

Redukciók

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2020. ősz

A cél

A cél

A korábbiakban több nyelvosztályt bevezettünk (\mathcal{L} , \mathcal{P} , \mathcal{D} , \mathcal{EXP}).

A cél

A korábbiakban több nyelvosztályt bevezettünk ($\mathcal{L}, \mathcal{P}, \mathcal{D}, \mathcal{EXP}$).

Láttunk több példát központi matematikai problémákra:

HAMILTON, \vec{st} -ELÉRHETŐSÉG, SZÓPROBLÉMA,
FAKTORIZÁCIÓ,...

A cél

A korábbiakban több nyelvosztályt bevezettünk ($\mathcal{L}, \mathcal{P}, \mathcal{D}, \mathcal{EXP}$).

Láttunk több példát központi matematikai problémákra:

HAMILTON, \vec{st} -ELÉRHETŐSÉG, SZÓPROBLÉMA,
FAKTORIZÁCIÓ,...

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

A cél

A korábbiakban több nyelvosztályt bevezettünk ($\mathcal{L}, \mathcal{P}, \mathcal{D}, \mathcal{EXP}$).

Láttunk több példát központi matematikai problémákra:

HAMILTON, \vec{st} -ELÉRHETŐSÉG, SZÓPROBLÉMA,
FAKTORIZÁCIÓ,...

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

Cél egy fontos probléma minél pontosabb helyénék/bonyolultságának meghatározása.

A cél két részre bontása

A cél két részre bontása

A „hely” meghatározása két feladatból áll.

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.

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.

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:

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

Nehézség igazolása

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

Nehézség igazolása

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

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.

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

2(b) elmesélve

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

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 $\tilde{\omega}$ új inputot és megkérdezem, hogy egy új \tilde{L} nyelvhez tartozik-e".

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 $\tilde{\omega}$ új inputot és megkérdezem, hogy egy új \tilde{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.

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 $\tilde{\omega}$ új inputot és megkérdezem, hogy egy új \tilde{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 kérdésekre.

\tilde{L} vállán állva L nem is nehéz.

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 $\tilde{\omega}$ új inputot és megkérdezem, hogy egy új \tilde{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 kérdésekre.

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

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 $\tilde{\omega}$ új inputot és megkérdezem, hogy egy új \tilde{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 kérdésekre.

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

\tilde{L} elég komplex ahhoz, hogy tetszőleges L által leírt problémát megfogalmazzassunk mint \tilde{L} probléma.

2(b) formálisan: Redukciók

2(b) formálisan: Redukciók

Nyelvek Karp-redukciói

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

2(b) formálisan: Redukciók

Nyelvek Karp-redukciói

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

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

2(b) formálisan: Redukciók

Nyelvek Karp-redukciói

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

- (i) R egy \mathcal{C} komplexitású gép/eljárás,
- (ii) $\omega \in L$ pontosan akkor, ha $\tilde{\omega} \in \hat{L}$, ahol $\tilde{\omega}$ az ω -ból R által kiszámolt jelsorozat.

Megjegyzések

Megjegyzések

A bevezetett reláció olvasata: L redukálható \widehat{L} -re \mathcal{C} -ben.

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} ”.

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.

Megjegyzések

A bevezetett reláció olvasata: L redukálható \hat{L} -re \mathcal{C} -ben. Jelentése: Az \hat{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 $\preceq_{\mathcal{C}}^{\text{Karp}}$ jelölést használjuk.

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 $\preceq_{\mathcal{C}}^{\text{Karp}}$ jelölést használjuk. Ebben a kurzusban legtöbbször ilyen redukciót látunk. Legtöbbször le is hagyjuk a felső indexet.

Feladatok

Feladatok

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

Feladatok

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

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

Feladatok

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

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

$$\text{LEFOGÁS} = \{[G, k] : G \text{ lefogható } k \text{ csúccsal}\}$$

Feladatok

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

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

$\text{LEFOGÁS} = \{[G, k] : G \text{ lefogható } k \text{ csúccsal}\}$

Észrevétel

A fenti feladatok minden irányban egymásba redukálhatók.

