

Kiszámíthatóság elmélet – Egy kis történelem

- 1900: **Hilbert 23 problémája**
 - Pl. 10-es: keressük meg egy tetszőleges egész együtthatós többváltozós egyenlet gyökeit
 - Pl. bemenet: $3xy - 2x^2 + 2z + 4 = 0$, kimenet: $x = 0, y = 0, z = -2$
- 1920-as évek: **Hilbert programja**
 - Célja a matematika összes elméletének axiomatizálása egy végesen reprezentálható axiómarendszerrel
- 1929, Gödel: Az **elsőrendű kalkulus teljes** (minden tautológia levezethető egy megfelelő axiómarendszerből)

Kiszámíthatóság elmélet – Egy kis történelem

- 1931, Gödel első **nemteljességi tétele**
 - Minden olyan effektíven kiszámítható elmélet ami tartalmazza az **elemi aritmetikát** nem lehet egyszerre helyes és teljes
 - **Következmény:** Hilbert programja megvalósíthatatlan
- 1930-as évek: különböző algoritmus modellek bevezetése
 - Gödel: **rekurzív függvények**
 - Church, Kleene, Rosser: **λ kalkulus**
 - Alan Turing: **Turing-gép**
- **Church-Turing-tézis:** a kiszámíthatóság különböző matematikai modelljei mind az effektíven kiszámítható függvények osztályát definiálják
 - **Megjegyzés:** A 10-es probléma algoritmikusan eldönthetetlen (1970, Matiyasevich)

Kiszámíthatóság elmélet – Alapfogalmak

- **Kiszámítási probléma:** olyan matematikailag precízen megfogalmazott probléma, amit **algoritmikusan szeretnénk megoldani**
 - Pl. Hilbert 10-es problémája
- Egy **hétköznapi problémához**, megfelelő absztrakcióval, általában megadható **ekvivalens kiszámítási probléma**
 - Pl. Utazó ügynök \rightarrow Súlyozott Hamilton kör
 - Hordók pakolása a teherautóra \rightarrow Körpakolás

Kiszámíthatóság elmélet – Alapfogalmak

- **Eldöntési probléma:** Egy I bemenetre a válasz *igen* vagy *nem* (rendre, I pozitív vagy negatív bemenet)
 - **Példa:** SAT probléma
- P probléma **megoldható**, ha minden I bemenetre az I -re adott P szerinti válasz algoritmikusan kiszámítható
- A megoldható eldöntési problémákat **eldönthetőnek** nevezzük
 - **Példa:** SAT eldönthető

Kiszámíthatóság elmélet – Alapfogalmak

- **Jelölés:** Tetszőleges D objektumra $\langle D \rangle$ jelöli a D **elkódolását** egy megfelelő ábécé felett
- Egy P eldöntési probléma **megfeleltethető** egy

