，因为 Connected Graph.

设：若 $e = \langle v_i, u \rangle \notin$ this walk. then:



则为一条 "no edge traversed more than once" 的 longer path. 矛盾. //

数有推论：
Corollary：
Eulerian-Circuit is in Log Space, ⇒ in PTime.

下面说明：Hamilton-Circuit is NPTime-Complete.
方法：consider directed version of Hamilton-Circuit problem:

DHAMILTONIAN-circuit	
Given:	A directed graph G
Return:	Yes if G has an Hamiltonian circuit, and No otherwise.

即有定理：
Theorem：
DHamiltonian-Circuit is NPTime-Complete.

证明：通过说明：
- DHamiltonian Circuit ∈ NPTime
- $SAT \leq_m^{\log}$ DHamiltonian Circuit.

并且不讨论具体的证明细节.

然后可得：
Theorem：
Hamiltonian-Circuit is NPTime-Complete.

证明：说明 DHamiltonian-Circuit $\leq_m^{\log}$ Hamiltonian-Circuit：

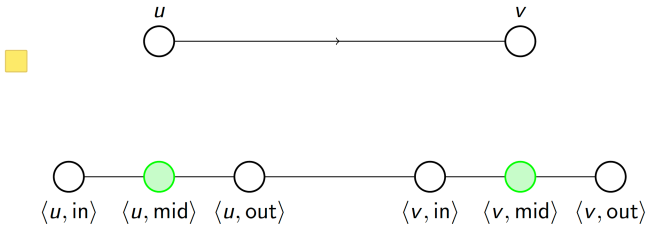
Membership in NPTIME is obvious. We prove NPTIME-hardness by easy reduction from DHAMILTONIAN-circuit. Given a directed graph $G = (V, E)$, form the undirected graph $H = (W, F)$, where

$$W = V \times \{\text{in}, \text{mid}, \text{out}\}$$

and

$$F = \{(\langle v, \text{in}\rangle, \langle v, \text{mid}\rangle), (\langle v, \text{mid}\rangle, \langle v, \text{out}\rangle) \mid v \in V\} \cup \{(\langle u, \text{out}\rangle, \langle v, \text{in}\rangle) \mid (u, v) \in E\}.$$

Thus, an edge of G becomes a path in H as follows:



It is straightforward to show that G has a Hamiltonian circuit if and only if H does. □

下面是一步引入 TSP（旅行商）问题的定义：

Definition (TSP Problem).

给定存储了图中每一条边距离（或权）的对称矩阵 $A_{(n\times n)}$：

1. 首先，称 a tour in A is a permutation of set $\{1, \dots, n\}$. （即指明访问的每一个 vertex）
2. 并称 length of a tour 为：

$$a_{p_1 p_2} + \cdots + a_{p_{n-1} p_n} + a_{p_n p_1}.$$

⇒ 称 TSP 问题为：

TSP
Given: A symmetric matrix A with entries in $\mathbb{N}$.
Return: A tour in A of minimal length.

注："Return" 不关心 tour 的顺序，但评估问到每一个点。
"关心 minimal length (cost) 的 tour.

并且 "找到这样的 tour" is not significantly harder than TSP Feasibility：

TSP feasibility
Given: A symmetric matrix A with entries in $\mathbb{N}$ and $k \in \mathbb{N}$.
Return: Y if A has a tour of length $\leq k$; N otherwise.

"是否有 length (cost) ≤ K 的 tour.

关于 TSP 问题，其时间复杂度结论为：

Theorem：

TSP-Feasibility is NPTime-Complete.

证明：

Membership in NPTIME is trivial. For NPTIME-hardness, we proceed by reduction from HAMILTONIAN-circuit.

Let $G = (V, E)$ be a graph with vertices $\{1, \dots, n\}$. Let A be the $n \times n$ symmetric matrix with entries

$$a_{i,j} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 2 & \text{otherwise.} \end{cases}$$

Then G has a Hamiltonian circuit iff there is a tour in A of length n. To the mapping $G \mapsto (A, n)$ is a (log-space) reduction from HAMILTONIAN-circuit to TSP feasibility. □

说明：TSP-Feasibility 可由 Hamilton-Circuit 问题 reduce 得来.