Példa redukcióra

Példa redukcióra

Példa

KLICK $\preceq_{\mathcal{P}}$ FÜGGETLEN-CSÚCSHALMAZ.

Példa redukcióra

Példa

$\text{KLIKK} \leq_p \text{FÜGGETLEN-CSÚCSHALMAZ}$.

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a KLIKK problémának.

Példa redukcióra

Példa

$\text{KLIKK} \leq_p \text{FÜGGETLEN-CSÚCSHALMAZ}$.

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a KLIKK problémának. Ekkor G kódjából kiszámoljuk a komplementerét (annak kódját). $\tilde{\omega}$ a $[\bar{G}, k]$ karaktersorozat lesz.

Példa redukcióra

Példa

KLICK \preceq_P FÜGGETLEN-CSÚCSHALMAZ.

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a KLIKK problémának. Ekkor G kódjából kiszámoljuk a komplementerét (annak kódját). $\tilde{\omega}$ a $[\bar{G}, k]$ karaktersorozat lesz. A korábbi gráfelméleti tanulmányainkból az új „kérdés” ekvivalens az eredetivel.

Példa redukcióra

Példa

KLICK \preceq_P FÜGGETLEN-CSÚCSHALMAZ.

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a KLICK problémának. Ekkor G kódjából kiszámoljuk a komplementerét (annak kódját). $\tilde{\omega}$ a $[\overline{G}, k]$ karaktersorozat lesz. A korábbi gráfelméleti tanulmányainkból az új „kérdés” ekvivalens az eredetivel.

$\tilde{\omega}$ kiszámításának bonyolultsága nyilván polinomiális (igazából logaritmikus tárbán megoldható).

Példa redukcióra

Példa redukcióra

Példa

FÜGGETLEN-CSÚCSHALMAZ \preceq_P LEFOGÁS.

Példa redukcióra

Példa

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

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a FÜGGETLEN-CSÚCSHALMAZ problémának.

Példa redukcióra

Példa

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

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a FÜGGETLEN-CSÚCSHALMAZ problémának. Ekkor G és k kódjából kiszámoljuk $|V(G)| - k$ értéket. $\tilde{\omega}$ a $[G, |V(G)| - k]$ karaktersorozat lesz.

Példa redukcióra

Példa

FÜGGETLEN-CSÚCSHALMAZ \preceq_P LEFOGÁS.

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

Példa redukcióra

Példa

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

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

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

Példa redukcióra

Példa redukcióra

Példa

LEFOGÁS \preceq_P KLIKK.

Példa redukcióra

Példa

LEFOGÁS \preceq_P KLIKK.

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a LEFOGÁS problémának.

Példa redukcióra

Példa

LEFOGÁS \preceq_P KLIKK.

A redukció rendkívül egyszerű: Adott egy $\omega = [G, k]$ inputja a LEFOGÁS problémának. Ekkor G és k kódjából kiszámoljuk a komplementerét és $|V(G)| - k$ -t. $\tilde{\omega}$ a $[\overline{G}, |V(G)| - k]$ karaktersorozat lesz.

Példa redukcióra

Példa

LEFOGÁS \preceq_P KLIKK.

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

Példa redukcióra

Példa

LEFOGÁS \preceq_P KLIKK.

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

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

Példa redukcióra

Példa

LEFOGÁS \preceq_P KLIKK.

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

$\tilde{\omega}$ 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.

Példa redukcióra

Példa

LEFOGÁS \preceq_P KLIKK.

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

$\tilde{\omega}$ 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

Turing-redukció

Legyen $L, \hat{L} \subseteq \Sigma^*$ két nyelv és \mathcal{C} egy bonyolultsági osztály.

Turing-redukció

Turing-redukció

Legyen $L, \hat{L} \subseteq \Sigma^*$ két nyelv és \mathcal{C} egy bonyolultsági osztály.

$L \preceq_{\mathcal{C}}^{\text{Turing}} \hat{L}$ pontosan akkor, ha megadható R eldöntő Turing-gép, amelyre

Turing-redukció

Turing-redukció

Legyen $L, \hat{L} \subseteq \Sigma^*$ két nyelv és \mathcal{C} egy bonyolultsági osztály.

