Redukciók

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2020. ősz

A cél

A korábbiakban több nyelvosztályt bevezettünk ($\mathcal{L}, \mathcal{P}, \mathcal{D}, \mathcal{EXP}$). Láttunk több példát központi matematikai problémákra: HAMILTON, $\overrightarrow{st}$-ELÉRHETŐSÉG, SZÓPROBLÉMA, FAKTORIZÁCIÓ,...

Központi kérdés, hogy az egyes problémákat elhelyezzük a bevezetett hierarchiában.

Cél egy fontos probléma minél pontosabb helyének/bonyolultságának meghatározása.

A cél két részre bontása

A „hely” meghatározása két feladatból áll.

(1) Egy L nyelv/probléma bonyolultságának felső becslése (azaz annak bizonyítása, hogy $L \in \mathcal{C}_1$) egy algoritmus megadását kívánja, majd az algoritmus analízését, ami mutatja, hogy a $\mathcal{C}_1$ osztályhoz tartozást igazol. Ilyen típusú eredmények (amelyek jóval a számítógépek megjelenése előtt felismerhetők a matematika történetben) alkotják az algoritmuselmélet kiindulópontját.

(2) Egy L nyelv/probléma bonyolultságának alsó becslése (azaz annak bizonyítása, hogy $L \notin \mathcal{C}_2$) jóval összetettebb. Azt kívánja, hogy rámutassunk egy elméleti nehézségre ami megakadályozza, hogy hatékony algoritmussal megoldhassunk egy feladatot, legyünk bármilyen okosak, legyenek bármilyen zseniális ötleteink.

(1) könnyű: Hatékony algoritmusok tervezése.

Nehézség igazolása

A (2) feladat jóval nehezebb, szinte azt mondhatjuk semilyen eredmény sem született ebben az irányban. Két fontos „támadási irányt" említünk meg:

(a) A Turing-gép általános modelljét helyettesítsük egy egyszerűbb számítási modellel (ami így várhatóan nem univerzális számítási fogalom) és próbáljunk ott alsó becsléseket bizonyítani. Például a SORTING (n szám nagyság szerinti sorbarendezése) problémánál csak két input szám összehasonlítása és az eredmény szerinti szétágazás alapján dolgozzon eljárásunk. A megkötés természetes. A legtöbb algoritmus ilyen. Belátható, hogy legalább $n \log n$ összehasonlítás szükséges az otuput kiszámításához.

(b) Ne (abszolút) nehézséget vizsgáljunk, hanem relatívet. Tehát célünk csak annak igazolása, hogy egy probléma legalább olyan nehéz mint egy másik.

2(b) elmesélve

Egy L nyelv/probléma ω inputja lényegében egy kérdés: ω hozzátartozik L-hez?

A redukció egy olyan hozzárendelés/számolás, ami a kérdés megválaszolása helyett egy új kérdést számol ki: "Leírok egy $\widetilde{\omega}$ új inputot és megkérdezem, hogy egy új $\widetilde{L}$ nyelvhez tartozik-e".

"Ha valaki megmondja a választ, akkor én meg tudom mondani az eredeti kérdésre a választ". Sőt az ugyanaz lesz mint az én kérdésemre.

$\widetilde{L}$ vállán állva L nem is nehéz. Persze az új kérdés számolásának elhanyagolhatónak kell lennie.

$\widetilde{L}$ elég komplex ahhoz, hogy tetszőleges L által leírt problémát megfogalmazhassunk mint $\widetilde{L}$ probléma.

2(b) formálisan: Redukciók

Nyelvek Karp-redukciói

Legyen $L, \widehat{L} \subset \Sigma^*$ két nyelv és $\mathcal{C}$ egy bonyolultsági osztály. L redukálható $\widehat{L}$-ra $\mathcal{C}$-ben, jelben: $L \preccurlyeq_{\mathcal{C}} \widehat{L}$, ha létezik R kiszámítható Turing-gép, hogy

(i) R egy $\mathcal{C}$ komplexitású gép/eljárás,

(ii) $\omega \in L$ pontosan akkor, ha $\widetilde{\omega} \in \widehat{L}$, ahol $\widetilde{\omega}$ az ω-ból R által kiszámolt jelsorozat.