§XII. Integer Linear Programming Feasibility

目标：介绍 Linear Programming Feasibility 和 Integer Linear Programming Feasibility 问题，并讨论它们的时间复杂度与可行性。

首先形式化定义 Linear Programming Feasibility 问题：

> **LP feasibility**
> Given: a system of linear inequalities $\mathcal{E}: A\underline{x} \le \underline{b}$ with integer coefficients.
> Return: Yes if $\mathcal{E}$ has a solution over $\mathbb{R}^+$ and No otherwise.

注：解可以为 $\mathbb{R}^+$，但系数仅为 Integer：+、−、0。

或用另一种方式定义：

> **LP feasibility (alternative form)**
> Given: a system of linear inequalities $\mathcal{E}: A\underline{x} = \underline{b}$ with integer coefficients.
> Return: Yes if $\mathcal{E}$ has a solution over $\mathbb{R}^+$ and No otherwise.

下面介绍几个有助于后面分析 LP 问题时间复杂度的 Thm：

☆ Theorem：(Caratheodory)
记 $\mathcal{E}$ 为由 m 个线性方程组所构成的 system，若 $\mathcal{E}$ 有在定义于 $\mathbb{R}^+$ 上的解，则它任何解中，最多 m 项为正。

证明：
记 $\mathcal{E}$ 为：$A\vec{x} = \vec{b}$。不妨设 $\vec{x} \in \mathbb{R}^+$，为一个非零项数目最少的解，记该数目为 n。
除去 A 中所有和 $\vec{x}$ 中 0 值 entry 对应的变量，则此时 A 只有 n 列。（A 中每一列代表一个变量）。

若 $m < n$：则知 A 的列向量一定线性相关，故 $\exists \vec{y}$，s.t. $A \cdot \vec{y} = \vec{0}$。
取任何 $\lambda > 0$，均有：$A \cdot (\lambda \cdot \vec{y}) = \vec{0}$。并且知：$\vec{y}$ 中至少有一个 entry 为正值。

下面知：$A(\vec{x} - \lambda\vec{y}) = \vec{b}$（因为 $A\vec{x} = \vec{b}$ 且 $A \cdot \lambda\vec{y} = 0$）。让正数 λ 逐渐↑，直到 $(\vec{x} - \lambda\vec{y})$ 中出现第一个新的 0 项，此时我得到了一个比 $\vec{x}$ 含非零 entry 更小的解，矛盾。//

☆ Theorem：
记 $\mathcal{E}$ 为某个系数绝对值最大不超过 M 的 linear equation system，若 $\mathcal{E}$ 有一个 $\mathbb{R}^+$ 上的解，则它在 $\mathbb{Q}^+$ 上亦有解，且 $\mathbb{Q}^+$ 上解的分子和分母的不超过 $(n^m M)^m$。

证明：
记 $\mathcal{E}$：$A\vec{x} = \vec{b}$。记 A 有 n 列。由 Caratheodory 定理：取在 $\mathbb{R}^+$ 上，非零系数的个数最小的解，去掉 A 中所有与零系数相关的变量对应的列，得到的列数 $\le m$。

使用 Gaussian 消元法和 Cramer's rule 求解，即得结论。//

并且可知：由于 $\mathcal{E}$：$A\vec{x} = \vec{b}$ 的解它每一项的值均为 polynomial bounded：

☆ Corollary：
LP-Feasibility is in NP Time.

