

Számítástudomány

GAZDAG ZSOLT

SZTE, SZÁMÍTÁSTUDOMÁNY ALAPJAI TANSZÉK

2025. TAVASZ

A kurzusról

Előadó: Gazdag Zsolt

Elérhetőség: Irinyi épület, III. lh. 67. szoba

Email: gazdag@inf.u-szeged.hu

A kurzus teljesítésének feltételei:

- Sikeres gyakorlat
- Sikeres géptermi Coospace teszt
 - kifejtős és teszt kérdések
 - alap összefüggések, definíciók, állítások, és ezek alapján kikövetkeztethető egyszerű kérdések és feladatok
 - a teszt kérdéseknél rossz válaszáért pontlevonás
- sikeres a teszt ha legalább 40%-os; ha a teszt sikertelen, akkor a vizsga érdemjegye elégtelen
- A sikeres tesztet megírók megajánlott jegyet kapnak az alábbiak szerint:
 - 40%-60% - elégséges (2)
 - 60%-80% - közepes (3)
 - 80%-100% - jó (4)
- A sikeres tesztet megírók jelentkezhetnek szóbeli vizsgára, amennyiben a megajánlott jegynél jobbat szeretnének.

Hogyan kell olvasni a fóliákat?

Az egyes részek elkülönítettek:

Fogalmak, definíciók, jelölések

Tételek, segédtételek (lemmák) és bizonyításaik

Példák

Megjegyzések

Hogyan kell olvasni a fóliákat?

Azon részek, melyek nagyobb valószínűséggel előjöhetnek a Coospace tesztben folyamatos keretben lesznek, pl. így:

Ábécé: tetszőleges véges, nem üres halmaz
Betűk: tetszőleges ábécé elemei

A többi meg szaggatott keretben, pl így:

Bizonyítás

Egyesítésre és metszetre való zártság

- Legyen L_i ($i = 1, 2$) felismerhető nyelv és $M_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$, úgy hogy $L_i = L(M_i)$
- Ötlet: megadunk egy M' -t, ami egyszerre szimulálja M_1 és M_2 számításait
- ...

Figyelem: főleg a kifejtős kérdésekben lehet olyan rész, amihez a szaggatott keretben lévő ismeretek szükségesek

Miről lesz szó? Véges automatak és reguláris kifejezések

Két különböző modell ugyanakkora kifejezőerővel

- **Véges automata:**
 - **lexikális** elemzők,
 - egyszerű gépek, protokollok modelljei
 - mintaillesztés, (formális) nyelvfelismerés
- **Reguláris kifejezés:**
 - Formalizmus hasonló tulajdonságú szavak jelölésére
- A két modell algoritmikusan egymásba **konvertálható**: azt használjuk, ami kényelmesebb

