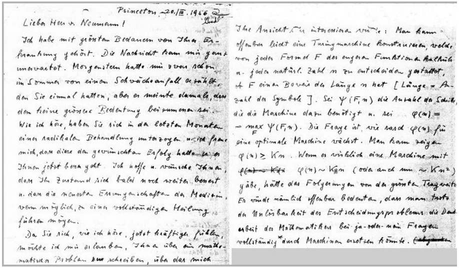
4-11 P and NP (II)

(NP $\neq$ No Problem)

Hengfeng Wei

hfwei@nju.edu.cn

May 27, 2019

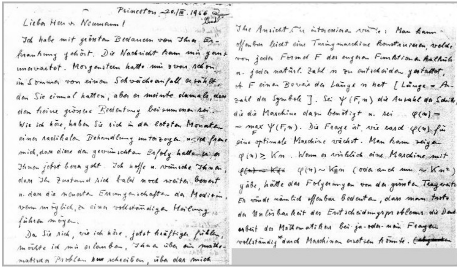


Princeton, 20./III. 1956

Lieber Herr v. Neumann!

Ich habe mit größtem Bedauern von Ihrer Erkrankung gehört. Die Nachricht kam mir ganz unerwartet. Morgenstern hatte mir zwar schon im Sommer von einem Schwächeanfall erzählt, den Sie einmal hatten, aber er meinte damals, dem keine grössere Bedeutung beizumessen sei. Wie ich höre, haben Sie sich in den letzten Monaten einer radikalen Behandlung unterzogen u. ich freue mich, dass diese den gewünschten Erfolg hatte u. es Ihnen jetzt besser geht. Ich hoffe u. wünsche Ihnen, dass Ihr Zustand sich bald noch weiter bessert u. dass die neuesten Errungenschaften der Medizin, wenn möglich, zu einer vollständigen Heilung führen mögen.

Da Sie sich, wie ich höre, jetzt kräftiger fühlen, möchte ich mir erlauben, Ihnen über ein mathematisches Problem zu schreiben, über das mich Ihre Ansicht sehr interessieren würde: Man kann offenbar leicht eine Turingmaschine konstruieren, welche von jeder Formel F des engeren Funktionenkalküls u. jeder natürl. Zahl n zu entscheiden gestattet, ob F einen Beweis der Länge n hat [Länge = Anzahl der Symbole]. Sei $\psi(F,n)$ die Anzahl der Schritte, die die Maschine dazu benötigt u. sei $\varphi(n) = \max_F \psi(F,n)$. Die Frage ist, wie rasch $\varphi(n)$ für eine optimale Maschine wächst. Man kann zeigen $\varphi(n) \geq Kn$. Wenn es wirklich eine Maschine mit $\cancel{\varphi(n) \sim Kn}$ $\varphi(n) \sim Kn$ (oder auch nur $\sim Kn^2$) gäbe, hätte das Folgerungen von der grössten Tragweite. Es würde nämlich offenbar bedeuten, dass man trotz der Unlösbarkeit des Entscheidungsproblems die Denkarbeit des Mathematikers bei ja-oder-nein Fragen vollständig* durch Maschinen ersetzen könnte. ~~(abgesehen~~



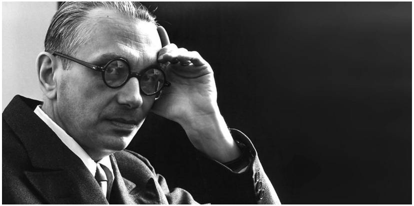
Princeton 20./III.1956

Lieber Herr v. Neumann!

Ich habe mit grösstem Bedauern von Ihrer Erkrankung gehört. Die Nachricht kam mir ganz unerwartet. Morgenstern hatte mir zwar schon im Sommer von einem Schwächeanfall erzählt, den Sie einmal hatten, aber er meinte damals, dass dem keine grössere Bedeutung beizumessen sei. Wie ich höre, haben Sie sich in den letzten Monaten einer radikalen Behandlung unterzogen u. ich freue mich, dass diese den gewünschten Erfolg hatte u. es Ihnen jetzt besser geht. Ich hoffe u. wünsche Ihnen, dass Ihr Zustand sich bald noch weiter bessert u. dass die neuesten Errungenschaften der Medizin, wenn möglich, zu einer vollständigen Heilung führen mögen.