Megjegyzések

A bevezetett reláció olvasata: L redukálható $\widehat{L}$-re $\mathcal{C}$-ben. Jelentése: Az $\widehat{L}$ nyelv eldöntési feladata „legalább olyan nehéz", mint az L-é „modulo $\mathcal{C}$".

Más redukció fogalmak is léteznek. A fenti definíció Karp munkásságában rejlik és általában Karp-redukcióként hivatkozzák. Ha szükség van ennek hangsúlyozására, akkor a $\preccurlyeq_{\mathcal{C}}^{\text{Karp}}$ jelölést használjuk. Ebben a kurzusban legtöbbször ilyen redukciót látunk. Legtöbbször le is hagyjuk a felső indexet.

Feladatok

$$\text{KLIKK} = \{\lceil G, k\rceil : \ G\text{-ben van } k \text{ elemű klikk}\}$$

$$\text{FÜGGETLEN-CSÚCSHALMAZ} = \{\lceil G, k\rceil : G\text{-ben van } k \text{ elemű független csúcshalmaz}\}$$

$$\text{LEFOGÁS} = \{\lceil G, k\rceil : G \text{ lefogható } k \text{ csúccsal}\}$$

Észrevétel

A fenti feladatok minden irányban egymásba redukálhatók.

Példa redukcióra

Példa

KLIKK$\preccurlyeq_{\mathcal{P}}$FÜGGETLEN-CSÚCSHALMAZ.

A redukció rendkívül egyszerű: Adott egy $\omega = \lceil G, k \rceil$ inputja a KLIKK problémának. Ekkor G kódjából kiszámoljuk a komplenterét (annak kódját). $\widetilde{\omega}$ a $\lceil \overline{G}, k \rceil$ karaktersorozat lesz. A korábbi gráfelméleti tanulmányainkból az új „kérdés" ekvivalens az eredetivel.

$\widetilde{\omega}$ kiszámításának bonyolultsága nyilván polinomiális (igazából logaritmikus tárban megoldható).

Példa redukcióra

Példa

FÜGGETLEN-CSÚCSHALMAZ$\preccurlyeq_{\mathcal{P}}$LEFOGÁS.

A redukció rendkívül egyszerű: Adott egy $\omega = \lceil G, k \rceil$ inputja a FÜGGETLEN-CSÚCSHALMAZ problémának. Ekkor G és k kódjából kiszámoljuk $|V(G)| - k$ értéket. $\widetilde{\omega}$ a $\lceil G, |V(G)| - k \rceil$ karaktersorozat lesz. A korábbi gráfelméleti tanulmányainkból az új „kérdés” ekvivalens az eredetivel.

$\widetilde{\omega}$ kiszámításának bonyolultsága nyilván polinomiális (igazából logaritmikus tárban megoldható).

Példa redukcióra

Példa

LEFOGÁS$\preccurlyeq_{\mathcal{P}}$KLIKK.

A redukció rendkívül egyszerű: Adott egy $\omega = \lceil G, k \rceil$ inputja a LEFOGÁS problémának. Ekkor G és k kódjából kiszámoljuk a komplmenterét és $|V(G)| - k$-t. $\widetilde{\omega}$ a $\lceil \overline{G}, |V(G)| - k \rceil$ karaktersorozat lesz. A korábbi gráfelméleti tanulmányainkból az új „kérdés” ekvivalens az eredetivel.

$\widetilde{\omega}$ kiszámításának bonyolultsága nyilván polinomiális (igazából logaritmikus tárban megoldható).

Megjegyezzük, hogy se a KLIKK, se a LEFOGÁS, se a FÜGGETLEN-CSÚCSHALMAZ nyelvre nem ismert hatékony algoritmus. Ha bármelyikre lenne, akkor az a másik problémára is jelentős kihatással lenne.

Szünet

Turing-redukció

Turing-redukció

Legyen $L, \widehat{L} \subseteq \Sigma^*$ két nyelv és $\mathcal{C}$ egy bonyolultsági osztály.
$L \preccurlyeq_{\mathcal{C}}^{\text{Turing}} \widehat{L}$ pontosan akkor, ha megadható R eldöntő Turing-gép, amelyre

(i) L-et dönti el és R egy L_2-orákulumos gép.

(ii) R bonyolultsága $\mathcal{C}$-beli.

(i)-ben szerepel egy eddig ismeretlen fogalom, amit tisztáznunk kell.

Orákulumos Turing-gép

Definíció: Orákulumos TG

Legyen $O \subset \Sigma^*$ egy nyelv.