Véges automata mint egyszerű gép – Példa

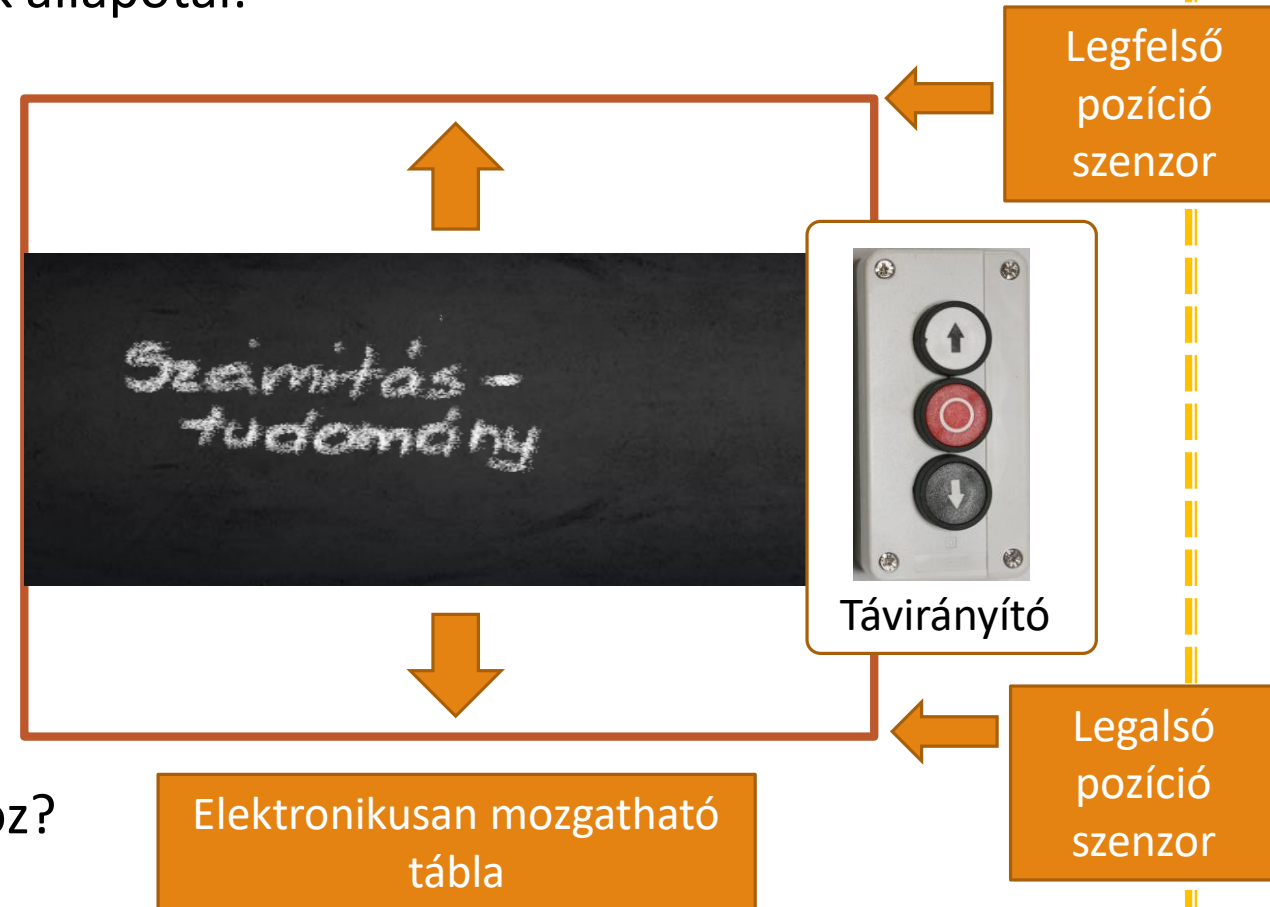
- Egy előadóterem mozgatható táblájának állapotai:

- Fent van (\wedge)
- Lent van (\vee)
- A kettő között áll (\ominus)
- Felfele megy (\Uparrow)
- Lefele megy (\Downarrow)

- Milyen jelek befolyásolják az állapotát?