$L \preceq_{\mathcal{C}}^{\text{Turing}} \hat{L}$ pontosan akkor, ha megadható R eldöntő Turing-gép, amelyre

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

Turing-redukció

Turing-redukció

Legyen $L, \hat{L} \subseteq \Sigma^*$ két nyelv és \mathcal{C} egy bonyolultsági osztály.

$L \preceq_{\mathcal{C}}^{\text{Turing}} \hat{L}$ 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.

Turing-redukció

Turing-redukció

Legyen $L, \hat{L} \subseteq \Sigma^*$ két nyelv és \mathcal{C} egy bonyolultsági osztály.

$L \preceq_{\mathcal{C}}^{\text{Turing}} \hat{L}$ 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.

Órákulumos Turing-gép

Orákulumos Turing-gép

Definíció: Orákulumos TG

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

Orákulumos Turing-gép

Definíció: Orákulumos TG

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

R egy O -orákulumos gép, ha

- van egy extra kérdés/orákulum-szalagja.

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).

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.

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.

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-kezdő jel) és közte lévő Σ^* -beli karaktersorozatról kérdezi meg a gép/algorithmus, hogy az orákulum O nyelvéhez tartozik-e.

Orákulumos Turing-gép (folytatás)

Definíció: Orákulumos TG (folytatás)

Órákulumos Turing-gép (folytatás)

Definíció: Órákulumos TG (folytatás)

- az állapothalmaz

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

alakú.

Orákulumos Turing-gép (folytatás)

Definíció: Orákulumos TG (folytatás)

- az állapothalmaz

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

alakú.

Az átmeneti függvény a következő konfigurációnak az állapotában csak a második komponensre hat.

Órákulumos Turing-gép (folytatás)

Definíció: Órákulumos TG (folytatás)

- az állapothalmaz

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

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 órákulumhoz.

Órákulumos Turing-gép (folytatás)

Definíció: Órákulumos TG (folytatás)

- az állapothalmaz

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

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 órákulumhoz. A változást a kérdés karaktersorozat O -hoz való viszonyától függ természetes módon.

Órákulumos Turing-gép (folytatás)

Definíció: Órákulumos TG (folytatás)

- az állapothalmaz

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

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 órákulumhoz. 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.

Órákulumos Turing-gép (folytatás)

Definíció: Órákulumos TG (folytatás)

- az állapothalmaz

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

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 órákulumhoz. 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 két redukció összehasonlítása

A Karp-redukció nagyon speciális Turing-redukció: Szokásos számolás után egyetlen kérdés hangozhat el az \widehat{L} nyelvhez tartozásról.

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 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 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).

A két redukció összehasonlítása

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

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

A redukció lényege, hogy ha a \hat{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

Tranzitivitás

Végül megemlítünk egy fontos tulajdonságát néhány redukciónak.

Tranzitivitás

Végül megemlítünk egy fontos tulajdonságát néhány redukciónak.

Lemma

Tranzitivitás

Végül megemlítünk egy fontos tulajdonságát néhány redukciónak.

Lemma

(i) $\preceq_{\mathcal{P}}$ tranzitív.

Tranzitivitás

Végül megemlítünk egy fontos tulajdonságát néhány redukciónak.

Lemma

- (i) $\preceq_{\mathcal{P}}$ tranzitív.
- (ii) $\preceq_{\mathcal{L}}$ tranzitív.

Tranzitivitás

Végül megemlítünk egy fontos tulajdonságát néhány redukciónak.

Lemma

- (i) $\preceq_{\mathcal{P}}$ tranzitív.
- (ii) $\preceq_{\mathcal{L}}$ tranzitív.
- (iii) Legyen $s(n) \geq \log n$ szép tárfüggvény. Ha $L_1 \preceq_{SPACE(\mathcal{O}(s(n)))} L_2$ és $L_2 \preceq_{\mathcal{L}} L_3$, akkor $L_1 \preceq_{SPACE(\mathcal{O}(s(n)))} L_3$

Polinomiális idejű redukciók tranzitivitása

Polinomiális idejű redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preceq_{\mathcal{P}} L_2$ és $L_2 \preceq_{\mathcal{P}} L_3$.