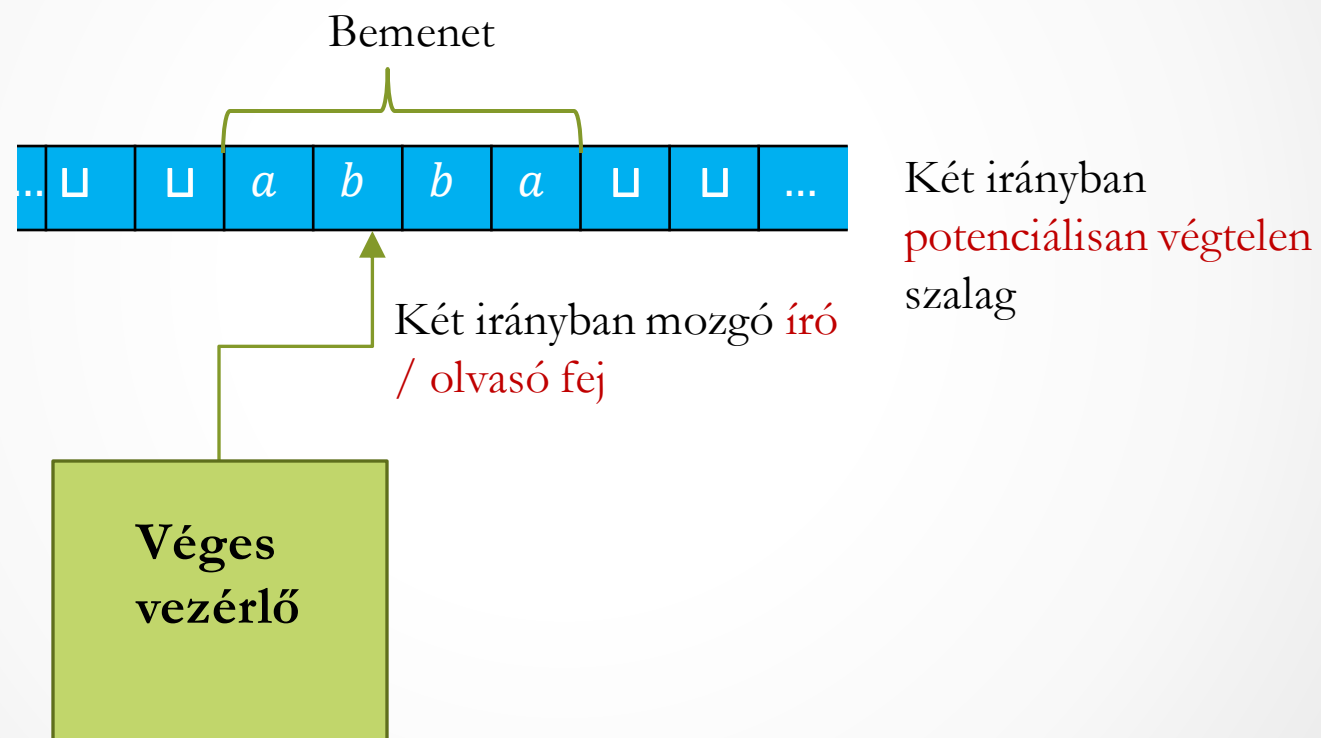
$$L_P := \{ \langle I \rangle \mid I \text{ a } P \text{ pozitív bemenete} \}$$

formális nyelvnek

- **Példa:** L_{SAT} (vagy SAT) $:= \{ \langle F \rangle \mid F \text{ kielégíthető} \}$