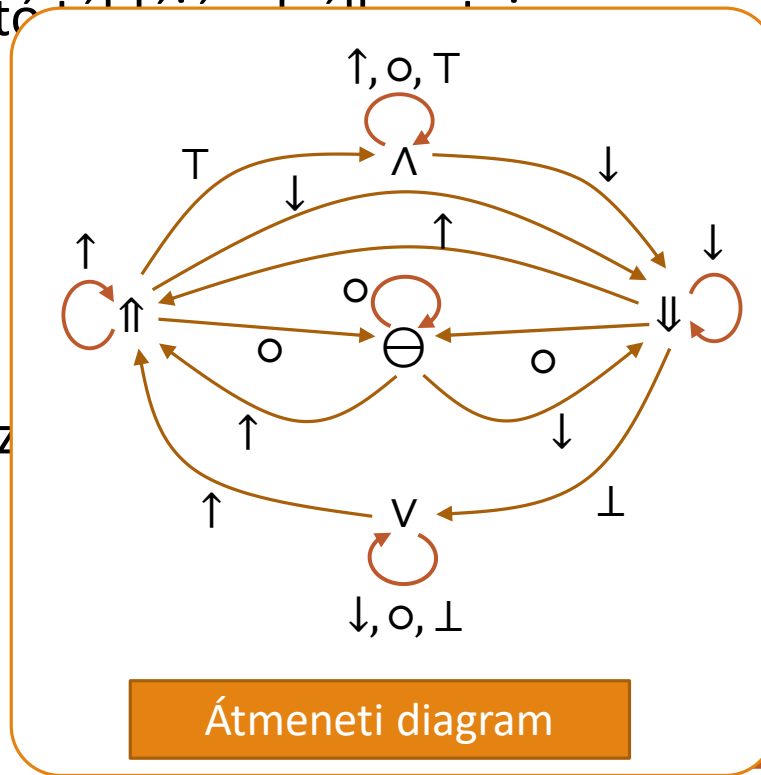
- Felfele gomb (\Uparrow)
- Lefele gomb (\Downarrow)
- Állj gomb (\circ)
- Fent van szenzor (\top)
- Lent van szenzor (\perp)

- És hogyan kapcsolódnak ezek egymáshoz?



Véges automata mint egyszerű gép – Példa

- Egy előadóterem mozgatható...
- Fent van (\wedge)
- Lent van (\vee)
- A kettő között áll (\ominus)
- Felfele megy (\Uparrow)
- Lefele megy (\Downarrow)
- Milyen jelek befolyásolják az...
- Felfele gomb (\uparrow)
- Lefele gomb (\downarrow)
- Állj gomb (\circ)
- Fent van szenzor (T)
- Lent van szenzor (\perp)
- És hogyan kapcsolódnak ezek egymáshoz?