同时，由于下列引理：

Lemma：(Farkas)
记 $A \in \mathbb{R}^{m \times n}$，$\vec{b} \in \mathbb{R}^m$。则：下列 assertion 中，恰有一个为真：

1. $\exists \vec{x} \in \mathbb{R}^n$，s.t. $A\vec{x} = \vec{b}$ 且 $\vec{x} \ge \vec{0}$。
2. $\exists \vec{y} \in \mathbb{R}^m$，s.t. $A^T \cdot \vec{y} \ge 0$ 且 $\vec{b} \cdot \vec{y} < 0$。

⇒ 有下列结论：

☆ Corollary:
LP-Infeasibility is in NPTime.

由上面两个引理知:
LP-Feasibility ∈ NPTime ∩ Co-NPTime.

并且下面不加证明地给出结论:

☆ Theorem: (Khachiyan)
LP-Infeasibility is in PTime.

故有结论:
1. LP-Feasibility ∈ NPTime & Co-NPTime.
2. LP-Infeasibility ∈ PTime.

下面讨论 ILP: Integer Linear Programming Feasibility 问题:

ILP feasibility
Given: a system of linear equations $\mathcal{E}$: $A\underline{x} \leq \underline{b}$ with integer coefficients.
Return: Yes if $\mathcal{E}$ has a solution over $\mathbb{N}$ and No otherwise.

和 LP 一样: require coefficients to be integer
但只关心非负整数解.

探究: 通过引入松弛变量,转化为等式线性方程组.

- By adding slack variables, we can consider instead the feasibility (over $\mathbb{N}$) of systems of linear equations :

$$\begin{array}{rcl} a_{1,1}x_1 + \ \ldots + & a_{1,n}x_n & = b_1 \\ \vdots & \vdots & \vdots \\ a_{m,1}x_1 + \ \ldots + & a_{m,n}x_n & = b_m. \end{array}$$

- A equation in which all coefficients are integers and solutions are sought over the non-negative integers is called Diophantine .

注: 1. ILP 和 LP 问题不同! 前者 require solutions over $\mathbb{N}$!
2. Caratheodory 定理因此不适用于 solutions over $\mathbb{N}$.
3. Diophantine equation: 整数系数方程,求解仅在整数范围内进行.

有下面的定理:

☆ Theorem:
记 $\mathcal{E}$ 为一个 system with m linear Diophantine equations $A\vec{x} = \vec{b}$ in n variables.
若 $\mathcal{E}$ 有解,则其解的每一项均不超过 $(2n \cdot M)^m$. 其中 M 为 A 和 b 中绝对值最大项的值.

证明: 首先考虑齐次线性方程组的情况:

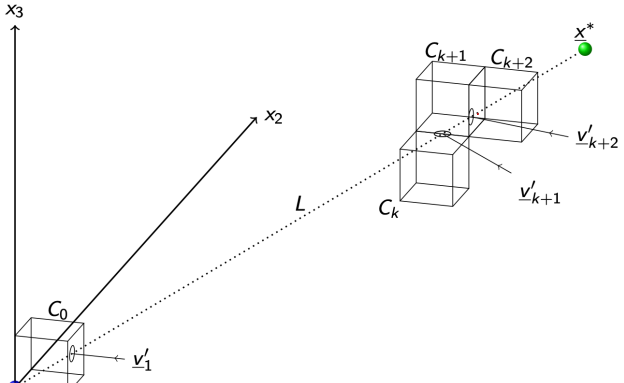
Consider the pointwise linear order on vectors in $\mathbb{N}^n$ defined by

$$(d_1, \ldots, d_n) \preceq (e_1, \ldots, e_n) \text{ if } d_i \leq e_i \text{ for all } i \ (1 \leq i \leq n).$$

This order is well-founded : in any infinite sequence of vectors $\underline{d}_1, \underline{d}_2 \ldots$, there exist $i < j$ with $\underline{d}_i \preceq \underline{d}_j$.
So if the system $A\underline{x} = \underline{0}$ has a non-trivial solution in $\mathbb{N}$, it has a minimal non-trivial solution under $\preceq$.

Let $\underline{x}^*$ be a minimal non-trivial solution of $A\underline{x} = \underline{0}$.