A Turing- gép

- Egy klasszikus **algoritmus modell** problémák eldöntésére

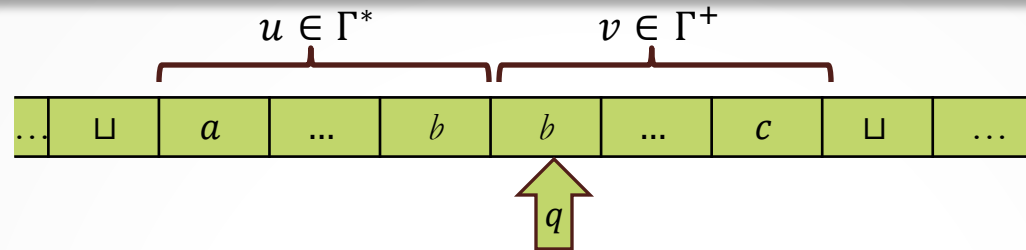


A Turing gép

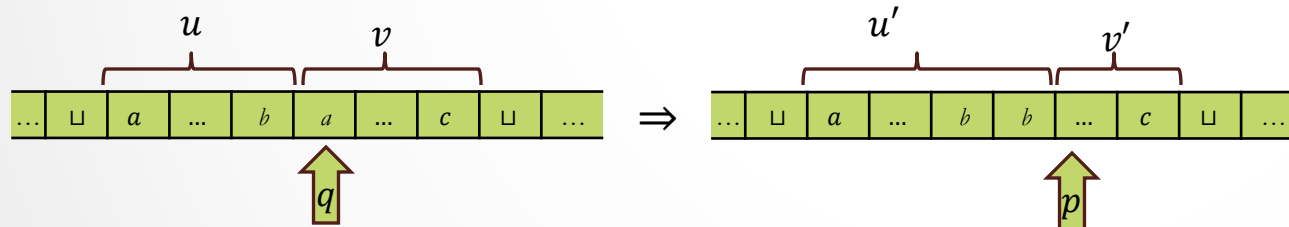
- **Formálisan:** $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$, ahol
 - Q az **állapotok** véges, nem üres halmaza, q_0 : **kezdő**-, q_i : **elfogadó**, q_n : **elutasító** állapot
 - Σ a **bemenő jelek** ábécéje
 - Γ a **szalagszimbólumok** ábécéje
($\Sigma \subset \Gamma$, $\sqcup \in \Gamma - \Sigma$, \sqcup : **üres szimbólum**)
 - $\delta: (Q - \{q_i, q_n\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ az **átmeneti függvény**
- M működését egy adott szón konfigurációk sorozatával írhatjuk le

A Turing gép működése

- Konfiguráció:



- Azaz, a szalagon csak az uv szó van (az azon kívüli rész csak \sqcup -t tartalmaz)
 - a fej a v első betűjére mutat és az M a q állapotban van
 - Tömörebben leírva: uqv
- M a működése során **konfigurációátmeneteket** (röviden: **lépéseket**) hajt végre
 - Ez mindig „kompatibilis” a δ -val, pl.



akkor és csak akkor ha $\delta(q, a) = (p, b, r)$

A Turing gép működése

- M **elfogad** egy $w \in \Sigma^*$ szót,
 - ha a w -t tartalmazó **kezdőkonfigurációból**
 - (azaz amikor M q_0 -ban van, a szalagon csak w van és M feje a w első betűjére ($w = \varepsilon$ esetén egy \sqcup -re) mutat)
 - konfigurációátmenetek sorozatát végrehajtva
 - egy **elfogadó** (azaz q_i -t tartalmazó) **konfigurációba** jut
- Az M által felismert nyelv (jelölése: $L(M)$):
 - azon Σ^* -beli szavak halmaza melyeket M elfogad

Turing-felismerhető nyelvek

- Egy L nyelv **Turing-felismerhető**, ha van olyan M Turing gép, hogy $L = L(M)$. Ha M olyan, hogy minden bemeneten meg is áll, akkor L **eldönthető**
- Megjegyzés:
 - A Turing-felismerhető nyelveket nevezzük még **rekurzívan felsorolható** nyelveknek; ezen nyelvek osztályát **RE** -vel jelöljük
 - Az eldönthető nyelveket nevezzük még **rekurzív** nyelveknek is; ezen nyelvek osztályát **R** -rel jelöljük
 - Triviálisan $R \subseteq RE$
- Turing-gépek **kiterjesztései**:
 - **többszalagos** Turing-gép
 - **nemdeterminisztikus** Turing-gép (lásd később)

Turing-felismerhető nyelvek

- Turing-felismerhető, de nem eldönthető problémák:
 - Hilbert 10-es problémája
 - Egy elsőrendű formula kielégíthetetlen-e?
 - Egy M Turing-gép elfogad-e legalább egy szót? Formálisan: $L = \{ \langle M \rangle \mid L(M) \neq \emptyset \}$
- Nem Turing-felismerhető problémák:
 - Egy elsőrendű formula kielégíthető-e?
 - Igaz-e az, hogy egy M Turing-gép nem fogad el egyetlen szót sem? Formálisan: $L = \{ \langle M \rangle \mid L(M) = \emptyset \}$
- Általánosan: minden $L \in RE - R$ nyelvre, $\bar{L} \notin RE$