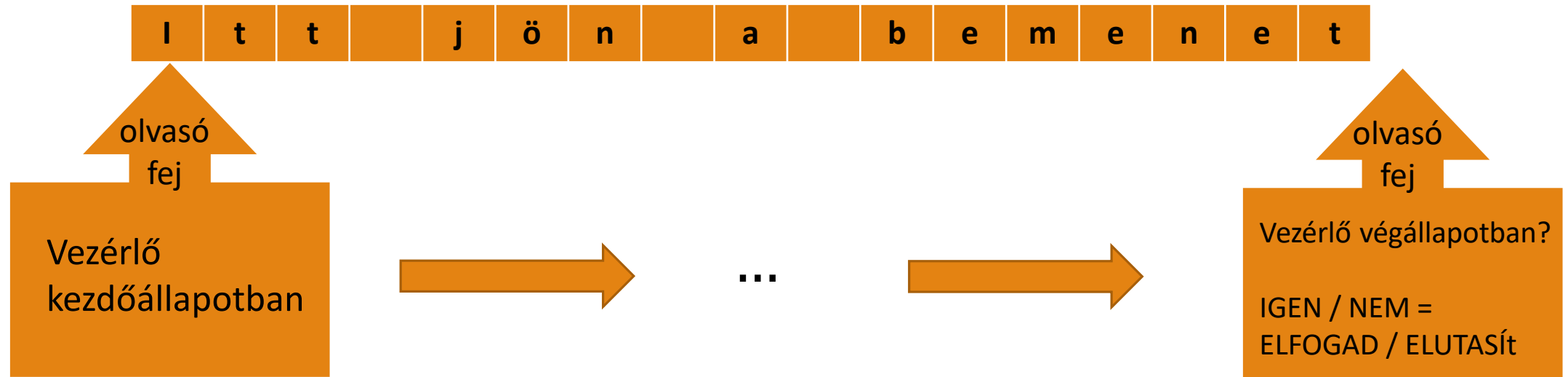
Átmeneti diagram

Elektronikusan mozgatható
tábla



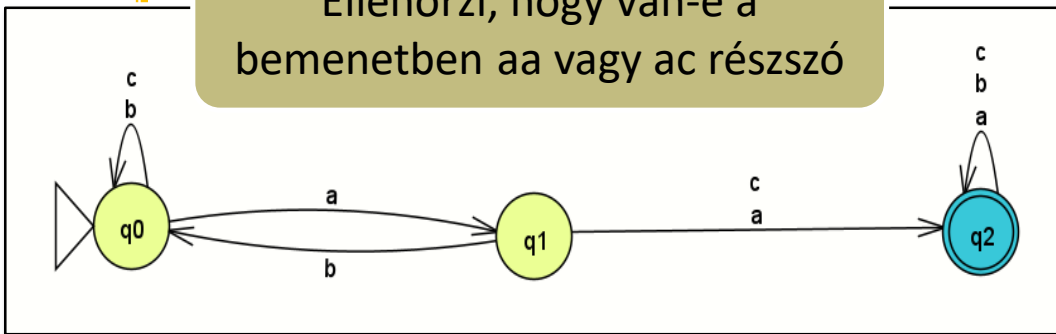
Véges automata mint szavak felismerője

- Ilyenkor arra vagyunk kíváncsiak, hogy mely jelsorozatok „sikeresek”
- Ekkor a véges automata rendelkezik
 - egy olvasófejjel (balról jobbra mozog) és egy input szalaggal amit ott a bemenő szó
 - egy kezdő- és legalább egy végállapottal
- A véges automata pontosan akkor fogadja el a bemeneten kapott szót ha
 - a szó első betűjétől kezdve balról jobbra végig olvasva a bemenetet
 - végállapotban áll meg



Véges automata mint mintaillesztő vs. reguláris kifejezés - Példa

Ellenőrzi, hogy van-e a bemenetben aa vagy ac részszó

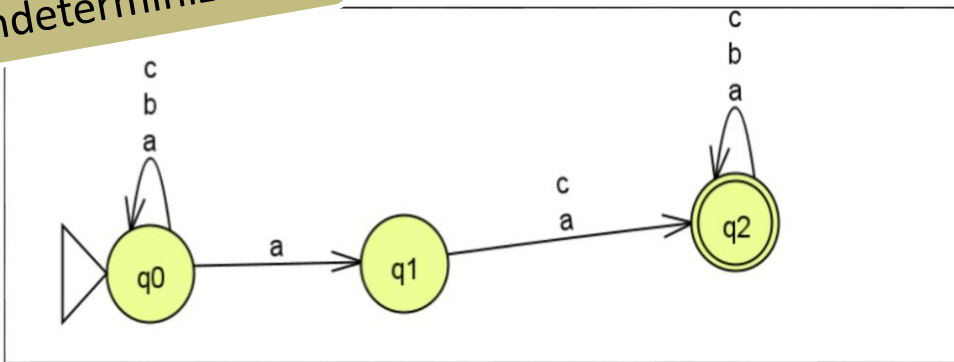


Edit the regular expression below:

$(b+c+ab)^*a(a+c)(a+b+c)^*$

Input Field Text Size (For optimization, adjust the size of this window after resizing the text field)

Nemdeterminizmus!