Now consider the line L joining $\underline{0}$ to $\underline{x}^*$.



well-founded: 该 order 总有下确界.

形如 $(x_1, x_2, \ldots, x_n)$.

此处为仅于 visualisation 用了 3 维度为 3 的情形.

注: 由于假设: $\underline{x}^*$ is minimal & non-trivial:
所有和 grid 相交的点代表值,此对应的解均为整数解!

将所有选择和在 grid 相交的符 $\vec{0} = \vec{v}_0, \vec{v}_1, \cdots, \vec{v}_{t-1}, \vec{v}_t = \vec{x}^*$ 满足: $\vec{v}_0 \perp \vec{v}_1 \perp \cdots \perp \vec{v}_t = \vec{x}^*$.

并且对于任何 $\vec{v}_i$: $\exists \vec{v}_i'$, 它位于 scalar line: L 上, 且 $\vec{v}_i$ 和 $\vec{v}_i'$ differs no more than 1 along any dimension.

记 $\vec{w}_k = A \cdot \vec{v}_k$: $\forall k \in [0, t]$. 则有:
对于 $\vec{w}_0, \cdots, \vec{w}_t$: $\vec{w}_k = A\vec{v}_k = \underbrace{A\vec{v}_k}_{\vec{v}_k \text{非}0} - \underbrace{A\vec{v}_k}_{\vec{v}_k \text{为}0} = A(\vec{v}_k - \vec{v}_k')$. 这个很重要!

故:

Hence every entry in any of the vectors $\underline{w}_k$ has absolute value at most $M + \cdots + M \leq nM$.

并且知: 由于假设中 $\vec{x}^*$ 的 minimality, $\vec{w}_0, \cdots, \vec{w}_t$ are all distinct:

- If $\underline{w}_j = \underline{w}_k$ with $j < k$ then
$$A(\underline{v}_k - \underline{v}_j) = A\underline{v}_k - A\underline{v}_j = \underline{0}$$
so that $\underline{v}_k - \underline{v}_j$ is a solution, contradicting minimality of $\underline{x}^*$.
- Hence, $\underline{w}_0, \ldots, \underline{w}_t$ are all distinct.

即由于 entries $\leq |M \cdot n|$ 的 integer vectors 最多有 $(2M \cdot n + 1)^m$ 个 (m 个 entry, each entry $\in [-Mn, Mn]$)
故 $\vec{x}^*$ 可能的取值受多项式限定: 其第一项不超过 $(2M \cdot n + 1)^m$. //

然后关注 $A\vec{x} = \vec{b}$:

Solutions of $A\underline{x} = \underline{b}$ are exactly the solutions of

$$(A \mid -\underline{b}) \begin{pmatrix} \underline{x} \\ y \end{pmatrix} = \underline{0},$$

with $y = 1$.

Now apply the same reasoning as before, but confining attention to solutions $\underline{x}^*$ in which the coordinate corresponding to y is 1.

知可得相同结论. //

故由上面的 Thm, 可推得结论:

Theorem:

ILP-Feasibility is in NPTime.

下面再证明 ILP-Feasibility is NPTime-Hard:

Theorem:

ILP-Feasibility is NPTime-Hard.

证明:

只需说明 3-SAT $\leq_m^{\log}$ ILP-Feasibility.

We show that 3-SAT $\leq_m^{\log}$ ILP.

Suppose we are given a set of 3-literal clauses Γ. We show how to compute system of linear Diophantine equations $\mathcal{E}$ such that $\mathcal{E}$ has a solution over $\{0, 1\}$ iff Γ is satisfiable.

For every proposition letter p mentioned in Γ, let x_p and $x_{\neg p}$ be variables and write the equation

$$x_p + x_{\neg p} = 1.$$

For every clause $\gamma := (\ell_1 \vee \ell_2 \vee \ell_3) \in \Gamma$, let y_1^γ, y_2^γ, z_1^γ, z_2^γ be variables, and write the equations

$$\begin{aligned} x_{\ell_1} + x_{\ell_2} + x_{\ell_3} + y_1^\gamma + y_2^\gamma &= 3 \\ y_1^\gamma + z_1^\gamma &= 1 \\ y_2^\gamma + z_2^\gamma &= 1 \end{aligned}$$

Call the resulting system of equations $\mathcal{E}_\Gamma$.