Turing-gépek időigénye

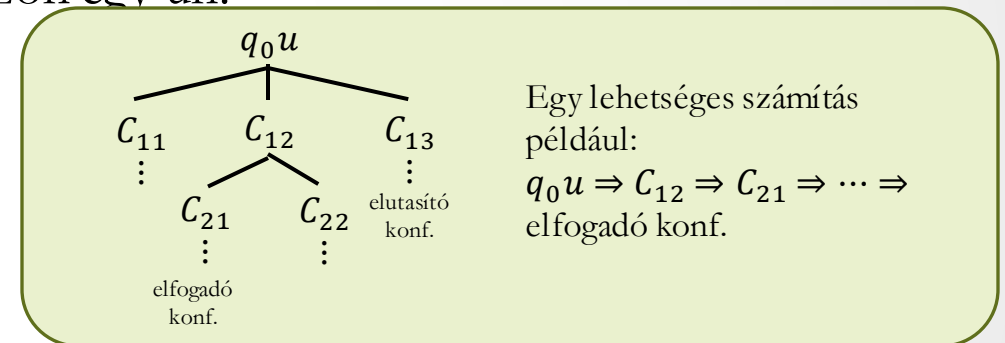
- Az M Turing-gép **időigénye** $f: \mathbb{N} \rightarrow \mathbb{N}$, ha minden n hosszú bemeneten M legfeljebb $f(n)$ lépésben megáll
- **Megjegyzés:** nem kell a pontos időigényt megadni!
- Használhatjuk a nagy O jelölést:
 - $f, g: \mathbb{N} \rightarrow \mathbb{R}^+$, az **f legfeljebb olyan gyorsan nő mint a g** (jelölés: $f = O(g)$), ha $\exists n_0 \in \mathbb{N}, c > 0: \forall n \geq n_0: f(n) \leq c \cdot g(n)$
 - Például, $n^{1000} = O\left(2^{\frac{n}{1000}}\right)$, $\log n = O(n)$ (de fordítva nem igaz egyik esetben sem)

Turing-gépek időigénye

- Milyen időigényű Turing-géppel dönthető el a **SAT probléma**?
 - Legyen F egy m karakterből álló, n változót tartalmazó KNF-ben lévő formula
 - Legyen M a következő Turing-gép
 - M a második szalagján legenerálja az **összes lehetséges** interpretációt, ami $O(n \cdot 2^n)$ lépés
 - Adott interpretációra **leellenőrzi**, hogy az kielégíti-e F -et, ami $O(m)$ lépés
 - Tehát M időigénye $O(m \cdot n \cdot 2^n)$ lépés
 - Azaz, ha nincs túl sok klóz F ben, a futási idő **exponenciális** (a klózik száma max. 3^n ; tehát ha nagyságrendileg ennyi klóz van, akkor akár lineáris is lehet a futási idő)
- Nem ismert (a legrosszabb esetet figyelembe véve) ennél hatékonyabb algoritmus ¹³

A Nemdeterminisztikus Turing-gép

- A Nemdeterminisztikus Turing-gép komponensei megegyeznek a determinisztikus Turing-gépével, kivéve az **átmeneti függvényt**:
 - $\delta: (Q - \{q_i, q_n\}) \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R, S\})$ (azaz δ most a $Q \times \Gamma \times \{L, R, S\}$ részhalmazainak halmazába képez)
- Egy M nemdet. Turing-gép lehetséges számításai egy u szón egy ún. **számítási (konfigurációs) fával** szemléltethetők:



- M **elfogadja** az u -t ha a számítási fa legalább egy levele elfogadó konfiguráció
- A felismert nyelv definíciója ugyanaz, mint a determinisztikus esetben

A Nemdeterminisztikus Turing-gép

- Az M nemdeterminisztikus Turing-gép **eldönti** az $L \subseteq \Sigma^*$ nyelvet, ha felismeri és minden $u \in \Sigma^*$ szóra az M számítási fája véges és minden levele elfogadó vagy elutasító konfiguráció
- M **$f(n)$ időigényű**, ha minden $u \in \Sigma^*$ n hosszú szóra a számítási fa legfeljebb **$f(n)$ magas**
- **Tétel:** M -hez megadható egy vele **ekvivalens** (ugyanazt a nyelvet felismerő) determinisztikus T-gép
 - A bizonyítás konstruktív de a kapott det. T-gép időigény romlása **exponenciális**
 - **nem ismert** jobb konstrukció