Edit the regular expression below:

$(a+b+c)^*a(a+c)(a+b+c)^*$

Input Field Text Size (For optimization, adjust the size of this window after resizing the text field)

Miről lesz még szó? Környezetfüggetlen nyelvtanok és veremautomaták

Két különböző modell ugyanakkora kifejezőerővel

- **Környezetfüggetlen nyelvtan:**

- formalizmus programozási nyelvek **szintaxisának** megadására
- a Backus-Naur forma (BNF) formális nyelvi megfelelője

Egyszerű kifejezések definíciója BNF-fel:

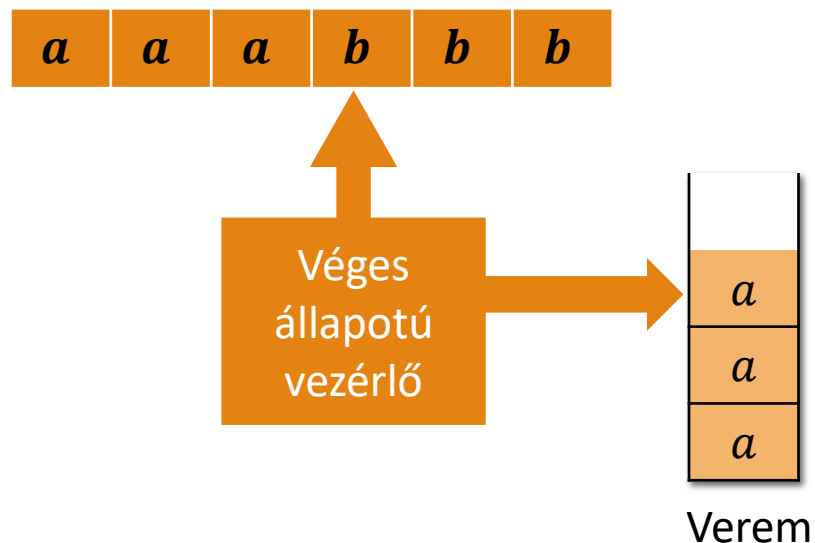
```
<kifejezés> ::= <kifejezés> "+" <kifejezés> | <kifejezés> "*" <kifejezés> | "("<kifejezés>")" | <konstans>  
<konstans> ::= "a" | "b" | "c"
```

- **Veremautomata:** véges automata kiegészítve egy veremmel

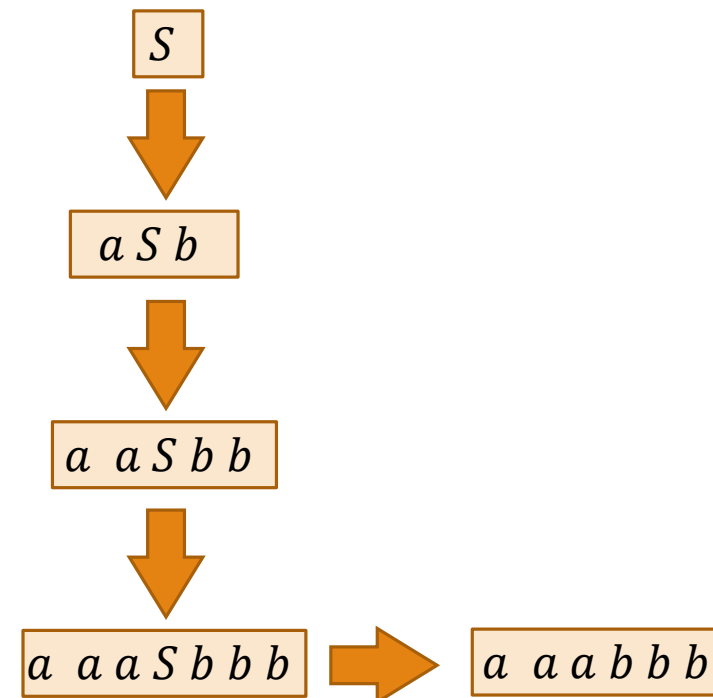
- **Elemzők** (parserek) formális nyelvi modelljei
- Szavak, formális nyelvek felismerése
- A két modell algoritmikusan egymásba **konvertálható**: azt használjuk, ami kényelmesebb

Veremautomata vs. környezetfüggetlen nyelvtan - Példa

Veremautomata: ellenőrzi, hogy ugyanannyi a van-e a bemenet elején, mint amennyi b jön végül



környezetfüggetlen nyelvtan: egy kezdőszimbólumból (S) kiindulva generálja az összes jó alakú szót