R egy O-orákulumos gép, ha

- van egy extra kérdés/orákulum-szalagja.

  Erre csak írhat a gép (nincs szem a szalag felett, a kéz csak jobbra mozogva írhat). Az írott karakterek $\Sigma \cup \{?\}$ elemei, azaz az O nyelv ábécéjének elemei és egy speciális '?' jel. A kérdésszalagra a ? jel feírása egy kérdés feltételét jelenti. Az előző kérdőjel (vagy szalag-kezdet jel) és közte lévő Σ^*-beli karaktersorozatról kérdezi meg a gép/algoritmus, hogy az orákulum O nyelvéhez tartozik-e.

Orákulumos Turing-gép (folytatás)

Definíció: Orákulumos TG (folytatás)

- az állapothalmaz

$$\{\text{ORÁKULUM-IGEN}, \text{ORÁKULUM-NEM}\} \times S_0$$

alakú.

Az átmeneti függvény a következő konfigurációnak az állapotában csak a második komponensre hat. Az első komponens csak akkor változik, ha az algoritmus kérdést tesz fel az orákulumhoz. A változást a kérdés karaktersorozat O-hoz való viszonyától függ természetes módon.

Ezek után a futás (a kiinduló konfigurációból generált konfigurációsorozat), a kiszámított nyelv értelemszerűen definiálható. A fentiekből a kérdés ára 1 időegység és 0 tár.

A két redukció összehasonlítása

A Karp-redukció nagyon speciális Turing-redukció: Szokásos számolás után egyetlen kérdés hangozhat el az $\widehat{L}$ nyelvhez tartozásról. A kérdésre adott válasz egyben a kiszámított bit is.

A Turing-redukció nyilván sokkal erősebb fogalom. Az $\widehat{L}$-ra úgy gondolhatunk, mint egy megíratlan szubrutin.

A redukció lényege, hogy ha a $\widehat{L}$ szubrutint valaki hatékonyan megírja, akkor L hatékonyan eldönthető (feltéve, hogy R hozzájárulása (ami $\mathcal{C}$ bonyolultságú) is hatékonynak tekinthető).

Mi nem kívánjuk a a szubrutin megvalósítását, „meghívását” és az eredmény megkapását egyetlen egy lépésnek számoljuk.

Tranzitivitás

Végül megemlítünk egy fontos tulajdonságát néhány redukcióknak.

Lemma

(i) $\preccurlyeq_{\mathcal{P}}$ *tranzitív.*

(ii) $\preccurlyeq_{\mathcal{L}}$ *tranzitív.*

(iii) Legyen $s(n) \geq \log n$ *szép tárfüggvény. Ha* $L_1 \preccurlyeq_{\mathcal{SPACE}(\mathcal{O}(s(n))} L_2$ *és* $L_2 \preccurlyeq_{\mathcal{L}} L_3$ *, akkor* $L_1 \preccurlyeq_{\mathcal{SPACE}(\mathcal{O}(s(n))} L_3$

Polinomiális idejű redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preccurlyeq_{\mathcal{P}} L_2$ és $L_2 \preccurlyeq_{\mathcal{P}} L_3$. Továbbá R_1 és R_2 kettő, a két redukciót igazoló algoritmus.

Speciálisan R_1 és R_2 is polinomiális. Legyen p_1 és p_2 két polinom, ami R_1 és R_2 időkorlátját adják. p_2-ről feltehetjük, hogy monoton növő.

Egy ω inputon futtassuk R_1-et, ami $\widetilde{\omega}$ karaktersorozatot számolja ki. Majd R_2-t futtasuk $\widetilde{\omega}$-n, ami $\widetilde{\widetilde{\omega}}$ kiszámításához vezet.

Az így kapott Turing-gép legyen R. Belátjuk, hogy R a $L_1 \preccurlyeq_{\mathcal{P}} L_3$ redukciót igazolja.

Polinomiális idejű redukciók tranzitivitása (folytatás)

$\omega \in L_1$ akkor és csak akkor teljesül, ha $\widetilde{\omega} \in L_2$. Ami akkor és csak akkor teljesül, ha $\widetilde{\widetilde{\omega}} \in L_3$ban,

Be kell még látni, hogy R polinomiális. ω inputon R időigénye $p_1(|\omega|) + p_2(|\widetilde{\omega}|)$. $\widetilde{\omega}$-t egy p_1 időkorlátú gép számolja ki ω-ból, így $|\widetilde{\omega}| \leq p_1(\omega)$.