A Nemdeterminisztikus Turing-gép

- Milyen időigénnyel lehet eldönteni a SAT problémát **nemdeterminisztikus** Turing-géppel?
 - Legyen F egy m karakterből álló, n változót tartalmazó KNF-ben lévő formula
 - Egy M nemdeterminisztikus Turing-gép a második szalagján **nemdeterminisztikusan** előállít **egy lehetséges interpretációt** (M képes az összes lehetséges interpretációt előállítani ily módon), ami $O(n)$ lépés
 - Erre az interpretációra **leellenőrzi**, hogy az kielégíti-e F -et, ami $O(m)$ lépés
 - Ez $O(n + m) = O(m)$ lépés (nyilván $n \leq m$, azaz legfeljebb annyi változó van F -ben, mint ami a hossza)
 - Azaz M időigénye **lineáris** függvénye a formula méretének (hosszának)
 - **Megjegyzés:** az M felfogható úgy, hogy párhuzamosan futtat 2^n számítást, melyek leellenőrzik az összes lehetséges interpretációt

A hatékony visszavezetésekéről

- Egy A probléma **hatékonyan visszavezethető** egy B problémára, ha van egy olyan M nagyon **hatékonyan** működő, pl. **logaritmikus táras** Turing-gép ami
 - az A egy I bemenetét kódoló $\langle I \rangle$ szóból **kiszámítja** a B egy I' bemenetét kódoló $\langle I' \rangle$ szót úgy, hogy a következő teljesül:
 - I akkor és csak akkor **pozitív bemenete** a A -nak ha I' **pozitív bemenete** a B -nek
- Ekkor ha van B -t hatékonyan megoldó algoritmus, akkor van ilyen A -ra is:
 - Tegyük fel, hogy el akarjuk dönteni, hogy az A egy I bemenete pozitív bemenet vagy sem
 - Konstruáljuk meg $\langle I' \rangle$ -t az M Turing-géppel
 - Döntsük el a B -t eldöntő algoritmussal, hogy I' pozitív bemenete-e B -nek vagy sem