Miről lesz még szó? Kiszámíthatóság elmélet

Eldöntési problémákkal foglalkozik

Akkor nevezünk egy problémát eldöntési problémának, ha a bemenetre a válasz „igen” v „nem”

Például az **ELÉRHETŐSÉG** probléma:
Bemenet: egy $G = (V, E)$ irányított gráf, ahol $V = \{1, \dots, n\}$
Kérdés: Vajon van-e út G -ben 1-ből n -be?

- Azt vizsgálja, hogy egyes problémák **eldönthetők** (vagy legalább **félíg eldönthetők**) vagy sem
- Hogyan lehet bebizonyítani, hogy egy probléma nem eldönthető?
 - Kell egy matematikai eszköz amiről mindenki „elfogadja” hogy amit algoritmikusan el lehet dönteni, azt ezzel az eszközzel is el lehet
 - Ilyen eszköz számos van, például:
 - **Turing-gép** (a véges automaták, veremautomaták általánosítása), Alan Turing 1936
 - **λ -kalkulus** (függvények absztrakcióján és alkalmazásán alapuló formális rendszer), A. Church 1936
 - **RAM gép** (egyszerű programozási nyelv, kb. mint a gépi kód)
 - **Általános nyelvtan** (formális nyelvek, a Chomsky-hierarchia legerősebb nyelvtana, lásd később), Chomsky 1956

Church-Turing tézis: a kiszámíthatóság fenti modelljei mind az effektíven (algoritmikusan) kiszámítható problémák osztályát definiálják (tézis és nem tétel, mert az, hogy „effektíven kiszámítható” egy intuitív fogalom)

Eldönthetetlenség

A Church-Turing tézis alapján ha egy probléma **nem dönthető el** a fenti kiszámítási modellek egyikével, akkor nem dönthető el semmilyen algoritmussal sem

Az **első problémák**, melyek eldönthetetlensége kiderült:

- A Turing-gépek **megállási problémája**
- λ -kalkulusbeli kifejezések ekvivalenciája

Az eldönthetetlenség bizonyítható **visszavezetéssel** is:

- Legyen K és L két eldöntési probléma, és tegyük fel, hogy K eldönthetetlen
- Ha van olyan A algoritmus, ami a K bemenetéből kiszámolja L bemenetét úgy, hogy pozitív bemenethez pozitívat, negatívhoz negatívát rendel, akkor L is eldönthetetlen
- Például vegyük a következő problémát: adott egy F elsőrendű formula, vajon **tautológia-e** F (másképp: **érvényes-e** F)?
 - Erre a problémára visszavezethető az elsőrendű formulák **kielégíthetetlenségének** eldöntése (hogyan?)
 - Ez a probléma eldönthetetlen (a Logika kurzuson erről volt szó)
 - Így tehát az érvényesség sem lehet eldönthető

Eldöntési probléma esetén azt a bemenetet hívjuk „pozitívnak”, amire a válasz „igen”, azt pedig „negatívnak”, amire a válasz „nem”

Miről lesz még szó? Bonyolultságelmélet

Itt **csak eldönthető** problémákat vizsgálunk

Arra vagyunk kíváncsiak, hogy **milyen erőforrásigénnyel** dönthető el egy probléma

- Például az ELÉRHETŐSÉG probléma **polinom időben, polinom tárral** eldönthető
- Érdekes kérdés, hogy eldönthető-e **szublineáris** tárfelhasználással
 - Szublineáris: lineárisnál kisebb, például logaritmikus
- Az ELÉRHETŐSÉG-ről tudjuk, hogy **$\log^2 n$ tárral** is eldönthető (lásd Savith tételét később)
- Jellemző a „**time-space tradeoff**”
 - Például Savitch algoritmusának időigényére nincs polinom felső korlát