We claim that the mapping $\Gamma \mapsto \mathcal{E}_\Gamma$ is a reduction from 3SAT to ILP-feasibility. □

Suppose θ is a truth-value assignment for the proposition letters in Γ. Now define

$$x_p = \begin{cases} 1 & \text{if } \theta(p) = \top \\ 0 & \text{otherwise.} \end{cases}$$

and define $x_{\neg p} = 1 - x_p$ (thus satisfying $x_p + x_{\neg p} = 1$).

If θ makes $\gamma := (\ell_1 \vee \ell_2 \vee \ell_3)$ true, then we can certainly find $y_1^\gamma, y_2^\gamma \in \{0, 1\}$ satisfying.

$$x_{\ell_1} + x_{\ell_2} + x_{\ell_3} + y_1^\gamma + y_2^\gamma = 3.$$

Setting $z_i^\gamma = 1 - y_i^\gamma$, all the equations in $\mathcal{E}_\Gamma$ are satisfied. □

Conversely, suppose we have a solution of $\mathcal{E}_\Gamma$ over $\mathbb{N}$.

All values x_p and $x_{\neg p}$ are 0 or 1. Define the truth-value assignment

$$\theta(p) = \begin{cases} \top & \text{if } x_p = 1 \\ \bot & \text{otherwise.} \end{cases}$$

All values y_1^γ and y_2^γ are 0 or 1. Hence, from the equations

$$x_{\ell_1} + x_{\ell_2} + x_{\ell_3} + y_1^\gamma + y_2^\gamma = 3,$$

every clause in Γ is made true by θ. □

Theorem:

ILP-feasibility is NPTime-complete. //

§XIII. Two Thms on Space Complexity

目标：介绍两个关于 Space Complexity 的重要 Thm ⇒ Simplify the hierarchy of complexity classes + Tidy-up KROM+SAT's proof.

首先以 st-CON 为例引入 Savitch's Thm：

1. 回顾 st-CON 问题：

> **st-CON**
> Given: A directed graph $G = (V, E)$ and nodes $s, t \in V$
> Return: Yes if t is reachable from s in G, No otherwise.

已知：st-CON ∈ linear time
且 st-CON ∈ NLogSpace

2. 考虑下面的算法：

```
begin isReachableNum(u,v,G,h)
    if h = 0
      if u = v or (u,v) ∈ E return Yes
      else return No
    for w ∈ V
      if (isReachableNum(u,w,G,h−1) and
          isReachableNum(w,v,G,h−1)) return Yes
    return No
```

作用：determine if there's any paths in directed graph G, from u to v, with length no more than 2^h. (exponential-bounded st-CON)

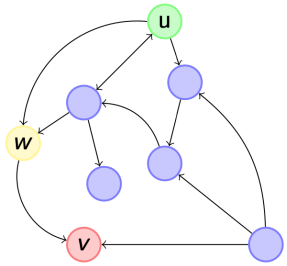
check if there's any "intermediate nodes" in the path
call itself recursively

if we need to determine connectivity between u and v:
Call `isReachableNum(u,v,G,⌈log|V|⌉)`, where $G = (V, E)$.

并说明算法的工作原理：

according to the algorithm:

Any path from u to v of length $\leq 2^h$ must have a midpoint w



There is a path from u to w of length $\leq 2^{h-1}$ and a path from w to v of length $\leq 2^{h-1}$

使用 Turing Machine 实现：
record a triple $\langle u, v, h \rangle$ every call to this function is made.

when running:
1. if any call succeed: write corresponding triple on the work tape.
2. if any call fails: move to the next intermediate point 'w' and overwrite the tape

故可知：该算法若 implement 为确定性 Turing Machine，则其 space complexity 为 $O(\log^2 n)$.