Da Sie sich, wie ich höre, jetzt kräftiger fühlen, möchte ich mir erlauben, Ihnen über ein mathematisches Problem zu schreiben, über das mich

Ihre Ansicht sehr interessieren würde: Man kann offenbar leicht eine Turingmaschine konstruieren, welche von jeder Formel F des engeren Funktionenkalküls u. jeder natürl. Zahl n zu entscheiden gestattet, ob F einen Beweis der Länge n hat [Länge = Anzahl der Symbole]. Sei $\psi(F, n)$ die Anzahl der Schritte, die die Maschine dazu benötigt u. sei $\varphi(n) = \max_F \psi(F, n)$. Die Frage ist, wie rasch $\varphi(n)$ für eine optimale Maschine wächst. Man kann zeigen $\varphi(n) \geq Kn$. Wenn es wirklich eine Maschine mit ~~$\varphi(n) \sim Kn$~~ $\varphi(n) \sim Kn$ (oder auch nur $\sim Kn^2$) gäbe, hätte das Folgerungen von der grössten Tragweite. Es würde nämlich offenbar bedeuten, dass man trotz der Unlösbarkeit des Entscheidungsproblems die Denkarbeit des Mathematikers bei ja-oder-nein Fragen vollständig* durch Maschinen ersetzen könnte.



Kurt Gödel (1906 ∼ 1978)

John von Neumann (1903 ∼ 1957)

$\vdash F$

$\vdash F$: F is provable

$\vdash F$: F is provable

$\vdash^n F$: F has a first-order proof of $\leq n$ symbols

$\vdash F$: F is provable

$\vdash^n F$: F has a first-order proof of $\leq n$ symbols

$$\text{THEOREM} = \left\{ (F, 1^n) : \vdash^n F \right\}$$

$\vdash F$: F is provable

$\vdash^n F$: F has a first-order proof of $\leq n$ symbols

$$\text{THEOREM} = \left\{ (F, 1^n) : \vdash^n F \right\}$$

"If there really were a machine with
$\varphi(n) \sim k \cdot n$ (or even $\sim k \cdot n^2$),
this would have consequences of the greatest importance."

$$\text{THEOREM} = \left\{ (F, 1^n) : \vdash^n F \right\}$$

$$\text{THEOREM} = \left\{ (F, 1^n) : \vdash^n F \right\}$$

$$\text{THEOREM} \in \text{NP}$$

$$\text{THEOREM} = \Big\{ (F, 1^n) : \vdash^n F \Big\}$$

$$\text{THEOREM} \in \text{NP}$$

THEOREM is NP-complete.

$$\text{THEOREM} = \left\{ (F, 1^n) : \vdash^n F \right\}$$

$$\text{THEOREM} \in \text{NP}$$

THEOREM is NP-complete.

Definition (NP)

$$L \in \mathrm{NP}$$

$$\Longleftrightarrow$$

$\exists$ poly. time *verifier* $V(x, c)$ such that

$$\forall x \in \{0,1\}^* : x \in L \iff \exists c \text{ with } |c| = O(|x|^k), V(x,c) = 1.$$

NP-problems has short certificates that are easy to verify.

Theorem

$$P \subseteq NP \subseteq EXP$$

Theorem

$$P \subseteq NP \subseteq EXP$$

$$\text{P} = \left\{ L : L \text{ is decided by a poly. time } (O(n^k)) \text{ algorithm } A \right\}$$

$$\text{EXP} = \left\{ L : L \text{ is decided by an exp. time } (O(2^{n^k})) \text{ algorithm } A \right\}$$

Theorem

$$P \subseteq NP \subseteq EXP$$

$$\text{P} = \left\{ L : L \text{ is decided by a poly. time } (O(n^k)) \text{ algorithm } A \right\}$$

$$\text{EXP} = \left\{ L : L \text{ is decided by an exp. time } (O(2^{n^k})) \text{ algorithm } A \right\}$$

Proof.