Így ω-n a futási idejére a következő felső becslés adódik

$$p_1(|\omega|) + p_2(|\widetilde{\omega}|) \leq p_1(|\omega|) + p_2(p_1(|\omega|)).$$

Ez egy polinomiális felső becslés.

Logaritmikus tárú redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preccurlyeq_{\mathcal{L}} L_2$ és $L_2 \preccurlyeq_{\mathcal{L}} L_3$. Továbbá R_1 és R_2 kettő, a két redukciót igazoló algoritmus. Speciálisan R_1 és R_2 is logaritmikus tárigényű.

A két redukcióból az előző módon rakunk össze egy R algoritmust: Egy ω inputon futtassuk R_1-et, ami $\widetilde{\omega}$ karaktersorozatot számolja ki. Majd R_2-t futtasuk $\widetilde{\omega}$-n, ami $\widetilde{\widetilde{\omega}}$ kiszámításához vezet.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

Az így kapott algoritmus NEM jó: A közbülsőnek kiszámolt $\widetilde{\omega}$-ra a munkaszalagon van szükség. Ez várhatóan nem fér el logaritmikus tárban. Ennek ellenére ezen R algoritmus futása legyen a fejünkben. Az igazi $\widetilde{R}$ redukcióban felismerjük R futásának töredékeit.

$\widetilde{R}$ munkaszalagjai megfelelnek R_1 munkaszalagjainak plusz R_2 munkaszalgajainak. Lesz két plusz szalagunk a korábbi szalag helyett, ami R_1 output- és a vele közös R_2 inputszalagja volt. A plusz két szalag közül az első R_1 output szalagjának egy pozíciójának indexét (a szalag feletti kéz pozícióját) tartalmazza, míg másik szalag tartalma R_2 inputszalagján egy pozíció indexe (a szalag feletti szem pozíciója).

Logaritmikus tárú redukciók tranzitivitása (folytatás)

$\widetilde{R}$ az R_2 szimulációját végzi az $\widetilde{\omega}$ inputszalag tartalom nélkül.

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorzat hanyadik karakterére vagyunk kiváncsiak. El kezdjük R_1 szimulálását. A szimuláció során a kiszámolt karaktereket nem írjuk le, csupán az output-kéz pozícióját tároljuk. Ha R_1 ír, akkor az új pozíciót összeasonlítjuk az olvasni kívánt pozícióval.

Ha megegyezik a két pozíció, akkor a le nem írt karaktert kiolvassuk az állapotból és az R_1-szimulációt leállítjuk, folytatjuk az R_2-szimulációt. Ha a két pozíció különbözik, akkor az R_1-szimulációt folytatjuk.

(iii) Az előző bizonyítás ötletei most is működnek.

Lemma

Lemma

(i) $L \preccurlyeq_{\mathcal{P}} \widehat{L}$ és $\widehat{L} \in \mathcal{P}$, akkor $L \in \mathcal{P}$.

(ii) $L \preccurlyeq_{\mathcal{L}} \widehat{L}$ és $\widehat{L} \in \mathcal{L}$, akkor $L \in \mathcal{L}$.

Lemma bizonyítása

(i) Tekintsünk egy A Turing-gépet, amely az L-ről $\widehat{L}$-ra történő redukciót végzi, valamint $\widehat{A}$-ot, amely a $\widehat{L}$ nyelvehz tartozási feladatot dönti el $\mathcal{P}$-ben.

Legyen adott az ω input.

Ekkor végezzük el a

$$\omega \to A(\omega) \in \Sigma^{p(n)} \to \widehat{A}(A(\omega)) \in \{\text{ELFOGAD}, \text{ELVET}\}$$

számolást.

Az első időigényét az n inputméretben egy p polinom korlátozza. A leghosszabb input, amit kiszámolhatunk $\Sigma^{p(n)}$-ba esik. A második lépés időigényét az inputméretben egy q polinom korlátozza. Az összidő $(p + q \circ p)(n)$, ami n egy polinomiális függvénye.

A két algoritmus együttese a Karp-redukció fogalma alapján éppen az L nyelvet dönti el, vagyis L is eldönthető polinom időben.