由此可引出 Savitch 算法的 First form：

☆ **Theorem: (Savitch, First form)**
st-CON is in Space($\log^2 n$).

并且 Savitch Theorem 对 Computational Graph 的计算有如下的重要结果：

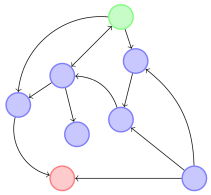
* consider M: 1-tape turing machine, 则它的 configuration 可记为：$\langle s, w, i \rangle$
  其中：$s \sim$ a state
  $w \sim$ a word over M's alphabet: "the content of the tape"
  $i \sim$ a head position.
* 若记 $w = a_1 a_2 \cdots a_{n-1} a_n$：则可将该 configuration 记为：
  $a_1 \cdots a_{n-1}\, s\, a_n \cdots a_\ell$
  并将它记录在另一个 work tape 上. ⇒ 同时也表示了 configuration graph 上的一个 node.
* 并且还有：
  给定某个 configuration：
  $a_1 \cdots a_{n-1}\, s\, a_n \cdots a_\ell$
  并假设 M 的某个 transition：$\langle s, a_n \mapsto b, \text{right}, t \rangle$

in state s, if reading a_h, then overwrite it as b,
move the head to right and transition the state to t

则可以直接计算出 subsequent configuration.

* 由此知:

- M has an accepting run on input x iff there is a path from the initial state of M with input x to an accepting state.



- If M runs in space f and $n = |x|$, the number of nodes of G is bounded by $f(n) \cdot c^{f(n)}$ for some constant c.

the possible location of the head => roughly considered as f(n)
total possible strings: c^{f(n)}, where c roughly interpreted as number of symbols

进一步: bounded by $2^{O(f(n))}$.

(以上全是在复习 configuration graph 相关定义…)
由 Savitch Thm (Form 1) 知:
可以在 Space $(\log^2|V|)$ 内找到 $|V| \le 2^{O(f(n))}$ 的图中给定起、终点间的 path.

$\Rightarrow$ Space $(O(f(n))^2) \Rightarrow$ Space $(O(f^2(n)))$.

故证明了 Savitch Theorem 的 Second Form:

☆ Theorem: (Savitch, Second Form)
对 f: space constructable complexity fun 且 $f(n) \ge \log n$:
则 NSpace $\subseteq$ Space (f^2).

(回顾 Complexity 中的 Thm II: NSpace $\subseteq$ Time $(2^{O(f(n))})$.
只不过得到了 st-CON 的 time-complexity 结论).

证明: 由 Savitch Thm (First Form): st-CON is in Space $(\log^2 n)$.
又由 Configuration graph: $|V| \le 2^{O(f(n))}$.
知: 对给定的, non-deterministic TM running in Space (f) of a $P \in$ NSpace (f):
find its "accepting path" costs Space $(O(f^2(n)))$. //

由此有一系列的推论:

☆ { NP-Space = P-Space, NExpSpace = ExpSpace, etc.

NP-Space = co-NPSpace, NExpSpace = co-NExpSpace, etc. ☆ (Since: co-NPSpace = co-PSpace = PSpace = NPSpace ...)

问题: Savitch Thm 并不能说明: NLogSpace = co-NLogSpace! 但实际上该结论成立.
如何说明? 引入 Immerman-Szelepcsenyi Theorem !!

下面基于 Unreachability 问题 (st-CON 的 complement) 引入 Immerman-Szelepcsenyi 定理的 First Form:

UNREACHABILITY
Given: A directed graph $G = (V, E)$ and nodes $s, t \in V$
Return: Yes if t is not reachable from s in G, No otherwise.

目标: 证明 UNREACHABILITY is in NLogSpace.