Bemenet: $G = (V, E)$ irányított gráf, ahol $V = \{1, \dots, n\}$

Legyen $S = \{1\}$

Amíg $n \notin S$ és $S \neq \emptyset$

Vegyünk ki egy u csúcsot S -ből;

Vegyük be S -be azon u -ból elérhető csúcsokat, melyek még nem szerepeltek S -ben

Ha $n \in S$, akkor **kimenet:** *igen*, egyébként **kimenet:** *nem*

Miért fontos, hogy ismerjük egy adott probléma komplexitását?

Mert például ha **felhőben** futtatjuk az algoritmust, akkor a felhasznált processzoridő és tár alapján fizetünk: <https://azure.microsoft.com/en-us/pricing/calculator/>

The screenshot shows the Azure Pricing Calculator interface. At the top, the browser address bar displays 'azure.microsoft.com/en-us/pricing/calculator/'. The page title is 'Cloud Services'. Under the 'Region:' section, 'East US' is selected. The 'Category:' dropdown is set to 'All'. The 'Instance Series:' dropdown is also set to 'All'. To the right, the 'INSTANCE:' section shows a search icon and the text 'A4: 8 Cores, 14 GB RAM, 605 GB Temporary storage, \$0.480/hour'. Below these, the 'Virtual machines' section shows a configuration of 1 instance, 730 hours, and a unit of 'Hours'. The 'Savings Options' section includes 'Pay as you go' (selected), '1 year reserved' (not available), and '3 year reserved' (not available). The 'Reservations' section has a help icon and two radio buttons for '1 year reserved' and '3 year reserved'. At the bottom, the total cost is displayed as '\$467.20' with a green equals sign, and below it, 'Average per month (\$0.00 charged upfront)'. On the right side, the same total cost '\$467.20' is shown with a green equals sign, and below it, 'Average per month'.

Cloud Services

Region:
East US

Category:
All

Instance Series:
All

INSTANCE: [\(Need help finding the right VM?\)](#)
A4: 8 Cores, 14 GB RAM, 605 GB Temporary storage, \$0.480/hour

Virtual machines
1 × 730 Hours

Savings Options

Pay as you go

1 year reserved option is not available for your instance selection.

3 year reserved option is not available for your instance selection.

☒ Pay as you go

Reservations ⓘ

☐ 1 year reserved

☐ 3 year reserved

\$467.20
Average per month
(\$0.00 charged upfront)

= \$467.20
Average per month

Hatékony visszavezetések

Tegyük fel, hogy adottak a következő ítéletkalkulusbeli formulák:

$$p, \neg p \vee \neg q, r \vee q, q \vee \neg r$$

Kérdés: **kielégíthető-e** ez a formulahalmaz?

Ahelyett, hogy értékadásokat vizsgálnánk, a következőt is tehetjük:

- Definiálunk egy G gráfot $\{p, \neg p, q, \neg q, r, \neg r\}$ csúcsokkal
- A V-kapcsolatokból implikációkat csinálunk, az implikációkat pedig éleknek tekintjük és ezeket felvesszük G -be:

p	$\neg p$
q	$\neg q$
r	$\neg r$

- A fenti formulahalmaz **akkor és csak akkor kielégíthetetlen** ha a következő teljesül:
 - Van olyan $\alpha \in \{p, \neg p, q, \neg q, r, \neg r\}$, hogy G -ben van út p -ből α -ba és α -ból az α komplementerébe

Hatékony visszavezetések

Mivel a példában van út p -ből $\neg q$ -ba, és onnan q -ba, a $p, \neg p \vee \neg q, r \vee q, q \vee \neg r$ formulahalmaz kielégíthetetlen