Polinomiális idejű redukciók tranzitivitása

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

Polinomiális idejű redukciók tranzitivitása

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

Speciálisan R_1 és R_2 is polinomiális.

Polinomiális idejű redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preceq_{\mathcal{P}} L_2$ és $L_2 \preceq_{\mathcal{P}} L_3$. 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 növvő.

Polinomiális idejű redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preceq_{\mathcal{P}} L_2$ és $L_2 \preceq_{\mathcal{P}} L_3$. 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 növvő.

Egy ω inputon futtassuk R_1 -et, ami $\tilde{\omega}$ karaktersorozatot számolja ki.

Polinomiális idejű redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preceq_{\mathcal{P}} L_2$ és $L_2 \preceq_{\mathcal{P}} L_3$. 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 növvő.

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

Polinomiális idejű redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preceq_{\mathcal{P}} L_2$ és $L_2 \preceq_{\mathcal{P}} L_3$. 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 $\tilde{\omega}$ karaktersorozatot számolja ki. Majd R_2 -t futtassuk $\tilde{\omega}$ -n, ami $\tilde{\tilde{\omega}}$ kiszámításához vezet.

Az így kapott Turing-gép legyen R .

Polinomiális idejű redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preceq_{\mathcal{P}} L_2$ és $L_2 \preceq_{\mathcal{P}} L_3$. 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 növvő.

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

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

Polinomiális idejű redukciók tranzitivitása (folytatás)

Polinomiális idejű redukciók tranzitivitása (folytatás)

$\omega \in L_1$ akkor és csak akkor teljesül, ha $\tilde{\omega} \in L_2$.

Polinomiális idejű redukciók tranzitivitása (folytatás)

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

Polinomiális idejű redukciók tranzitivitása (folytatás)

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

Be kell még látni, hogy R polinomiális.

Polinomiális idejű redukciók tranzitivitása (folytatás)

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

Be kell még látni, hogy R polinomiális. ω inputon R időigénye $p_1(|\omega|) + p_2(|\tilde{\omega}|)$.

Polinomiális idejű redukciók tranzitivitása (folytatás)

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

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

Polinomiális idejű redukciók tranzitivitása (folytatás)

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

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

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

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

Polinomiális idejű redukciók tranzitivitása (folytatás)

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

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

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

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

Ez egy polinomiális felső becslés.

Logaritmikusan tárú redukciók tranzitivitása

Logaritmikus tárú redukciók tranzitivitása

Tegyük fel, hogy $L_1 \preceq_{\mathcal{L}} L_2$ és $L_2 \preceq_{\mathcal{L}} L_3$.

Logaritmikusan táru redukciók tranzitivitása

Tegyünk fel, hogy $L_1 \preceq_{\mathcal{L}} L_2$ és $L_2 \preceq_{\mathcal{L}} L_3$. Továbbá R_1 és R_2 kettő, a két redukciót igazoló algoritmus.

Logaritmikusan táru redukciók tranzitivitása

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

Logaritmikusan táru redukciók tranzitivitása

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

A két redukcióból az előző módon rakunk össze egy R algoritmust:

Logaritmikusan táru redukciók tranzitivitása

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

A két redukcióból az előző módon rakunk össze egy R algoritmust: Egy ω inputon futtassuk R_1 -et, ami $\tilde{\omega}$ karaktorsorozatot számolja ki.

Logaritmikus tárú redukciók tranzitivitása

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

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

Logaritmikusan tárú redukciók tranzitivitása (folytatás)

Logaritmikus tárú redukciók tranzitivitása (folytatás)

Az így kapott algoritmus NEM jó:

Logaritmikus tárú redukciók tranzitivitása (folytatás)

Az így kapott algoritmus NEM jó: A közbülsőnek kiszámolt $\tilde{\omega}$ -ra a munkaszalagon van szükség.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

Az így kapott algoritmus NEM jó: A közbülsőnek kiszámolt $\tilde{\omega}$ -ra a munkaszalagon van szükség. Ez várhatóan nem fér el logaritmikus tárban.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Logaritmikusan táru redukciók tranzitivitása (folytatás)

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

Logaritmikusan tárú redukciók tranzitivitása (folytatás)

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

\tilde{R} munkaszalagjai megfelelnek R_1 munkaszalagjainak plusz R_2 munkaszalagjainak.