Lemma bizonyítása

(ii) Az (i) rész és az előző lemma ötletei alapján nyilvánvaló.

Szünet

Példa redukcióra

Példa

Legyen L egy tetszőleges $\mathcal{NL}$-beli nyelv. Ekkor

$$L \preccurlyeq_{\mathcal{L}} \text{IRÁNYÍTOTT-ELÉRHETŐSÉG.}$$

A példához tartozó igazolás tulajdonképpen már elhangzott korábban. A korábbi érvelés lényeges gondolatait egy tételben összefoglaljuk.

A tétel

Tétel

Legyen $L \in_T \mathcal{NL}$. Feltehető, hogy a tartalmazást igazoló Turing-gépnek adott hosszúságú inputokon kétféle leálló konfigurációja van (így elfogadó futások ugyanabban a konfigurációban érnek véget).
$\omega \in \Sigma^*$-hoz rendelhető a T logaritmikus tárú Turing-gép redukált konfigurációinak (irányított) $\overrightarrow{G} = \overrightarrow{G}_{T,\omega}$ gráfja. Ebben ott van v_0 a kiinduló konfigurációnak megfelelő csúcs és v_+ az elfogadó konfigurációnak megfelelő csúcs. Ez a hozzárendelés olyan, hogy

(i) $\omega \in L$ akkor és csak akkor, ha $\lceil \overrightarrow{G}_{\omega,T}, v_0, v_+ \rceil \in \overrightarrow{st}$-ELÉRHETŐSÉG.

(ii) A hozzárendelés kiszámítható és tárigénye $\mathcal{O}(\log(n))$.

Következmények

Ez a példa jóval általánosabb mint a korábbi példa. Hogy ezt lássuk nézzük meg néhány következményét.

Következmény

Ha $\overrightarrow{st}$-ELÉRHETŐSÉG $\in \mathcal{P}$, akkor $\mathcal{NL} \subset \mathcal{P}$.

A feltétel igaz, ez egyszerűen adódik például az algoritmuselméleti előadásokon megismert gráf-bejárási algoritmusok leírásából és elemzéséből.

A fenti következmény a $\mathcal{NL} \subset \mathcal{P}$ tartalmazásra adott korábbi bizonyításunk újratárgyalása.

Következmény

Következmény

Ha IRÁNYÍTOTT-ELÉRHETŐSÉG $\in \mathcal{L}$, akkor $\mathcal{NL} = \mathcal{L}$.

A konkluzió igazából „csak” $\mathcal{NL} \subset \mathcal{L}$ (a másik irányú tartalmazás nyilvánvaló).

Itt a feltétel igazsága nem ismert, sőt sokan úgy hiszik nem is igaz.

A teljesség definíciója

A fenti okoskodás nagyon fontos. A gondolatmenetet lényege, hogy a példa alapján IRÁNYÍTOTT-ELÉRHETŐSÉG az egész $\mathcal{NL}$ nyelvosztály nehézségét magában foglalja. Ez vezet el egy általánosabb fogalom megalkotásához:

Definíció

$\widehat{L}$ nyelv *teljes* a $\mathcal{C}$ osztályban az $\mathcal{R}$ bonyolultságú redukcióra, ha:

(i) $\widehat{L} \in \mathcal{C}$,

(ii) minden $L \in \mathcal{C}$ esetén $L \preccurlyeq_{\mathcal{R}} \widehat{L}$.

Speciális esetek

Négy speciális esetet kiemelünk.

Definíció

Az $\widehat{L}$ nyelv $\mathcal{NP}$-teljes ha teljes $\mathcal{NP}$-ben a $\mathcal{P}$ redukcióra. Azaz $\widehat{L}$ nyelv $\mathcal{NP}$-*teljes*, ha

(i) $\widehat{L} \in \mathcal{NP}$,

(ii) minden $L \in \mathcal{NP}$ esetén $L \preccurlyeq_{\mathcal{P}} \widehat{L}$.

A fenti megállapodás egy alternatívája a $\preccurlyeq_{\mathcal{L}}$ redukcióval való dolgozás. Ez a szigorúbb értelmezés is igaz lesz a legtöbb későbbi $\mathcal{NP}$-teljességet igazoló redukciókra. Mi azonban csak a redukciók könnyebben ellenőrizhető polinom időbeliségét követeljük meg.