下面首先定义一个 non-deterministic subroutine:

```
begin reachableLossy(u,v,k)
    set u' := u   <-- marker: curr loc
    until k = 0
        guess any node v'
        if u' ≠ v' and (u', v') ∉ E return No
        set u' := v'
        decrement k
    if u' = v return Yes
    return No
```

连着猜k次, 以此检测能否从u到v找到一条最长为k的路径.
如果在猜任何一次时发现随机选定的下一个节点无法和之前路径末尾节点连接, 则终止.

知: 若 it returns yes: definitely know ∃ a path
若 it returns no: we know nothing
(it may return no even if it should return yes)

然后来定义另一个 non-deterministic procedure:

Assume we have an algorithm `isReachableFail`(u, v, k) which, for $1 \leq k < n$, *either* returns $\perp$, Yes or No:

- `isReachableFail` has a run returning Yes, iff v is reachable from u in at most k steps;
- `isReachableFail` has a run returning No, iff v is not reachable from u in at most k steps;

still a non-deterministic procedure:
if return yes => then there's indeed a path
if return no => then there definitely has no such path
if return "fail" => then cannot be determined

then we can build this wrapper: return the number of reachable nodes in at most k steps

```
begin numReachableFail(u, k)
    if k = 0 return 1   the only node that is reachable in 0 step is itself
    set m = 0   maintain a counter
    for i = 0, ..., n − 1   for every nodes
        let Q = isReachableFail(u, u_i, k)   use this unreliable procedure to check if reachable
        if Q = ⊥, then return ⊥
        if Q = Yes, then increment m
    return m
```

Now define "isReachableFail":

```
begin isReachableFail(u, v, k)
    let s = numReachableFail(u, k − 1)
    if s = ⊥ then return ⊥
    let m = 0
    for i = 0, ..., n − 1
        if reachableLossy(u, u_i, k − 1) = Yes   non-deterministic search
            if u_i = v or (u_i, v) ∈ E then return Yes   guarantee   (found direct edge)
            increment m   no direct edge, increment counter.
    if m < s then return ⊥   (if m=s: we are lucky, the detector behaves reliable)
    return No
```

最后:

```
begin isUnreachable(u, v, (V, E))
    if isReachableFail(u, v, |V| − 1) = No
        then return Yes
    return No
```

我们有 Immerman - Szelepcsenyi Theorem 的 First Form:

Theorem: (Immerman - Szelepcsenyi First Form)
UNREACHABILITY is in NLogSpace.

和有:

Theorem: (Immerman - Szelepcsenyi Second Form)
若 f 为 space - constructable 的且 $f(n) \geq \log(n)$, 则 $NSpace(f) = Co\text{-}NSpace(f)$.

证明:

设 $P \in NSpace(f)$, $\bar{P}$ 为其 complement. 记 M 为一个 non-deterministic TM that runs in $Space(f)$ and accepting $\bar{P}$.

记 x 为 P 的某个 instance, 作为 M 的 input. 记 G 为 M 以 x 为 input 的 configuration graph

则有: x is a positive instance of $\bar{P}$ $\Leftrightarrow$ its corresponding (Starting) node in G is unreachable to success-end node in G

$\Rightarrow$ 判断 x 的 acceptance $\Leftrightarrow$ 判断 x 的 non-acceptance

$\Rightarrow NSpace(f) = Co\text{-}NSpace(f)$ //

故有下列推论:

1. KROM-SAT is NLogSpace-Complete. //
2. 由 Savitch:
   $\{$ PSpace = NPSpace

ExpSpace = NExpSpace, ...

3. NLogSpace ≠ NPSpace = PSpace.

Hence at least one of the inequalities

$$\text{NLogSpace} \subseteq \text{PTime} \subseteq \text{NPTime} \subseteq \text{PSpace}.$$

is strict; but it is not known which.

结论：

1. deterministic classes (time or space) are always equal to their complement classes;
2. non-deterministic space classes from NPSpace upwards are equal to their deterministic variants (Savitch) and hence to their complement classes;
3. NLogSpace is equal to its complement class (Immerman-Szelepcseńyi).