Logaritmiikus tárú redukciók tranzitivitása (folytatás)

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

\tilde{R} munkaszalagjai megfelelnek R_1 munkaszalagjainak plusz R_2 munkaszalagjainak. Lesz két plusz szalagunk a korábbi szalag helyett, ami R_1 output- és a vele közös R_2 inputszalagja volt.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

\tilde{R} munkaszalagjai megfelelnek R_1 munkaszalagjainak plusz R_2 munkaszalagjainak. 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).

Logaritmikusan tárú redukciók tranzitivitása (folytatás)

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorozat hanyadik karakterére vagyunk kíváncsiak.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorozat hanyadik karakterére vagyunk kíváncsiak. El kezdjük R_1 szimulálását. A szimuláció során a kiszámolt karaktereket nem írjuk le, csupán az output-kéz pozícióját tároljuk.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorozat hanyadik karakterére vagyunk kíváncsiak. 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.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorozat hanyadik karakterére vagyunk kíváncsiak. 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,

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorozat hanyadik karakterére vagyunk kíváncsiak. 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.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorozat hanyadik karakterére vagyunk kíváncsiak. 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.

Logaritmikus tárú redukciók tranzitivitása (folytatás)

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

Minden olvasási feladatnál meg kell dolgoznunk.

Ekkor tudjuk, hogy az R_1 által kiszámolt karaktersorozat hanyadik karakterére vagyunk kíváncsiak. 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

Lemma

Lemma

Lemma

(i) $L \preceq_{\mathcal{P}} \hat{L}$ és $\hat{L} \in \mathcal{P}$, akkor $L \in \mathcal{P}$.

Lemma

Lemma

- (i) $L \preceq_{\mathcal{P}} \hat{L}$ és $\hat{L} \in \mathcal{P}$, akkor $L \in \mathcal{P}$.
- (ii) $L \preceq_{\mathcal{L}} \hat{L}$ és $\hat{L} \in \mathcal{L}$, akkor $L \in \mathcal{L}$.

Lemma bizonyítása

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} nyelvehez tartozási feladatot dönti el \mathcal{P} -ben.

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} nyelvez tartozási feladatot dönti el \mathcal{P} -ben.

Legyen adott az ω input.

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

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

számolást.

Lemma bizonyítása

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

Legyen adott az ω input.

Ekkor végezzük el a

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

számolást.

Az első időigényét az n inputméretben egy p polinom korlátozza. A leghosszabb input, amit kiszámolhatunk $\Sigma^{p(n)}$ -ba esik.

Lemma bizonyítása

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

Legyen adott az ω input.

Ekkor végezzük el a

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

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 összydő $(p + q \circ p)(n)$, ami n egy polinomiális függvénye.

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} nyelvezh tartozási feladatot dönti el \mathcal{P} -ben.

Legyen adott az ω input.

Ekkor végezzük el a

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

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

Lemma bizonyítása

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

Szünet



Példa redukcióra

Példa redukcióra

Példa

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

$L \preceq_{\mathcal{L}}$ IRÁNYÍTOTT-ELÉRHETŐSÉG.

Példa redukcióra

Példa

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

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

A példához tartozó igazolás tulajdonképpen már elhangzott korábban.

Példa redukcióra

Példa

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

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

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

A tétel

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 logaritmusos tárú Turing-gép redukált konfigurációinak (irányított) $\vec{G} = \vec{G}_{T,\omega}$ 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

$[\vec{G}_{\omega,T}, v_0, v_+] \in \vec{st}$ -ELÉRHETŐSÉG.

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

Következmények

Következmények

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

Következmények

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

Következmény

Ha $\vec{s}t$ -ELÉRHETŐSÉG $\in \mathcal{P}$, akkor $\mathcal{NL} \subset \mathcal{P}$.