Speciális esetek

Definíció

Az L nyelv $\mathcal{NL}$-teljes, ha teljes $\mathcal{NL}$-ben az $\mathcal{L}$-beli redukcióra nézve.

Definíció

Az L nyelv $\mathcal{P}$-teljes, ha teljes $\mathcal{P}$-ben az $\mathcal{L}$-beli redukcióra nézve.

Definíció

Az L nyelv $\mathcal{PSPACE}$-teljes, ha teljes $\mathcal{PSPACE}$-ben a $\mathcal{P}$-beli redukcióra nézve.

Nehézség

Definíció

$\widehat{L}$ *nehéz* a $\mathcal{C}$ osztályra $\mathcal{R}$ bonyolultságú redukcióval, ha minden $L \in \mathcal{C}$ esetén $L \preccurlyeq_{\mathcal{R}} \widehat{L}$.

Azaz a nehézség a teljesség fogalma az (i) feltétel nélkül. Azaz nem követeljük meg az osztályhoz tartozást csak az osztály elemeinek visszavezethetőségét rá.

Gyakran kiszámíthatósági feladatokra is mondják, ha $\mathcal{C}$-nehéz, ha van olyan redukciós algoritmus, amely által kiszámolt $\widetilde{\omega}$ karaktersorozatra a feladat olyan értéket ad, amiből ω nyelvhez tartozása könnyen látható.

Egy konkrét $\mathcal{NL}$-teljes probléma

Tétel

IRÁNYÍTOTT-ELÉRHETŐSÉG $\mathcal{NL}$-teljes.

A redukciók vizsgálatánál láttuk az $\mathcal{NL}$-nehézséget. Még korábban láttuk az $\mathcal{NL}$-hez tartozást.

A tétel alapján az $\overrightarrow{st}$-ELÉRHETŐSÉG-ről szerzett tudásunk egész $\mathcal{NL}$-re kihat.

Erre már láttunk egy példát: $\overrightarrow{st}$-ELÉRHETŐSÉG$\in \mathcal{P}$. Ez alapján adódott, hogy $\mathcal{NL} \subset \mathcal{P}$.

Savitch-algoritmus következménye

Savitch algoritmusa a tárral spórol. Az IRÁNYÍTOTT-ELÉRHETŐSÉG-et $\log^2$ tárban oldja meg. Egyből adódik a következő tétel:

Tétel

$$\mathcal{NL} \subset \mathcal{SPACE}(\log^2 n).$$

Tétel

Szép $s(n) > \log n$ tárfüggvényre tetszőleges $\mathcal{NSPACE}(s(n))$-beli probléma visszavezethető $\mathcal{O}(s(n))$ tárban egy $\overrightarrow{st}$-ELÉRHETŐSÉG problémára, amely csúcshalmazának mérete $2^{\mathcal{O}(s(n))}$.

Ez az elérhetőség probléma $\mathcal{O}(s^2(n))$ tárban megoldható (Savitch-algoritmus). A következő tétel adódott:

Tétel

Legyen $s(n) > \log n$ egy szép tárfüggvény. Ekkor

$$\mathcal{NSPACE}(s(n)) \subset \mathcal{SPACE}(\mathcal{O}(s^2(n))).$$

Következmények

Speciálisan kapjuk a $\mathcal{PSPACE} \subset \mathcal{NPSPACE}$ tartalmazás fordítottját. Azaz

Tétel

$\mathcal{PSPACE} = \mathcal{NPSPACE}$.

Tudjuk, hogy a determinisztikus osztályok zártak a komplementálásra. Speciálisan kapjuk a következőt:

Tétel

$\mathcal{NPSPACE}$ zárt a komplementálásra.

Absztrakció

Következmény

Legyen S szép tárfüggvények egy osztálya, ami zárt a négyzetre emelésre. Ekkor $\mathcal{NPSPACE}(\cup_{s\in S} s(n)) = \mathcal{PSPACE}(\cup_{s\in S} s(n))$.

Speciálisan $\mathcal{NPSPACE}(\cup_{s\in S} s(n))$ zárt a komplementálásra, azaz

$$\mathcal{NPSPACE}(\cup_{s\in S} s(n)) = co\mathcal{NPSPACE}(\cup_{s\in S} s(n)).$$

Így egy másik speciális eset: $\mathcal{EXPSPACE} = \mathcal{NEXPSPACE}$.

Vége van!

Köszönöm a figyelmet!