Theorem

$$P \subseteq NP \subseteq EXP$$

$$\text{P} = \left\{ L : L \text{ is decided by a poly. time } (O(n^k)) \text{ algorithm } A \right\}$$

$$\text{EXP} = \left\{ L : L \text{ is decided by an exp. time } (O(2^{n^k})) \text{ algorithm } A \right\}$$

Proof.

$$\text{P} \subseteq \text{NP}$$

Theorem

$$P \subseteq NP \subseteq EXP$$

$$\mathrm{P} = \left\{L : L \text{ is decided by a poly. time } (O(n^k)) \text{ algorithm } A\right\}$$

$$\mathrm{EXP} = \left\{L : L \text{ is decided by an exp. time } (O(2^{n^k})) \text{ algorithm } A\right\}$$

Proof.

$$\mathrm{P} \subseteq \mathrm{NP}$$

$$V \leftarrow A$$

$$c \leftarrow \epsilon$$

Theorem

$$P \subseteq NP \subseteq EXP$$

$$\text{P} = \Big\{ L : L \text{ is decided by a poly. time } (O(n^k)) \text{ algorithm } A \Big\}$$

$$\text{EXP} = \Big\{ L : L \text{ is decided by an exp. time } (O(2^{n^k})) \text{ algorithm } A \Big\}$$

Proof.

$\text{P} \subseteq \text{NP}$

$V \leftarrow A$

$c \leftarrow \epsilon$

$\text{NP} \subseteq \text{EXP}$

Theorem

$$P \subseteq NP \subseteq EXP$$

$$\text{P} = \Big\{ L : L \text{ is decided by a poly. time } (O(n^k)) \text{ algorithm } A \Big\}$$

$$\text{EXP} = \Big\{ L : L \text{ is decided by an exp. time } (O(2^{n^k})) \text{ algorithm } A \Big\}$$

Proof.

$\text{P} \subseteq \text{NP}$

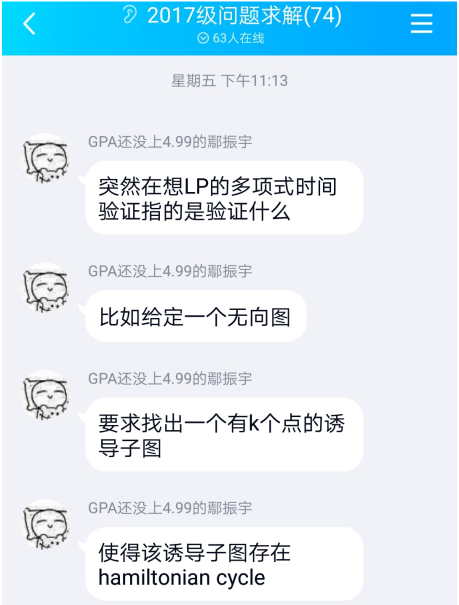
$V \leftarrow A$

$c \leftarrow \epsilon$

$\text{NP} \subseteq \text{EXP}$

Enumerate all possible c's
$(\# = 2^{O(|x|^k)})$

□


2017级问题求解(74)
63人在线
星期五 下午11:13
GPA还没上4.99的鄢振宇
突然在想LP的多项式时间验证指的是验证什么
GPA还没上4.99的鄢振宇
比如给定一个无向图
GPA还没上4.99的鄢振宇
要求找出一个有k个点的诱导子图
GPA还没上4.99的鄢振宇
使得该诱导子图存在hamiltonian cycle

Definition (HC-SUBGRAPH)

INSTANCE: Graph $G = (V, E)$, $k \in \mathbb{N}$

QUESTION: Is there a V'-induced subgraph $G[V']$ of G with $|V'| \geq k$ which is Hamiltonian?