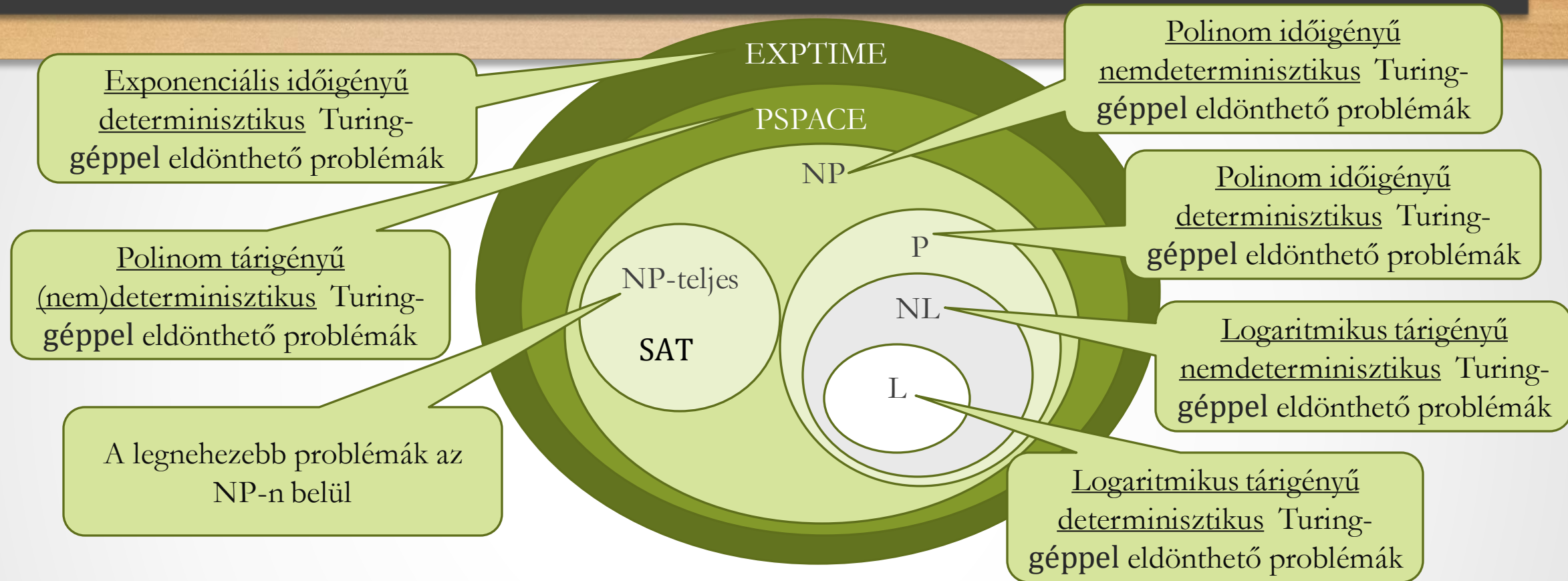
NP-teljes problémák

- Legyen C egy bonyolultsági osztály
 - **C-teljes problémák:** azon C -beli problémák, melyek melyekre minden további C -beli probléma **hatékonyan visszavezethető**
 - Ezek a problémák a **legnehezebben megoldhatók** a C -n belül
- **Tétel:** Ha egy NP-teljes problémáról kiderülne, hogy P -beli, akkor **$P = NP$** is igaz lenne
- Következik, hogy az NP-teljes problémák feltehetően **nincsenek P -ben**, azaz az NP-teljes problémákat feltehetően nem lehet polinom időben (hatékonyan) megoldani
- **Tétel:** SAT NP-teljes (Cook tétele)
- **Következmény:** SAT-ra valószínűleg nincs polinom idejű algoritmus

Emlékeztető:

P és NP rendre a determinisztikus és nemdeterminisztikus Turing-gépekkel polinom időben megoldható problémák osztályai

A klasszikus időbonyolultsági osztályok viszonyai (sejtés)



Megjegyzés: Az a sejtés, hogy az **összes tartalmazás valódi**, de egyelőre csak az bizonyított, hogy $P \subsetneq EXPTIME$

Klasszikus Időbonyolultsági osztályok

- Nehezen megoldható problémáknak néha érdemes a **megszorításait** vizsgálni
- $k \geq 1$, **k SAT**: Adott egy olyan F KNF, melyben minden tag pontosan k literált tartalmaz. Kérdés: **Kielégíthető-e F** ?
 - SAT, 3SAT: **NP-teljes**, azaz nehezen megoldhatók
 - 2SAT: **NL-teljes**, azaz már polinom időben megoldható
- **HORNSAT**: A Horn-formulák (olyan KNF amiben minden klóz legfeljebb egy pozitív literált tartalmaz) kielégíthetőségének eldöntése
 - HORNSAT: **P-teljes**

Klasszikus Időbonyolultsági osztályok

- Néha a **gyengébb keresési feltételekkel** megfogalmazott problémák egyszerűbben megoldhatók:
- **HAMILTON-ÚT**: Adott egy G irányított gráf és ennek s, t csúcsai. Kérdés: Van-e G -ben s -ből t -be vezető Hamilton-út?
 - HAMILTON-ÚT **NP-teljes**, azaz nehezen megoldható a probléma
- Ha csak tetszőleges utat keresünk s -ből t -be: **NL-teljes**, azaz polinom időben megoldható a probléma
- Ha G ráadásul irányítatlan: **L-teljes**, azaz akár log-tárral is megoldható a probléma