- A gráf csak segít követni a függőségeket: p igaz kell legyen, de akkor $\neg q$ is, amiből következik, hogy r is, és végül q is, ami ellentmondás

p	$\neg p$
q	$\neg q$
r	$\neg r$

Erről a módszerről már volt szó ma, ez is egy **visszavezetés**: a példánkban egy egyszerű formulahalmaz kielégíthetőségének eldöntését vezettük vissza az ELÉRHETŐSÉG-re

Ha **hatékony** (gyorsan, mondjuk polinom időben kiszámítható) a visszavezetés, akkor a visszavezetett probléma, általában, **legfeljebb olyan nehezen** oldható meg, mint az, amire visszavezettünk

A példában:

- A gráf megkonstruálása elvégezhető polinom időben a formulahalmaz méretében, és
- az ELÉRHETŐSÉG is megoldható polinom időben
- Ezért a fentihez hasonló formulahalmazok kielégíthetősége is eldönthető polinom időben

Nehezen megoldható problémák

A fentiekből következik: amire hatékonyan visszavezetünk nem lehet könnyebben megoldható, mint az amit visszavezetünk

Így a visszavezetést arra is felhasználhatjuk, hogy egy problémáról megmutassuk, hogy **nehezen megoldható**

A **SAT probléma**:

- Bemenet: egy ítéletkalkulusbeli konjunktív normálformában lévő formula
- Kérdés: vajon kielégíthető-e ez a formula?

A logika kurzuson tanult összes algoritmus a legrosszabb esetben **exponenciális** ideig fut erre a problémára

A SAT megoldására **nem ismert hatékony** (polinom idejű) algoritmus

- Az a sejtés, hogy ilyen nem is létezik
- Emiatt a SAT-ra „**nehezen megoldható**” (ún. NP-teljes, lásd később) problémaként fogunk hivatkozni

Ezért ha a SAT-ot hatékonyan **vissza tudjuk vezetni** egy K problémára, akkor K -ról is bizonyítjuk, hogy **nehezen megoldható**

Nehezen megoldható problémák

Vegyünk például az ELÉRHETŐSÉG következő általánosítását, a **HAMILTON-ÚT** problémát:

- Bemenet: egy $G = (V, E)$ irányított gráf, ahol $V = \{1, \dots, n\}$
- Kérdés: Vajon van-e olyan út G -ben 1-ből n -be **ami minden csúcsot pontosan egyszer érint?**

Tetszőleges F formulához **hatékonyan megkonstruálható** olyan G_F gráf, hogy

F akkor és csak akkor kielégíthető ha G_F -ben van Hamilton út adott két csúcs között






(azt hogy hogyan látni fogjuk később)

Ezért a HAMILTON-ÚT nem lehet könnyebben megoldható, mint a SAT

Vagyis a HAMILTON-ÚT is egy **nehezen megoldható** probléma (NP-teljes)

De miért baj az, hogy ha valamire csak exponenciális időigényű algoritmus létezik?

Az alábbi táblázat megadja, hogy adott futásidejű algoritmus adott számítási kapacitású architektúrán mekkora inputra fut még le **egy napon belül**.

	 C64 1Mflops	 Cray Y-MP 1Gflops	 mai GPU 5TFlops	 Tianhe-2 34PFlops	 Föld bolygó 10EFlops
n	86.4G	8.64×10^{13}	4.32×10^{17}	2.94×10^{21}	8.64×10^{23}
$n \log n$	2.75G	2.1×10^{12}	8.1×10^{15}	4.5×10^{19}	1.17×10^{22}
n^2	300k	9.3M	657M	54G	929G
n^3	4420	44.208	756k	14.3M	95M
2^n	36	46	58	71	79
1.1^n	264	336	426	518	578
$n!$	14	16	19	22	24

Amíg Moore törvénye (nagyjából) igaz, addig a polinomidejű algoritmusok egy-egy újabb év elteltével konstans**szor** több adattal tudnak adott időn belül elbánni.

A kép forrása: Iván Szabolcs, Bonyolultságelmélet fóliasor