Definition (HC-SUBGRAPH)

INSTANCE: Graph $G = (V, E)$, $k \in \mathbb{N}$

QUESTION: Is there a V'-induced subgraph $G[V']$ of G with $|V'| \geq k$ which is Hamiltonian?

Q : HC-SUBGRAPH $\in$ NP?

Definition (HC-SUBGRAPH)

INSTANCE: Graph $G = (V, E)$, $k \in \mathbb{N}$

QUESTION: Is there a V'-induced subgraph $G[V']$ of G with $|V'| \geq k$ which is Hamiltonian?

Q : HC-SUBGRAPH $\in$ NP?

c : V' in HC order

Definition (HC-SUBGRAPH)

INSTANCE: Graph $G = (V, E)$, $k \in \mathbb{N}$

QUESTION: Is there a V'-induced subgraph $G[V']$ of G with $|V'| \geq k$ which is Hamiltonian?

Q : HC-SUBGRAPH $\in$ NP?

c : V' in HC order

Q : HC-SUBGRAPH $\in$ NP-complete?

Definition (HC-SUBGRAPH)

INSTANCE: Graph $G = (V, E)$, $k \in \mathbb{N}$

QUESTION: Is there a V'-induced subgraph $G[V']$ of G with $|V'| \geq k$ which is Hamiltonian?

Q : HC-SUBGRAPH $\in$ NP?

c : V' in HC order

Q : HC-SUBGRAPH $\in$ NP-complete?

HAM-CYCLE $\leq_p$ HC-SUBGRAPH

Closure of NP (CLRS 34.2-4)

NP is closed under $\cup, \cap, \cdot, \star$.

$$L_1 \in \text{NP}, L_2 \in \text{NP} \implies L = L_1 \circ L_2 \in \text{NP}$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cup L_2 \in \mathrm{NP}$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cup L_2 \in \mathrm{NP}$$

```
procedure V(x, c)
    if c ≠ c_1#c_2 then
        return 0

    return V(x, c_1) ∨ V(x, c_2)
```

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cup L_2 \in \mathrm{NP}$$

procedure $\mathrm{V}(x, c)$
    if $c \neq c_1 \# c_2$ **then**
        return 0

    return $V(x, c_1) \lor V(x, c_2)$

$$x \in L_1 \cup L_2 \iff \exists c, V(x, c) = 1$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cap L_2 \in \mathrm{NP}$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cap L_2 \in \mathrm{NP}$$

```
procedure V(x, c)
    if c ≠ c_1#c_2 then
        return 0

    return V(x, c_1) ∧ V(x, c_2)
```

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cap L_2 \in \mathrm{NP}$$

```
1: procedure V(x, c)
2:     if c ≠ c1#c2 then
3:         return 0
4:     return V(x, c1) ∧ V(x, c2)
```

$$x \in L_1 \cap L_2 \iff \exists c, V(x, c) = 1$$

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cdot L_2 \in \mathrm{NP}$$

$$L_1 \in \text{NP}, L_2 \in \text{NP} \implies L = L_1 \cdot L_2 \in \text{NP}$$

```
procedure V(x, c)
    if c ≠ c_1#c_2&m then
        return 0

    return V(x_{1...m}, c_1) ∧ V(x_{m+1...|x|}, c_2)
```

$$L_1 \in \mathrm{NP}, L_2 \in \mathrm{NP} \implies L = L_1 \cdot L_2 \in \mathrm{NP}$$