Következmények

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

Következmény

Ha $\vec{s}t$ -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.

Következmények

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

Következmény

Ha \vec{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

Következmény

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

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).

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 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:

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

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

- (i) $\hat{L} \in \mathcal{C}$,
- (ii) minden $L \in \mathcal{C}$ esetén $L \preceq_{\mathcal{R}} \hat{L}$.

Speciális esetek

Speciális esetek

Négy speciális esetet kiemelünk.

Speciális esetek

Négy speciális esetet kiemelünk.

Definíció

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

Speciális esetek

Négy speciális esetet kiemelünk.

Definíció

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

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

Speciális esetek

Négy speciális esetet kiemelünk.

Definíció

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

- (i) $\hat{L} \in \mathcal{NP}$,
- (ii) minden $L \in \mathcal{NP}$ esetén $L \preceq_{\mathcal{P}} \hat{L}$.

Speciális esetek

Négy speciális esetet kiemelünk.

Definíció

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

- (i) $\hat{L} \in \mathcal{NP}$,
- (ii) minden $L \in \mathcal{NP}$ esetén $L \preceq_{\mathcal{P}} \hat{L}$.

A fenti megállapodás egy alternatívája a $\preceq_{\mathcal{L}}$ redukcióval való dolgozás.

Speciális esetek

Négy speciális esetet kiemelünk.

Definíció

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

- (i) $\hat{L} \in \mathcal{NP}$,
- (ii) minden $L \in \mathcal{NP}$ esetén $L \preceq_{\mathcal{P}} \hat{L}$.

A fenti megállapodás egy alternatívája a $\preceq_{\mathcal{L}}$ redukcióval való dolgozás. Ez a szigorúbb értelmezés is igaz lesz a legtöbb későbbi \mathcal{NP} -teljességet igazoló redukciókra.

Speciális esetek

Négy speciális esetet kiemelünk.

Definíció

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

- (i) $\hat{L} \in \mathcal{NP}$,
- (ii) minden $L \in \mathcal{NP}$ esetén $L \preceq_{\mathcal{P}} \hat{L}$.

A fenti megállapodás egy alternatívája a $\preceq_{\mathcal{L}}$ 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

Speciális esetek

Definíció

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

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.

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

Nehézség

Definíció

\hat{L} nehéz a \mathcal{C} osztályra \mathcal{R} bonyolultságú redukcióval, ha minden $L \in \mathcal{C}$ esetén $L \preceq_{\mathcal{R}} \hat{L}$.

Nehézség

Definíció

\hat{L} nehéz a \mathcal{C} osztályra \mathcal{R} bonyolultságú redukcióval, ha minden $L \in \mathcal{C}$ esetén $L \preceq_{\mathcal{R}} \hat{L}$.

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

Nehézség

Definíció

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

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 $\tilde{\omega}$ karaktersorozatra a feladat olyan értéket ad, amiből ω nyelvhez tartozása könnyen látható.

Egy konkrét \mathcal{NL} -teljes probléma

Egy konkrét \mathcal{NL} -teljes probléma

Tétel

IRÁNYÍTOTT-ELÉRHETŐSÉG \mathcal{NL} -teljes.

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.

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.

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-algoritmus következménye

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

Savitch-algoritmus következménye

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

Tétel

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

Tétel

Tétel

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

Tétel

Szép $s(n) > \log n$ tárfüggvényre tetszőleges $\mathcal{NSPACE}(s(n))$ -beli probléma visszavezethető $\mathcal{O}(s(n))$ tárban egy \vec{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

Szép $s(n) > \log n$ tárfüggvényre tetszőleges $\mathcal{NSPACE}(s(n))$ -beli probléma visszavezethető $\mathcal{O}(s(n))$ tárban egy \vec{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

Következmények

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

Következmények

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

Tétel

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

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:

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

Absztrakció

Következmény

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

Absztrakció

Következmény

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

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

Absztrakció

Következmény

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

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

Így egy másik speciális eset: $\mathcal{EXSPACE} = \mathcal{NEXSPACE}$.

Vége van!

Köszönöm a figyelmet!