procedure V(x, c)
    if $c \neq c_1 \# c_2 \& m$ **then**
        return 0

    return $V(x_{1 \ldots m}, c_1) \land V(x_{m+1 \ldots |x|}, c_2)$

$$x \in L_1 \cdot L_2 \iff \exists c, V(x, c) = 1$$

$$L \in \mathrm{NP} \implies L^{\star} \in \mathrm{NP}$$

$$L \in \mathrm{NP} \implies L^{\star} \in \mathrm{NP}$$

procedure $\mathrm{V}(x, c)$

  for $k \leftarrow 1$ **to** $|x|$ **do**

    $m_0 \leftarrow 0, m_k \leftarrow |x|$

    if $c = c_1 \# c_2 \# \cdots \# c_k \& m_1 \& m_2 \& \cdots \& m_{k-1}$ **then**

      return $\bigwedge_{i=1}^{i=k} V(x_{m_{i-1}+1 \ldots m_i}, c_i)$

$$L \in \mathrm{NP} \implies L^{\star} \in \mathrm{NP}$$

procedure $\mathrm{V}(x, c)$
    for $k \leftarrow 1$ **to** $|x|$ **do**
        $m_0 \leftarrow 0, m_k \leftarrow |x|$
        if $c = c_1\#c_2\#\cdots\#c_k\&m_1\&m_2\&\cdots\&m_{k-1}$ **then**
            return $\bigwedge_{i=1}^{i=k} V(x_{m_{i-1}+1 \ldots m_i}, c_i)$

$$x \in L^{\star} \iff \exists c, A(x, c) = 1$$

$$\mathrm{coNP} = \left\{ L : \overline{L} \in \mathrm{NP} \right\}$$

$$\mathrm{coNP} = \left\{ L : \overline{L} \in \mathrm{NP} \right\}$$

$$\mathrm{UNSAT} = \left\{ \varphi : \varphi \text{ is unsatisfiable.} \right\}$$

$$\text{coNP} = \left\{ L : \overline{L} \in \text{NP} \right\}$$

$$\text{UNSAT} = \left\{ \varphi : \varphi \text{ is unsatisfiable.} \right\}$$

Definition (coNP)

$$L \in \text{coNP}$$

$$\iff$$

$\exists$ poly. time *verifier* $V(x, u)$ such that

$$\text{coNP} = \left\{ L : \overline{L} \in \text{NP} \right\}$$

$$\text{UNSAT} = \left\{ \varphi : \varphi \text{ is unsatisfiable.} \right\}$$

Definition (coNP)

$$L \in \text{coNP}$$

$$\Longleftrightarrow$$

$$\exists \text{ poly. time } \textit{verifier } V(x,u) \text{ such that}$$

$$\forall x \in \{0,1\}^* : x \in L \iff \forall u \text{ with } |u| = O(|x|^k), V(x,u) = 1.$$

$$\text{coNP} \neq \{0,1\}^* \setminus \text{NP}$$

$$\text{coNP} \neq \{0,1\}^* \setminus \text{NP}$$

$$\text{P} \subseteq \text{NP} \cap \text{coNP}$$

$$\text{coNP} \neq \{0,1\}^* \setminus \text{NP}$$

$$\text{P} \subseteq \text{NP} \cap \text{coNP}$$

$$\text{P} = \text{NP} \implies \text{NP} = \text{coNP}$$

$$\text{coNP} \neq \{0,1\}^* \setminus \text{NP}$$

$$\text{P} \subseteq \text{NP} \cap \text{coNP}$$

$$\text{P} = \text{NP} \implies \text{NP} = \text{coNP}$$

Unsolved problem in computer science:

? $\text{NP} \stackrel{?}{=} \text{co-NP}$

(more unsolved problems in computer science)

$$\mathrm{coNP} \neq \{0,1\}^* \setminus \mathrm{NP}$$

$$\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{coNP}$$

$$\mathrm{P} = \mathrm{NP} \implies \mathrm{NP} = \mathrm{coNP}$$

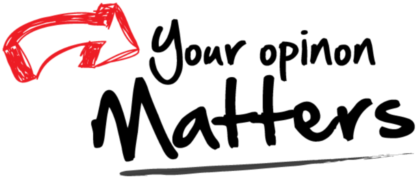
Unsolved problem in computer science:

? $\mathrm{NP} \stackrel{?}{=} \text{co-NP}$

(more unsolved problems in computer science)

$$\mathrm{NP} \neq \mathrm{coNP} \stackrel{?}{\Rightarrow} \mathrm{P} \neq \mathrm{NP}$$


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn