A Turing-gép fogalma

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2020. ősz

Karakterek, szavak

Karakterek, szavak

A számítási feladat egy $f : \mathcal{I} \to \mathcal{O}$ függvény az inputok halmazából az outputok halmazába. Ezt formalizáljuk. Ehhez az $\mathcal{I}/\mathcal{O}$ halmazokat kódoljuk.

Karakterek, szavak

A számítási feladat egy $f : \mathcal{I} \to \mathcal{O}$ függvény az inputok halmazából az outputok halmazába. Ezt formalizáljuk. Ehhez az $\mathcal{I}/\mathcal{O}$ halmazokat kódoljuk.

Definíció: Ábécé

Σ *ábécé* egy nemüres véges halmaz. Elemeire mint *betűk* vagy *karakterek* hívatkozunk.

Karakterek, szavak

A számítási feladat egy $f : \mathcal{I} \to \mathcal{O}$ függvény az inputok halmazából az outputok halmazába. Ezt formalizáljuk. Ehhez az $\mathcal{I}/\mathcal{O}$ halmazokat kódoljuk.

Definíció: Ábécé

Σ *ábécé* egy nemüres véges halmaz. Elemeire mint *betűk* vagy *karakterek* hívatkozunk.

Definíció: Szavak

Σ ábécé esetén Σ^n az n hosszú karaktersorozatok, másképpen *szavak* halmaza. Σ^0 egy egyelemű halmaz, egyetlen eleme az ϵ üres (0 hosszú) szó. Σ^* a Σ ábécé-t használó véges szavak halmaza, azaz $\Sigma^* = \cup_{i=0}^{\infty} \Sigma^i$.

Kódolás

Kódolás

Definíció: Kódolás

Az input/output *kódolása* egy Σ ábécé választása és az input elemeinek azonosítása kódszavakkal, azaz Σ^* elemeivel.

Kódolás

Definíció: Kódolás

Az input/output *kódolása* egy Σ ábécé választása és az input elemeinek azonosítása kódszavakkal, azaz Σ^* elemeivel.

A kódoláshoz először választani kell egy ábécé-t. Ekkor a $\Sigma = \{0, 1\}$ választáshoz is ragaszkodhatnánk. Néha azonban technikailag egyszerűbb lesz nagyobb ábécé-vel dolgozni.

Kódolás

Definíció: Kódolás

Az input/output *kódolása* egy Σ ábécé választása és az input elemeinek azonosítása kódszavakkal, azaz Σ^* elemeivel.

A kódoláshoz először választani kell egy ábécé-t. Ekkor a $\Sigma = \{0, 1\}$ választáshoz is ragaszkodhatnánk. Néha azonban technikailag egyszerűbb lesz nagyobb ábécé-vel dolgozni.

Megjegyzés: Megjegyezzük, hogy csak megszámolható $\mathcal{I}$ és $\mathcal{O}$ halmazok kódolhatók. Azaz halmazelméleti okok miatt a valós számok halmaza nem kódolható.

Példák kódolásokra

Példák kódolásokra

Példa: Kettes számrendszer I

$\mathbb{N}$ egy kódolása lehet a következő. Legyen $\Sigma = \{0, 1\}$. Az x szám kódját úgy definiáljuk, hogy felírjuk kettes számrendszerben. $0 \mapsto 0$, $1 \mapsto 1$, $9 \mapsto 1001$, hiszen $9 = 1001_2$. Ez egy-egy, de nem bijektív leképezés $\mathbb{N}$ és $\{0, 1\}^*$ között: ϵ és a 0-val kezdődő, legalább kettő hosszú 0-1 sorozatok nem kódszavak.

Példák kódolásokra

Példa: Kettes számrendszer I

$\mathbb{N}$ egy kódolása lehet a következő. Legyen $\Sigma = \{0, 1\}$. Az x szám kódját úgy definiáljuk, hogy felírjuk kettes számrendszerben. $0 \mapsto 0$, $1 \mapsto 1$, $9 \mapsto 1001$, hiszen $9 = 1001_2$. Ez egy-egy, de nem bijektív leképezés $\mathbb{N}$ és $\{0, 1\}^*$ között: ϵ és a 0-val kezdődő, legalább kettő hosszú 0-1 sorozatok nem kódszavak.

Példa: Kettes számrendszer II

$\mathbb{N}_+$ (a pozitív egészek) egy kódolása lehet a következő. Legyen $\Sigma = \{0, 1\}$. Az x szám kódját úgy definiáljuk, hogy felírjuk kettes számrendszerben és a kezdő 1-est elhagyjuk. $1 \mapsto \epsilon$, $9 \mapsto 001$, hiszen $9 = 1001_2$. Ez egy bijekció $\mathbb{N}_+$ és $\{0, 1\}^*$ között. Ezt nevezzük $\mathbb{N}_+$ *standard kettes számrendszerbeli bijektív kódolásának*.

Példák kódolásokra

Példák kódolásokra

Példa: Tízes számrendszer I

A tízes számrendszerben való felírás is egy kódolása $\mathbb{N}$-nek. Ekkor $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden x természetes számnak egy kódját definiáljuk, a szokásost.

Példák kódolásokra

Példa: Tízes számrendszer I

A tízes számrendszerben való felírás is egy kódolása $\mathbb{N}$-nek. Ekkor $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden x természetes számnak egy kódját definiáljuk, a szokásost.

Ekkor Σ^* nem minden eleme kódol inputot. 002017 például nem kódol inputot.

Példák kódolásokra

Példa: Tízes számrendszer I

A tízes számrendszerben való felírás is egy kódolása $\mathbb{N}$-nek. Ekkor $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden x természetes számnak egy kódját definiáljuk, a szokásost.

Ekkor Σ^* nem minden eleme kódol inputot. 002017 például nem kódol inputot.

Példa: Tízes számrendszer II

Legyen $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden kódszót, számjegysorozatot a kezdő 0-kat elhagyva értelmezzük.

Példák kódolásokra

Példa: Tízes számrendszer I

A tízes számrendszerben való felírás is egy kódolása $\mathbb{N}$-nek. Ekkor $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden x természetes számnak egy kódját definiáljuk, a szokásost.

Ekkor Σ^* nem minden eleme kódol inputot. 002017 például nem kódol inputot.

Példa: Tízes számrendszer II

Legyen $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden kódszót, számjegysorozatot a kezdő 0-kat elhagyva értelmezzük.

Ekkor egy számnak több kódja is van. $2020, 02020, 000002020$ kódok mindegyike ugyanazt a számot kódolja.

Példák kódolásokra

Példa: Tízes számrendszer I

A tízes számrendszerben való felírás is egy kódolása $\mathbb{N}$-nek. Ekkor $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden x természetes számnak egy kódját definiáljuk, a szokásost.

Ekkor Σ^* nem minden eleme kódol inputot. 002017 például nem kódol inputot.

Példa: Tízes számrendszer II

Legyen $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Minden kódszót, számjegysorozatot a kezdő 0-kat elhagyva értelmezzük.

Ekkor egy számnak több kódja is van. $2020, 02020, 000002020$ kódok mindegyike ugyanazt a számot kódolja.

Mindkét megoldás kérdéseket vet fel.

Kérdések

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz.

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz. Esetleg elvárjuk, hogy ebben az esetben az algoritmus jelezze, hogy „rossz kód”.

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz. Esetleg elvárjuk, hogy ebben az esetben az algoritmus jelezze, hogy „rossz kód”.

Ha több kódja is van egy inputnak, akkor Ha igen, akkor mit teszünk?

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz. Esetleg elvárjuk, hogy ebben az esetben az algoritmus jelezze, hogy „rossz kód”.

Ha több kódja is van egy inputnak, akkor Ha igen, akkor mit teszünk? Gyakran kijelölhetünk egy standard kódot, amihez ragaszkodunk.

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz. Esetleg elvárjuk, hogy ebben az esetben az algoritmus jelezze, hogy „rossz kód”.

Ha több kódja is van egy inputnak, akkor Ha igen, akkor mit teszünk? Gyakran kijelölhetünk egy standard kódot, amihez ragaszkodunk. Feltehetjük, hogy a dtandard alakra hozás legyen része az algoritmusnak.

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz. Esetleg elvárjuk, hogy ebben az esetben az algoritmus jelezze, hogy „rossz kód”.

Ha több kódja is van egy inputnak, akkor Ha igen, akkor mit teszünk? Gyakran kijelölhetünk egy standard kódot, amihez ragaszkodunk. Feltehetjük, hogy a dtandard alakra hozás legyen része az algoritmusnak. Ha több kódja van egy inputnak, akkor elvárjuk-e, hogy két kód esetén el tudjuk dönteni, hogy ugyanazt az inputot kódolják-e?

Kérdések

Adott Σ^* egy eleme. Lehet, hogy nem kódol inputot? Ha igen, akkor mit teszünk?

Elvárjuk az algoritmust felhasználótól, hogy az általa megadott jelsorozat garantáltan inputot kódóljon és az algoritmussal bármi történhet, ha nem így tesz. Esetleg elvárjuk, hogy ebben az esetben az algoritmus jelezze, hogy „rossz kód”.

Ha több kódja is van egy inputnak, akkor Ha igen, akkor mit teszünk? Gyakran kijelölhetünk egy standard kódot, amihez ragaszkodunk. Feltehetjük, hogy a dtandard alakra hozás legyen része az algoritmusnak. Ha több kódja van egy inputnak, akkor elvárjuk-e, hogy két kód esetén el tudjuk dönteni, hogy ugyanazt az inputot kódolják-e?

Ezekkel a problémákkal nem foglalkozunk. A szokásos kódolások olyanok, hogy algoritmusaink a legnagyobb követelményt is könnyen teljesítik (esetleg kis többlet munkával).

Példák kódolásra

Példák kódolásra

Példa

$\mathbb{N}$ kódolása az $\Sigma = \{1\}$ ábécé-vel is lehetséges: n kódja legyen n darab 1-es (1^n). Ezt $\mathbb{N}$ *unáris kódolásának* nevezzük.

Példák kódolásra

Példa

$\mathbb{N}$ kódolása az $\Sigma = \{1\}$ ábécé-vel is lehetséges: n kódja legyen n darab 1-es (1^n). Ezt $\mathbb{N}$ *unáris kódolásának* nevezzük.

Példa

$\mathbb{Q}$ szokásos kódolásában $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, /, -\}$ A racionális számot kódoló szvak egyetlen „/" jelet tartalmaznak, ami előtt egy természetes szám áll (esetleg egyetlen $-$ előjellel kezdve), míg a törtjel után egy pozitív egész kódja áll. Ebben a kódolásban a „fél" racionális számnak $1/2$ és $2017/3034$ is kódja. $1/2/3$, $12/0$, $-1/-2$ nem kódolnak racionális számot (legalábbis nem a fenti megállapodások alapján).

Példák számítási feladatra

Példák számítási feladatra

A kódolás után egy számítási feladat egy $f : \Sigma^* \to \Sigma^*$ függvény.

Példák számítási feladatra

A kódolás után egy számítási feladat egy $f : \Sigma^* \to \Sigma^*$ függvény.

FAKTORIZÁCIÓ I

Legyen $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ;, .\} \cup \{,\}$ Input: egy pozitív egész szám tízes számrendszerben felírva. Output: prím osztóinak növekvő listája, mindegyik osztót multiplicitása követi egy vesszővel elválasztva, majd egy pontos vessző követi, kivéve az utolsó prím osztót és multiplicitását, amit egy pont követ. Azaz:

$$24 \mapsto 2, 3; 3, 1.$$

$$121 \mapsto 11, 2.$$

$$43 \mapsto 43, 1.$$

$$2010 \mapsto 2, 1; 3, 1; 5, 1; 67, 1.$$

Példák számítási feladatra

Példák számítási feladatra

FAKTORIZÁCIÓ II

Legyen $\Sigma = 0, 1$. Az n input számot kettes számrendszerben kódoljuk. $f(\omega)$ legyen az ω által kódolt szám legkisebb prímtényezőjének kódja.

Példák számítási feladatra

FAKTORIZÁCIÓ II

Legyen $\Sigma = 0, 1$. Az n input számot kettes számrendszerben kódoljuk. $f(\omega)$ legyen az ω által kódolt szám legkisebb prímtényezőjének kódja.

FAKTORIZÁCIÓ III

Legyen $\Sigma = 0, 1$. Az input egy n, t számpar mondjuk 10 számrendszerben kódolva ($\Sigma = \{0, 1, `,'\}$ a számpár két elemének elhatárolására használjuk a vessző karaktert). Egy bit számítandó ki: Van-e n-nek olyan o osztója, hogy $2 \leq o \leq t$

Példák számítási feladatra

FAKTORIZÁCIÓ II

Legyen $\Sigma = 0, 1$. Az n input számot kettes számrendszerben kódoljuk. $f(\omega)$ legyen az ω által kódolt szám legkisebb prímtényezőjének kódja.

FAKTORIZÁCIÓ III

Legyen $\Sigma = 0, 1$. Az input egy n, t számpar mondjuk 10 számrendszerben kódolva ($\Sigma = \{0, 1, `, '\}$ a számpár két elemének elhatárolására használjuk a vessző karaktert). Egy bit számítandó ki: Van-e n-nek olyan o osztója, hogy $2 \leq o \leq t$

Talán zavaró, de sok FAKTORIZÁCIÓ problémát formalizáltunk. Ezek mind formálisan különböznek, mégis mind ugyanazon matematikai problémát jelenítik meg. Ezen tegyük túl magunkat.

Egy zavaró példa

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Bináris és tízes számrenszer közötti oda.vissza konverzió standard középiskolás feladat.

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Bináris és tízes számrenszer közötti oda.vissza konverzió standard középiskolás feladat.

Az unárisan kódolt szám könnyen felírható például tízes számrendszerben.

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Bináris és tízes számrenszer közötti oda.vissza konverzió standard középiskolás feladat.

Az unárisan kódolt szám könnyen felírható például tízes számrendszerben. Fordítva — habár nem látunk nehézséget — mégis problémás a feladat. Az unáris felírás hossza (ennek megfelelően a felíráshoz szükséges idő is) hatalmas.

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Bináris és tízes számrenszer közötti oda.vissza konverzió standard középiskolás feladat.

Az unárisan kódolt szám könnyen felírható például tízes számrendszerben. Fordítva — habár nem látunk nehézséget — mégis problémás a feladat. Az unáris felírás hossza (ennek megfelelően a felíráshoz szükséges idő is) hatalmas.

$\{0, 1, 2, \ldots, n-1\}$ elemű halmaz standard kódolásaiban minden szám kódja legfeljebb $\mathcal{O}(\log n)$ hosszú.

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Bináris és tízes számrenszer közötti oda.vissza konverzió standard középiskolás feladat.

Az unárisan kódolt szám könnyen felírható például tízes számrendszerben. Fordítva — habár nem látunk nehézséget — mégis problémás a feladat. Az unáris felírás hossza (ennek megfelelően a felíráshoz szükséges idő is) hatalmas.

$\{0, 1, 2, \ldots, n-1\}$ elemű halmaz standard kódolásaiban minden szám kódja legfeljebb $\mathcal{O}(\log n)$ hosszú. Bármilyen nagyobb ábécé-t választunk ennél nagyságrendileg jobb kódolást nem találhatunk.

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Bináris és tízes számrenszer közötti oda.vissza konverzió standard középiskolás feladat.

Az unárisan kódolt szám könnyen felírható például tízes számrendszerben. Fordítva — habár nem látunk nehézséget — mégis problémás a feladat. Az unáris felírás hossza (ennek megfelelően a felíráshoz szükséges idő is) hatalmas.

$\{0, 1, 2, \ldots, n-1\}$ elemű halmaz standard kódolásaiban minden szám kódja legfeljebb $\mathcal{O}(\log n)$ hosszú. Bármilyen nagyobb ábécé-t választunk ennél nagyságrendileg jobb kódolást nem találhatunk. Az ábécé növelése csak konstansszorosára „nyomja össze” a kódokat.

Egy zavaró példa

A $\Sigma = \{1\}$ „minimál ábécé” is egy lehetőség. Ez azonban természetellenes és ennek használata egy veszélyt rejt.

Bináris és tízes számrenszer közötti oda.vissza konverzió standard középiskolás feladat.

Az unárisan kódolt szám könnyen felírható például tízes számrendszerben. Fordítva — habár nem látunk nehézséget — mégis problémás a feladat. Az unáris felírás hossza (ennek megfelelően a felíráshoz szükséges idő is) hatalmas.

$\{0, 1, 2, \ldots, n-1\}$ elemű halmaz standard kódolásaiban minden szám kódja legfeljebb $\mathcal{O}(\log n)$ hosszú. Bármilyen nagyobb ábécé-t választunk ennél nagyságrendileg jobb kódolást nem találhatunk. Az ábécé növelése csak konstansszorosára „nyomja össze” a kódokat.

Az egyelemű ábécé viszont rossz, ebben legalább $n-1$ hosszú kódszó is szükséges bármit teszünk.

Az input mérete

Az input mérete

Egy kódolás után beszélhetünk az *input méretéről*, az input jelsorozat karaktereinek száma, az input hossza.

Az input mérete

Egy kódolás után beszélhetünk az *input méretéről*, az input jelsorozat karaktereinek száma, az input hossza.

Példa

$\mathbb{N}$ standard kódolásában n kódjának hossza $\lceil \log_2 n \rceil$. $\mathbb{N}$ unáris kódolásában n kódjának hossza n.

Az input mérete

Egy kódolás után beszélhetünk az *input méretéről*, az input jelsorozat karaktereinek száma, az input hossza.

Példa

$\mathbb{N}$ standard kódolásában n kódjának hossza $\lceil \log_2 n \rceil$. $\mathbb{N}$ unáris kódolásában n kódjának hossza n.

Példa

Legyen G egy egyszerű gráf a $V = \{1, 2, \ldots, v\}$ csúcshalmazon. Kódolásához legyen $\Sigma = \{0, 1\}$. G kódjának hossza $\binom{v}{2}$ lesz (az ilyen alakú számokat nevezzük *háromszögszámoknak*). A kód pozíciói V kételemű részhalmazaival vannak azonosítva. Egy karakter/bit azt kódolja, hogy a megfelelő két csúcs össze van-e kötve. Mivel $2^{\binom{v}{2}}$ a kódolandó objektumok száma és $|\Sigma| = 2$ ezért rövidebb kódszavakkal dolgozva nem is lehetséges az összes v csúcsú egyszerű gráf kódolása.

Az input mérete

Az input mérete

Példa

Legyen G egy e élű egyszerű gráf a $V = \{1, 2, \ldots, v\}$ csúcshalmazon. Ekkor kódja lehet, hogy V elemeit felsoroljuk és mindegyikhez kettőspont után felsoroljuk szomszédait (amelyek sorát pontosvesszővel választjuk el és ponttal zárjuk le). Azaz $\Sigma = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0, ;, :, .\}$. Például egy kódolt gráf $1 : 2; 4.2 : 1.3 : 1; 5.4 : 1.5 : 3$. A gráf kódjának hosszát felülről becsülhetjük $(v + 2e)(\lceil \log_{10} \rceil v + 1)$-gyel. Nagyságrendileg tömörebb kódolást nem is remélhetünk, hisz a kódolandó objektumok száma $\binom{\binom{v}{2}}{e}$.

Döntési problémát

Döntési problémát

Definíció: Döntési problémák

Az f számítási feladat *döntési feladat*, ha csak 0/1 értékeket vesz fel.

Döntési problémát

Definíció: Döntési problémák

Az f számítási feladat *döntési feladat*, ha csak 0/1 értékeket vesz fel.

Azt is mondhatjuk (ez így szokás) hogy egy inputot vagy elfogadunk vagy elvetünk.

Döntési problémát

Definíció: Döntési problémák

Az f számítási feladat *döntési feladat*, ha csak 0/1 értékeket vesz fel.

Azt is mondhatjuk (ez így szokás) hogy egy inputot vagy elfogadunk vagy elvetünk. Tehát egy eldöntési probléma azonosítható Σ^* egy részhalmazával, az elfogadandó inputok halmazával.

Döntési problémát

Definíció: Döntési problémák

Az f számítási feladat *döntési feladat*, ha csak 0/1 értékeket vesz fel.

Azt is mondhatjuk (ez így szokás) hogy egy inputot vagy elfogadunk vagy elvetünk. Tehát egy eldöntési probléma azonosítható Σ^* egy részhalmazával, az elfogadandó inputok halmazával.

Definíció

A szavak egy részhalmazát *nyelvnek* nevezzük. Azaz egy nyelv: $L \subset \Sigma^*$.

Döntési problémát

Definíció: Döntési problémák

Az f számítási feladat *döntési feladat*, ha csak 0/1 értékeket vesz fel.

Azt is mondhatjuk (ez így szokás) hogy egy inputot vagy elfogadunk vagy elvetünk. Tehát egy eldöntési probléma azonosítható Σ^* egy részhalmazával, az elfogadandó inputok halmazával.

Definíció

A szavak egy részhalmazát *nyelvnek* nevezzük. Azaz egy nyelv: $L \subset \Sigma^*$.

Összegezve: Egy döntési probléma leírható egy $L \subset \Sigma^*$ nyelvvel.

Döntési problémát

Definíció: Döntési problémák

Az f számítási feladat *döntési feladat*, ha csak 0/1 értékeket vesz fel.

Azt is mondhatjuk (ez így szokás) hogy egy inputot vagy elfogadunk vagy elvetünk. Tehát egy eldöntési probléma azonosítható Σ^* egy részhalmazával, az elfogadandó inputok halmazával.

Definíció

A szavak egy részhalmazát *nyelvnek* nevezzük. Azaz egy nyelv: $L \subset \Sigma^*$.

Összegezve: Egy döntési probléma leírható egy $L \subset \Sigma^*$ nyelvvel. Illetve a nyelv értelmezhető, mint a „hozzátartozik-e" döntési probléma.

Algoritmus: statikus kép

Algoritmus: statikus kép

Az algoritmusok véges módon leírható (mondhatjuk azt is hogy kódolhatók).

Algoritmus: statikus kép

Az algoritmusok véges módon leírható (mondhatjuk azt is hogy kódolhatók).

Ilyen kódokat láthatunk algoritmuselméleti könyvekben vagy más tankönyvekben, ahol algoritmusokat ismertetnek.

Algoritmus: statikus kép

Az algoritmusok véges módon leírható (mondhatjuk azt is hogy kódolhatók).

Ilyen kódokat láthatunk algoritmuselméleti könyvekben vagy más tankönyvekben, ahol algoritmusokat ismertetnek.

Persze a kódolás egy megállapodás. Sokféle megállapodás létezik.

Algoritmus: statikus kép

Az algoritmusok véges módon leírható (mondhatjuk azt is hogy kódolhatók).

Ilyen kódokat láthatunk algoritmuselméleti könyvekben vagy más tankönyvekben, ahol algoritmusokat ismertetnek.

Persze a kódolás egy megállapodás. Sokféle megállapodás létezik. Ezt tapasztalja az, aki egy programozási nyelvet választ.

Algoritmus: statikus kép

Az algoritmusok véges módon leírható (mondhatjuk azt is hogy kódolhatók).

Ilyen kódokat láthatunk algoritmuselméleti könyvekben vagy más tankönyvekben, ahol algoritmusokat ismertetnek.

Persze a kódolás egy megállapodás. Sokféle megállapodás létezik. Ezt tapasztalja az, aki egy programozási nyelvet választ. Mindegyik nyelv egy-egy kódolási módja algoritmusoknak.

Algoritmus: Dinamikus kép

Algoritmus: Dinamikus kép

Emellett van egy dinamikus kép is.

Algoritmus: Dinamikus kép

Emellett van egy dinamikus kép is.

Gondoljunk egy filmre, ami azt mutatja, hogy két háromjegyű szám esetén szorzatukat kiszámoljuk.

Algoritmus: Dinamikus kép

Emellett van egy dinamikus kép is.

Gondoljunk egy filmre, ami azt mutatja, hogy két háromjegyű szám esetén szorzatukat kiszámoljuk. Az idő telésével a papírunkon számok jelennek meg, egész a számítás végéig szervezzünk munkaterületünket, amikor is leállunk (például az eredmény kétszeres aláhúzásával jelezve a számítás végét).

Algoritmus: Dinamikus kép

Emellett van egy dinamikus kép is.

Gondoljunk egy filmre, ami azt mutatja, hogy két háromjegyű szám esetén szorzatukat kiszámoljuk. Az idő telésével a papírunkon számok jelennek meg, egész a számítás végéig szervezzünk munkaterületünket, amikor is leállunk (például az eredmény kétszeres aláhúzásával jelezve a számítás végét).

A tankönyvben leírt eljárást akkor értettük meg, ha képesek vagyunk futtani azt különböző inputokon.

Algoritmus: Dinamikus kép

Emellett van egy dinamikus kép is.

Gondoljunk egy filmre, ami azt mutatja, hogy két háromjegyű szám esetén szorzatukat kiszámoljuk. Az idő telésével a papírunkon számok jelennek meg, egész a számítás végéig szervezzünk munkaterületünket, amikor is leállunk (például az eredmény kétszeres aláhúzásával jelezve a számítás végét).

A tankönyvben leírt eljárást akkor értettük meg, ha képesek vagyunk futtani azt különböző inputokon. Illetve, ha jól begyakoroltunk egy algoritmust és a matematikai és nyelvi kifejezés egy szintjét elértük, akkor képeseknek kell lennünk valamilyen szintű leírását adni az eljárásunknak.

Algoritmus: Dinamikus kép

Emellett van egy dinamikus kép is.

Gondoljunk egy filmre, ami azt mutatja, hogy két háromjegyű szám esetén szorzatukat kiszámoljuk. Az idő telésével a papírunkon számok jelennek meg, egész a számítás végéig szervezzünk munkaterületünket, amikor is leállunk (például az eredmény kétszeres aláhúzásával jelezve a számítás végét).

A tankönyvben leírt eljárást akkor értettük meg, ha képesek vagyunk futtani azt különböző inputokon. Illetve, ha jól begyakoroltunk egy algoritmust és a matematikai és nyelvi kifejezés egy szintjét elértük, akkor képeseknek kell lennünk valamilyen szintű leírását adni az eljárásunknak.

A statikus és dinamikus szemlélet együtt jár. Azért életünk/matematikai tanulmányaink során a dinamikus képpel találkozunk először.

Történeti megjegyzések

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki.

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt.

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt. Ebben azt kérdezte Hilbert, hogy van-e olyan algoritmus, ami egy adott egészegyütthatós polinomról eldönti, hogy van-e egész gyöke (Diophantikus számelmélet alapproblémája).

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt. Ebben azt kérdezte Hilbert, hogy van-e olyan algoritmus, ami egy adott egészegyütthatós polinomról eldönti, hogy van-e egész gyöke (Diophantikus számelmélet alapproblémája).

A válasz, mint később kiderült: NINCS.

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt. Ebben azt kérdezte Hilbert, hogy van-e olyan algoritmus, ami egy adott egészegyütthatós polinomról eldönti, hogy van-e egész gyöke (Diophantikus számelmélet alapproblémája).

A válasz, mint később kiderült: NINCS. Ennek igazolása már lehetetlen az algoritmus fogalmának matematizálása nélkül.

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt. Ebben azt kérdezte Hilbert, hogy van-e olyan algoritmus, ami egy adott egészegyütthatós polinomról eldönti, hogy van-e egész gyöke (Diophantikus számelmélet alapproblémája).

A válasz, mint később kiderült: NINCS. Ennek igazolása már lehetetlen az algoritmus fogalmának matematizálása nélkül.

A probléma érdekes. Az algoritmus/eljárás szavak részei mindennapi szóhasználatunknak, de a matematikában nincsenek értelmezve.

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt. Ebben azt kérdezte Hilbert, hogy van-e olyan algoritmus, ami egy adott egészegyütthatós polinomról eldönti, hogy van-e egész gyöke (Diophantikus számelmélet alapproblémája).

A válasz, mint később kiderült: NINCS. Ennek igazolása már lehetetlen az algoritmus fogalmának matematizálása nélkül.

A probléma érdekes. Az algoritmus/eljárás szavak részei mindennapi szóhasználatunknak, de a matematikában nincsenek értelmezve. Egy definíció megadása, akkor sikeres, ha a matematikus társadalom többsége rábólint: „elfogadom korrekt definíciónak”.

Történeti megjegyzések

Az algoritmus matematikai fogalma a XX. század első harmadában alakult ki. Egyik ösztönzője a logika fejlődése, illetve Hilbert nevezetes problémái közül a tizedik volt. Ebben azt kérdezte Hilbert, hogy van-e olyan algoritmus, ami egy adott egészegyütthatós polinomról eldönti, hogy van-e egész gyöke (Diophantikus számelmélet alapproblémája).

A válasz, mint később kiderült: NINCS. Ennek igazolása már lehetetlen az algoritmus fogalmának matematizálása nélkül.

A probléma érdekes. Az algoritmus/eljárás szavak részei mindennapi szóhasználatunknak, de a matematikában nincsenek értelmezve. Egy definíció megadása, akkor sikeres, ha a matematikus társadalom többsége rábólint: „elfogadom korrekt definíciónak". Az a pillanat, amihez ezt kötik az Church egy 1935. április 19-én tartott előadása az Amerikai Matematikai Társulat egy találkozóján.

Történeti megjegyzések (folytatás)

Történeti megjegyzések (folytatás)

A felhívást, hogy az algoritmus korrekt definíciójának keresése véget érhet (ott vagyunk a „szent grálnál”) *Church-tézisként* hivatkozzák.

Történeti megjegyzések (folytatás)

A felhívást, hogy az algoritmus korrekt definíciójának keresése véget érhet (ott vagyunk a „szent grálnál") *Church-tézisként* hivatkozzák.

Érdekes megjegyezni, hogy a tézis bejelentését megelőzte egy Gödelhez írt levél. Gödel a matematikai logika megkérdőjelezhetetlen óriása volt, aki szintén aktívan foglalkozott a kérdéssel. A Church levelében leírt kiszámíthatóság fogalmat Gödel nem fogadta el. Az 1935-ös változat ennek módosítása volt, Church kötni tudta saját fogalmát Gödel próbálkozásaihoz. De a tézisben megfogalmazott indokokat a matematikai „egyezséghez" Gödel nem tartotta kielégítőnek.

Történeti megjegyzések (folytatás)

A felhívást, hogy az algoritmus korrekt definíciójának keresése véget érhet (ott vagyunk a „szent grálnál") *Church-tézisként* hivatkozzák.

Érdekes megjegyezni, hogy a tézis bejelentését megelőzte egy Gödelhez írt levél. Gödel a matematikai logika megkérdőjelezhetetlen óriása volt, aki szintén aktívan foglalkozott a kérdéssel. A Church levelében leírt kiszámíthatóság fogalmat Gödel nem fogadta el. Az 1935-ös változat ennek módosítása volt, Church kötni tudta saját fogalmát Gödel próbálkozásaihoz. De a tézisben megfogalmazott indokokat a matematikai „egyezséghez" Gödel nem tartotta kielégítőnek.

Az igazi áttörést Turing egy 1936-os cikke okozta. Ebben egy új definíciót fogalmazott meg. Ezt már széles körben elfogadták mint a kiszámíthatósag/algoritmus fogalma.

Történeti megjegyzések (folytatás)

Történeti megjegyzések (folytatás)

Később a kutatók belátták, hogy Turing, Church, Gödel és mások próbálkozásai mind ekvivalens fogalomhoz vezetnek. A tézist mind a mai napig elfogadják.

Történeti megjegyzések (folytatás)

Később a kutatók belátták, hogy Turing, Church, Gödel és mások próbálkozásai mind ekvivalens fogalomhoz vezetnek. A tézist mind a mai napig elfogadják.

Ennek ellenére, ha valami technológiai áttörés történik, amely gyökeresen átalakítja a mindennapi képünket a számítás fogalmáról, akkor a tézist újra kell értékelni.

Történeti megjegyzések (folytatás)

Később a kutatók belátták, hogy Turing, Church, Gödel és mások próbálkozásai mind ekvivalens fogalomhoz vezetnek. A tézist mind a mai napig elfogadják.

Ennek ellenére, ha valami technológiai áttörés történik, amely gyökeresen átalakítja a mindennapi képünket a számítás fogalmáról, akkor a tézist újra kell értékelni.

Megemlítjük, hogy kvantum gépekkel is ugyanazon függvények lesznek kiszámíthatók (amennyiben technológialilag megvalósíthatók), mint klasszikus gépekkel.

Történeti megjegyzések (folytatás)

Később a kutatók belátták, hogy Turing, Church, Gödel és mások próbálkozásai mind ekvivalens fogalomhoz vezetnek. A tézist mind a mai napig elfogadják.

Ennek ellenére, ha valami technológiai áttörés történik, amely gyökeresen átalakítja a mindennapi képünket a számítás fogalmáról, akkor a tézist újra kell értékelni.

Megemlítjük, hogy kvantum gépekkel is ugyanazon függvények lesznek kiszámíthatók (amennyiben technológialilag megvalósíthatók), mint klasszikus gépekkel.

Turing definíciója a dinamikus szemléletét írja le. A fogalom a Turing-gép, amely konfigurációk egy sorozatán áthaladva jut el az outputig, hajt végre egy algoritmust. A Turing-gép definíciója több definíció eredője lesz.

Történeti megjegyzések (folytatás)

Később a kutatók belátták, hogy Turing, Church, Gödel és mások próbálkozásai mind ekvivalens fogalomhoz vezetnek. A tézist mind a mai napig elfogadják.

Ennek ellenére, ha valami technológiai áttörés történik, amely gyökeresen átalakítja a mindennapi képünket a számítás fogalmáról, akkor a tézist újra kell értékelni.

Megemlítjük, hogy kvantum gépekkel is ugyanazon függvények lesznek kiszámíthatók (amennyiben technológialilag megvalósíthatók), mint klasszikus gépekkel.

Turing definíciója a dinamikus szemléletét írja le. A fogalom a Turing-gép, amely konfigurációk egy sorozatán áthaladva jut el az outputig, hajt végre egy algoritmust. A Turing-gép definíciója több definíció eredője lesz. A definíciók szöveges részt és száraz, matematikai „desztillációt” tartalmaznak.

Szünet

TURING-GÉP KONFIGURÁCIÓJA

TURING-GÉP KONFIGURÁCIÓJA

Ennek „fizikai” részei: inputszalag, munkaszalag, outputszalag, fej.

TURING-GÉP KONFIGURÁCIÓJA

Ennek „fizikai” részei: inputszalag, munkaszalag, outputszalag, fej.

A szalagok mezők sorozatát tartalmazzák. A t szalag mezői $\{M_i^t\}_{i=0}^{\infty}$ (azaz M_{100}^{input} az inputszalag 100 indexű/101-edik mezője, M_0^{munka} a munkaszalag első/0 indexű mezője).

TURING-GÉP KONFIGURÁCIÓJA

Ennek „fizikai” részei: inputszalag, munkaszalag, outputszalag, fej.

A szalagok mezők sorozatát tartalmazzák. A t szalag mezői $\{M_i^t\}_{i=0}^{\infty}$ (azaz M_{100}^{input} az inputszalag 100 indexű/101-edik mezője, M_0^{munka} a munkaszalag első/0 indexű mezője).

A mezőket úgy kell elképzelni mint egy négyzethálós papír egy sorát, amelyik jobb irányban végtelen.

TURING-GÉP KONFIGURÁCIÓJA

Ennek „fizikai” részei: inputszalag, munkaszalag, outputszalag, fej.

A szalagok mezők sorozatát tartalmazzák. A t szalag mezői $\{M_i^t\}_{i=0}^{\infty}$ (azaz M_{100}^{input} az inputszalag 100 indexű/101-edik mezője, M_0^{munka} a munkaszalag első/0 indexű mezője).

A mezőket úgy kell elképzelni mint egy négyzethálós papír egy sorát, amelyik jobb irányban végtelen.

A mezők tartalma egy-egy karakter. Az input- és outputszalag mezői a feladat kódolásához használt Σ ábécé elemeit tartalmazzák.

TURING-GÉP KONFIGURÁCIÓJA

Ennek „fizikai” részei: inputszalag, munkaszalag, outputszalag, fej.

A szalagok mezők sorozatát tartalmazzák. A t szalag mezői $\{M_i^t\}_{i=0}^{\infty}$ (azaz M_{100}^{input} az inputszalag 100 indexű/101-edik mezője, M_0^{munka} a munkaszalag első/0 indexű mezője).

A mezőket úgy kell elképzelni mint egy négyzethálós papír egy sorát, amelyik jobb irányban végtelen.

A mezők tartalma egy-egy karakter. Az input- és outputszalag mezői a feladat kódolásához használt Σ ábécé elemeit tartalmazzák.

Az outputszalagra a kiszámolt $f(\omega) \in \Sigma^*$ kerül a számítás során. Az output már leírt részétől jobbra lévő mezőkön egy (új) $\smile$ üres/érintetlen karakter szerepel.

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

A munkaszalaghoz egy Γ ábécé-t használunk, emellett itt is szerepelhet az érintetlen karakter.

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

A munkaszalaghoz egy Γ ábécé-t használunk, emellett itt is szerepelhet az érintetlen karakter.

A fentiekben leírt karakterektől különböző „határolójeleink” is vannak: $\triangleright$ és $\triangleleft$. Ezek a szalagok „határmezőinek” „bevésett” tartalma (lásd későbbi formális leírás).

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

A munkaszalaghoz egy Γ ábécé-t használunk, emellett itt is szerepelhet az érintetlen karakter.

A fentiekben leírt karakterektől különböző „határolójeleink” is vannak: $\triangleright$ és $\triangleleft$. Ezek a szalagok „határmezőinek” „bevésett” tartalma (lásd későbbi formális leírás).

A fej a Turing-gép szalagaival szemmel, illetve kézzel/tollal kapcsolódik.

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

A munkaszalaghoz egy Γ ábécé-t használunk, emellett itt is szerepelhet az érintetlen karakter.

A fentiekben leírt karakterektől különböző „határolójeleink” is vannak: $\triangleright$ és $\triangleleft$. Ezek a szalagok „határmezőinek” „bevésett” tartalma (lásd későbbi formális leírás).

A fej a Turing-gép szalagaival szemmel, illetve kézzel/tollal kapcsolódik.

A „szem” egy mező értékét olvassa, a kéz a látott mezőt írhatja felül.

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

A munkaszalaghoz egy Γ ábécé-t használunk, emellett itt is szerepelhet az érintetlen karakter.

A fentiekben leírt karakterektől különböző „határolójeleink” is vannak: $\triangleright$ és $\triangleleft$. Ezek a szalagok „határmezőinek” „bevésett” tartalma (lásd későbbi formális leírás).

A fej a Turing-gép szalagaival szemmel, illetve kézzel/tollal kapcsolódik.

A „szem” egy mező értékét olvassa, a kéz a látott mezőt írhatja felül.

A gépnek egy input-, munka-szemmel, munka-kézzel és output-kézzel rendelkezik. Ezek helyzetét az írja le, hogy melyik mező felett helyezkednek el.

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

Azaz mindegyik szalaghoz tartozik egy-egy pozíció, amit a fej kontrolál (az inputszalag esetén ez az input-szem által látott mező, a munkaszalag esetén ez a munka-szem által látott és egyben a munka-kéz által írható mező).

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

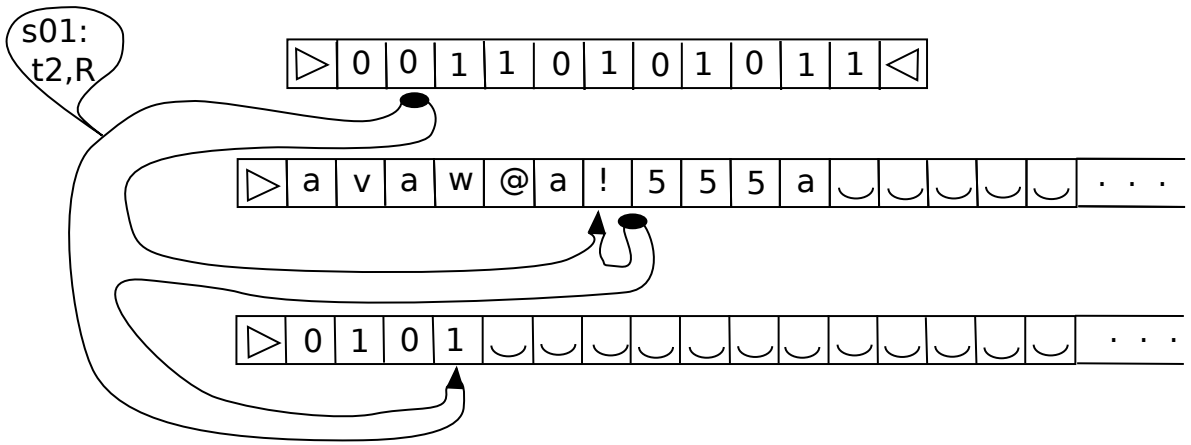
Azaz mindegyik szalaghoz tartozik egy-egy pozíció, amit a fej kontrolál (az inputszalag esetén ez az input-szem által látott mező, a munkaszalag esetén ez a munka-szem által látott és egyben a munka-kéz által írható mező).

A fejnek van egy bizonyos állapota. A lehetséges állapotok egy S véges halmazt alkotnak (S elemeit állapotoknak hívjuk, S az állapothalmaz).

TURING-GÉP KONFIGURÁCIÓJA (folytatás)

Azaz mindegyik szalaghoz tartozik egy-egy pozíció, amit a fej kontrolál (az inputszalag esetén ez az input-szem által látott mező, a munkaszalag esetén ez a munka-szem által látott és egyben a munka-kéz által írható mező).

A fejnek van egy bizonyos állapota. A lehetséges állapotok egy S véges halmazt alkotnak (S elemeit állapotoknak hívjuk, S az állapothalmaz).



Egy képzeletbeli gép képzeletbeli konfigurációja

TURING-GÉP KONFIGURÁCIÓJA: A definíció

Turing-gép konfigurációja

A konfiguráció egy n hosszú input mellett egy hetes:

$$\langle \{M_i^{input}\}_{i=0}^{n+1}, \{M_i^{munka}\}_{i=0}^{\infty}, \{M_i^{output}\}_{i=0}^{\infty}, p^{input}, p^{munka}, p^{output}, s \rangle,$$

ahol $M_0^{input} = M_0^{munka} = M_0^{output} = \triangleright$, $M_{n+1}^{input} = \triangleleft$,
$p^{input} \in \{0, 1, 2, \ldots, n+1\}$, $p^{munka}, p^{output} \in \mathbb{N}$, $s \in S$.

Előzetes megállapítások

Előzetes megállapítások

Néhány fontos megállapítást kell tennünk:

Előzetes megállapítások

Néhány fontos megállapítást kell tennünk:

1) A fej a konfigurációnak csak egy szűk részét „látja”. Ez a két szem által látott mező tartalma és az állapota (azaz $(\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S$ egy eleme).

Előzetes megállapítások

Néhány fontos megállapítást kell tennünk:

1) A fej a konfigurációnak csak egy szűk részét „látja”. Ez a két szem által látott mező tartalma és az állapota (azaz $(\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S$ egy eleme).
2) Az inputszalag funkciója csak az input tárolása. Ezt olvashatjuk az inputszemmel, de nem írhatjuk felül. Azt mondjuk az „inputszalag csak olvasható”.

Előzetes megállapítások

Néhány fontos megállapítást kell tennünk:

1) A fej a konfigurációnak csak egy szűk részét „látja”. Ez a két szem által látott mező tartalma és az állapota (azaz $(\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S$ egy eleme).
2) Az inputszalag funkciója csak az input tárolása. Ezt olvashatjuk az inputszemmel, de nem írhatjuk felül. Azt mondjuk az „inputszalag csak olvasható”.
3) A munkaszalag a „munka helye”. Ide kerülhetnek, „feljegyzések”, „részeredmények”, „részletszámítások”. Ezt a részét a konfigurációnak írhatjuk, „radírozhatjuk”.

Előzetes megállapítások

Néhány fontos megállapítást kell tennünk:

1) A fej a konfigurációnak csak egy szűk részét „látja”. Ez a két szem által látott mező tartalma és az állapota (azaz $(\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S$ egy eleme).
2) Az inputszalag funkciója csak az input tárolása. Ezt olvashatjuk az inputszemmel, de nem írhatjuk felül. Azt mondjuk az „inputszalag csak olvasható”.
3) A munkaszalag a „munka helye”. Ide kerülhetnek, „feljegyzések”, „részeredmények”, „részletszámítások”. Ezt a részét a konfigurációnak írhatjuk, „radírozhatjuk”. Az algoritmus a munkaszalagot csak lokálisan írja felül. Csak a munkaszalag látható mezőjére (aktuális pozíciójába) lehet írni. A munkaszalag tartalmát a munkaszem látja.

Előzetes megállapítások

Néhány fontos megállapítást kell tennünk:

1) A fej a konfigurációnak csak egy szűk részét „látja”. Ez a két szem által látott mező tartalma és az állapota (azaz $(\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S$ egy eleme).
2) Az inputszalag funkciója csak az input tárolása. Ezt olvashatjuk az inputszemmel, de nem írhatjuk felül. Azt mondjuk az „inputszalag csak olvasható”.
3) A munkaszalag a „munka helye”. Ide kerülhetnek, „feljegyzések”, „részeredmények”, „részletszámítások”. Ezt a részét a konfigurációnak írhatjuk, „radírozhatjuk”. Az algoritmus a munkaszalagot csak lokálisan írja felül. Csak a munkaszalag látható mezőjére (aktuális pozíciójába) lehet írni. A munkaszalag tartalmát a munkaszem látja. Azt mondjuk az „munkaszalag olvasható/írható”.

Előzetes megállapítások (folytatás)

Előzetes megállapítások (folytatás)

4) A szemek (és velük a kezek) mozgása „folytonos”. Ez alatt azt értjük, hogy mindegyik szalagon az új pozíció legfeljebb 1-gyel tér el az előzőtől.

Előzetes megállapítások (folytatás)

4) A szemek (és velük a kezek) mozgása „folytonos”. Ez alatt azt értjük, hogy mindegyik szalagon az új pozíció legfeljebb 1-gyel tér el az előzőtől.
5) Az outputszalag funkciója csak az output leírása. Ezt a következő megállapodással garantáljuk: Az outpuszalagra akkor írunk majd, ha a kéz egyet jobbra mozdul. Más mozgást nem is engedélyezünk. Azt mondjuk az „outputszalag csak írható”.

Előzetes megállapítások (folytatás)

4) A szemek (és velük a kezek) mozgása „folytonos”. Ez alatt azt értjük, hogy mindegyik szalagon az új pozíció legfeljebb 1-gyel tér el az előzőtől.
5) Az outputszalag funkciója csak az output leírása. Ezt a következő megállapodással garantáljuk: Az outpuszalagra akkor írunk majd, ha a kéz egyet jobbra mozdul. Más mozgást nem is engedélyezünk. Azt mondjuk az „outputszalag csak írható”.
6) A határoló jelek ($\triangleright$ és $\triangleleft$) szerepe a szemek/kezek szalag fölött tartása.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: A definíció

TURING-GÉP ÁTMENETIFÜGGVÉNYE: A definíció

A fenti megjegyzések alapján egy konfiguráció változtatását egy egyszerű függvény írja le.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: A definíció

A fenti megjegyzések alapján egy konfiguráció változtatását egy egyszerű függvény írja le.

Definíció: Turing-gép átmenetifüggvénye

Egy Turing-gép *átmenetifüggvénye* egy

$$\delta : (\Sigma \cup \{\triangleright, \triangleleft\}) \times (\Gamma \cup \{\triangleright, \smile\}) \times S \to \{B, ., J\} \times \Gamma \times \{B, ., J\} \times (\{.\} \cup \Sigma) \times S.$$

függvény.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Feltételek

A 6) megjegyzést külön feltételek fogalmazzák meg:

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Feltételek

A 6) megjegyzést külön feltételek fogalmazzák meg:

- Azaz $\delta(\triangleright, karakter, STATE)$-nek olyan értéknek kell lenni, hogy első koordinátája nem B. Második koordinátája lényegtelen mert a határoló jel nem írható felül, a rákövetkező konfigurációban a munkaszalag tartalma ugyanaz lesz mint korábban.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Feltételek

A 6) megjegyzést külön feltételek fogalmazzák meg:

- Azaz $\delta(\triangleright, karakter, STATE)$-nek olyan értéknek kell lenni, hogy első koordinátája nem B. Második koordinátája lényegtelen mert a határoló jel nem írható felül, a rákövetkező konfigurációban a munkaszalag tartalma ugyanaz lesz mint korábban.
- Azaz $\delta(\triangleleft, karakter, STATE)$-nek olyan értéknek kell lenni, hogy első koordinátája nem J.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Feltételek

A 6) megjegyzést külön feltételek fogalmazzák meg:

- Azaz $\delta(\triangleright, \mathit{karakter}, \mathit{STATE})$-nek olyan értéknek kell lenni, hogy első koordinátája nem B. Második koordinátája lényegtelen mert a határoló jel nem írható felül, a rákövetkező konfigurációban a munkaszalag tartalma ugyanaz lesz mint korábban.
- Azaz $\delta(\triangleleft, \mathit{karakter}, \mathit{STATE})$-nek olyan értéknek kell lenni, hogy első koordinátája nem J.
- $\delta(\mathit{karakter}, \triangleright, \mathit{STATE})$-nek olyan értéknek kell lenni, hogy harmadik koordinátája nem B.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Feltételek

A 6) megjegyzést külön feltételek fogalmazzák meg:

- Azaz $\delta(\triangleright, \mathit{karakter}, \mathit{STATE})$-nek olyan értéknek kell lenni, hogy első koordinátája nem B. Második koordinátája lényegtelen mert a határoló jel nem írható felül, a rákövetkező konfigurációban a munkaszalag tartalma ugyanaz lesz mint korábban.
- Azaz $\delta(\triangleleft, \mathit{karakter}, \mathit{STATE})$-nek olyan értéknek kell lenni, hogy első koordinátája nem J.
- $\delta(\mathit{karakter}, \triangleright, \mathit{STATE})$-nek olyan értéknek kell lenni, hogy harmadik koordinátája nem B.

A teljes feltételrendszer felírását nem végeztük el. Mindenki megteheti, aki úgy érzi, hogy a formalizálás segíti megértését.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

Az átmeneti függvény értékkészletében minden értéknek öt komponense van. Ezek rendere a következőt írják le:

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

Az átmeneti függvény értékkészletében minden értéknek öt komponense van. Ezek rendere a következőt írják le:

(i) input-szem mozgása ('B'= bal szomszédra ugrás, '.' = maradás, 'J' = jobb szomszédra ugrás),

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

Az átmeneti függvény értékkészletében minden értéknek öt komponense van. Ezek rendere a következőt írják le:

(i) input-szem mozgása ('B'= bal szomszédra ugrás, '.' = maradás, 'J' = jobb szomszédra ugrás),

(ii) munka-kéz által leírt karakter (a nem írás a már ott lévő karakter újraírása),

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

Az átmeneti függvény értékkészletében minden értéknek öt komponense van. Ezek rendere a következőt írják le:

(i) input-szem mozgása ('B'= bal szomszédra ugrás, '.' = maradás, 'J' = jobb szomszédra ugrás),

(ii) munka-kéz által leírt karakter (a nem írás a már ott lévő karakter újraírása),

(iii) munka-szem mozgása (ami egyben a kéz mozgása is lásd az input-szem mozgására vonatkozó magyarázatot),

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

Az átmeneti függvény értékkészletében minden értéknek öt komponense van. Ezek rendere a következőt írják le:

(i) input-szem mozgása ('B'= bal szomszédra ugrás, '.' = maradás, 'J' = jobb szomszédra ugrás),

(ii) munka-kéz által leírt karakter (a nem írás a már ott lévő karakter újraírása),

(iii) munka-szem mozgása (ami egyben a kéz mozgása is lásd az input-szem mozgására vonatkozó magyarázatot),

(iv) az output-kéz mozgása és írása egybeolvasztva (a kéz vagy marad és nem ír (.), vagy jobbra mozog és ír egy karaktert (Σ egy eleme)),

TURING-GÉP ÁTMENETIFÜGGVÉNYE: Értelmezése

Az átmenetifüggvény értelmezési tartománya az 1) megjegyzés alatt leírtakat formalizálja.

Az átmeneti függvény értékkészletében minden értéknek öt komponense van. Ezek rendere a következőt írják le:

(i) input-szem mozgása ('B'= bal szomszédra ugrás, '.' = maradás, 'J' = jobb szomszédra ugrás),

(ii) munka-kéz által leírt karakter (a nem írás a már ott lévő karakter újraírása),

(iii) munka-szem mozgása (ami egyben a kéz mozgása is lásd az input-szem mozgására vonatkozó magyarázatot),

(iv) az output-kéz mozgása és írása egybeolvasztva (a kéz vagy marad és nem ír (.), vagy jobbra mozog és ír egy karaktert (Σ egy eleme)),

(v) az új konfigurációban a fej állapota.

Rákövetkező konfiguráció

Rákövetkező konfiguráció

Az értelmezés után egyszerű gyakorlat κ-ból kiolvasni a konfiguráció „látott részét”.

Rákövetkező konfiguráció

Az értelmezés után egyszerű gyakorlat κ-ból kiolvasni a konfiguráció „látott részét”.

Egyszerű venni az átmeneti függvény ezen felvett értékét és ez alapján módosítnai κ-t.

Rákövetkező konfiguráció

Az értelmezés után egyszerű gyakorlat κ-ból kiolvasni a konfiguráció „látott részét”.

Egyszerű venni az átmeneti függvény ezen felvett értékét és ez alapján módosítnai κ-t.

Definíció: Rákövetkező konfiguráció

κ-ból a fenti módon nyert konfiguráció κ *konfiguráció* κ^+ *rákövetkezője*.

Rákövetkező konfiguráció

Az értelmezés után egyszerű gyakorlat κ-ból kiolvasni a konfiguráció „látott részét”.

Egyszerű venni az átmeneti függvény ezen felvett értékét és ez alapján módosítnai κ-t.

Definíció: Rákövetkező konfiguráció

κ-ból a fenti módon nyert konfiguráció κ *konfiguráció* κ^+ *rákövetkezője*.

A formális leírást az olvasóra bízzuk.

TURING-GÉP KEZDŐKONFIGURÁCIÓJA

TURING-GÉP KEZDŐKONFIGURÁCIÓJA

Leírunk egy ω inputhoz tartozó konfigurációt. Ebből a konfigurációból indul az ω inputhoz tartozó számítás.

TURING-GÉP KEZDŐKONFIGURÁCIÓJA

Leírunk egy ω inputhoz tartozó konfigurációt. Ebből a konfigurációból indul az ω inputhoz tartozó számítás.

Turing-gép adott inputhoz tartozó kezdőkonfigurációja

Az $\omega \in \Sigma^n$ inputhoz tartozó kezdőkonfiguráció definíciójához szükséges néhány előzetes megállapodás. Legyen $START \in S$ egy speciális állapot (a számítás kezdetét jelző állapota gépünknek) és legyen $\smile$ egy speciális eleme Γ-nak (az üres karakter). Legyen

$$\kappa_0(\omega) = \langle \{M_i^{input}\}_{i=0}^{n+1}, \{M_i^{munka}\}_{i=0}^{\infty}, \{M_i^{output}\}_{i=0}^{\infty}, p^{input}, p^{munka}, p^{output}, s \rangle,$$

ahol $M_0^{input} = M_0^{munka} = M_0^{output} = \triangleright$, $M_{n+1}^{input} = \triangleleft$, $M_i^{input} = \omega_i$ $(i = 1, 2, \ldots, n)$, $M_i^{munka} = M_i^{output} = \smile$ $(i \in \mathbb{N}_+)$, $p^{input} = p^{munka} = p^{output} = 0$, $s = START$.

TURING-GÉP FUTÁSA

TURING-GÉP FUTÁSA

Ezekután már természetes az algoritmus futásának dinamikus képét leírni.

TURING-GÉP FUTÁSA

Ezekután már természetes az algoritmus futásának dinamikus képét leírni.

Turing-gép futása adott inputon

Legyen $\{\kappa_i\}_{i=0}^{\infty}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$. Ezt a konfiguráció-sorozatot az ω *inputhoz tartozó futásnak* nevezzük.

TURING-GÉP FUTÁSA

Ezekután már természetes az algoritmus futásának dinamikus képét leírni.

Turing-gép futása adott inputon

Legyen $\{\kappa_i\}_{i=0}^{\infty}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$. Ezt a konfiguráció-sorozatot az ω *inputhoz tartozó futásnak* nevezzük.

A fenti definíció egy végtelen konfigurációsorozatot nevez futásnak. Az igazi algoritmus azonban egy ponton leáll, a számítást befejezte, az eredmény kihirdeti.

Csonkított futás

Csonkított futás

(STOP)

Legyen $STOP$ egy speciális állapot, azaz $STOP \in S$. Ennek szerepe a számítás végének jelölése.

Csonkított futás

(STOP)

Legyen $STOP$ egy speciális állapot, azaz $STOP \in S$. Ennek szerepe a számítás végének jelölése.

A futás végtelen sorozatát bizonyos esetekben „levágjuk”.

Csonkított futás

(STOP)

Legyen *STOP* egy speciális állapot, azaz $STOP \in S$. Ennek szerepe a számítás végének jelölése.

A futás végtelen sorozatát bizonyos esetekben „levágjuk”.

Legyen $\{\kappa_i\}_{i=0}^{\ell}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$, továbbá $\ell = \min\{i : \kappa_i \text{ állapota } STOP\}$.

Csonkított futás

(STOP)

Legyen *STOP* egy speciális állapot, azaz $STOP \in S$. Ennek szerepe a számítás végének jelölése.

A futás végtelen sorozatát bizonyos esetekben „levágjuk”.

Legyen $\{\kappa_i\}_{i=0}^{\ell}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$, továbbá $\ell = \min\{i : \kappa_i \text{ állapota } STOP\}$. Amennyiben a minimum mögött álló halmaz üres $\ell = \infty$.

Csonkított futás

(STOP)

Legyen *STOP* egy speciális állapot, azaz $STOP \in S$. Ennek szerepe a számítás végének jelölése.

A futás végtelen sorozatát bizonyos esetekben „levágjuk”.

Legyen $\{\kappa_i\}_{i=0}^{\ell}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$, továbbá $\ell = \min\{i : \kappa_i \text{ állapota } STOP\}$. Amennyiben a minimum mögött álló halmaz üres $\ell = \infty$.

Ha $\ell < \infty$, akkor azt mondjuk *a futás véges/leáll*. Ezt a futást redukált futásnak nevezzük.

Csonkított futás

(STOP)

Legyen *STOP* egy speciális állapot, azaz $STOP \in S$. Ennek szerepe a számítás végének jelölése.

A futás végtelen sorozatát bizonyos esetekben „levágjuk”.

Legyen $\{\kappa_i\}_{i=0}^{\ell}$ konfigurációk azon sorozata, amelyre $\kappa_0 = \kappa_0(\omega)$ (az ω-hoz tartozó kezdőkonfiguráció) és $\kappa_{i+1} = \kappa_i^+$, továbbá $\ell = \min\{i : \kappa_i \text{ állapota } STOP\}$. Amennyiben a minimum mögött álló halmaz üres $\ell = \infty$.

Ha $\ell < \infty$, akkor azt mondjuk *a futás véges/leáll*. Ezt a futást redukált futásnak nevezzük.

Ha ω-n gépünk futása leáll, akkor κ_ℓ-ben az outputszalag tartalma a $\triangleright$ jel után az output-kézig tartalmazza a *kiszámított outputot*.

TURING-GÉP/ALGORITMUS ÖSSZETEVŐI

TURING-GÉP/ALGORITMUS ÖSSZETEVŐI

Tulajdonképpen most értünk a Turing-gép definiciójának végére.
Egy $f : \Sigma^* \to \Sigma^*$ feladathoz tartozó Turing-géphez szükségünk van

TURING-GÉP/ALGORITMUS ÖSSZETEVŐI

Tulajdonképpen most értünk a Turing-gép definiciójának végére.
Egy $f : \Sigma^* \to \Sigma^*$ feladathoz tartozó Turing-géphez szükségünk van

(1) egy Γ munka ábécére,

TURING-GÉP/ALGORITMUS ÖSSZETEVŐI

Tulajdonképpen most értünk a Turing-gép definiciójának végére.
Egy $f : \Sigma^* \to \Sigma^*$ feladathoz tartozó Turing-géphez szükségünk van

(1) egy Γ munka ábécére,

(2) speciális karakterekre ($\triangleright$, $\triangleleft$, $\smile$),

TURING-GÉP/ALGORITMUS ÖSSZETEVŐI

Tulajdonképpen most értünk a Turing-gép definiciójának végére.
Egy $f : \Sigma^* \to \Sigma^*$ feladathoz tartozó Turing-géphez szükségünk van

(1) egy Γ munka ábécére,

(2) speciális karakterekre ($\triangleright$, $\triangleleft$, $\smile$),

(3) egy S állapothalmazra (speciális állapotokkal: START, STOP),

TURING-GÉP/ALGORITMUS ÖSSZETEVŐI

Tulajdonképpen most értünk a Turing-gép definiciójának végére.
Egy $f : \Sigma^* \to \Sigma^*$ feladathoz tartozó Turing-géphez szükségünk van

(1) egy Γ munka ábécére,

(2) speciális karakterekre ($\triangleright$, $\triangleleft$, $\smile$),

(3) egy S állapothalmazra (speciális állapotokkal: START, STOP),

(4) egy δ átmeneti függvényre.

TURING-GÉP/ALGORITMUS ÖSSZETEVŐI

Tulajdonképpen most értünk a Turing-gép definiciójának végére. Egy $f : \Sigma^* \to \Sigma^*$ feladathoz tartozó Turing-géphez szükségünk van

(1) egy Γ munka ábécére,

(2) speciális karakterekre ($\triangleright$, $\triangleleft$, $\smile$),

(3) egy S állapothalmazra (speciális állapotokkal: START, STOP),

(4) egy δ átmeneti függvényre.

Fontos észrevenni, hogy egy konkrét algoritmus/Turing-gép például 2020 állapotot használ és munka-ábécéje 1001 karakteres, DE minden inputhosszra el kell végeznie a „munkáját”.

TURING-GÉP/ALGORITMUS ÁLTAL KISZÁMOLT FÜGGVÉNY

TURING-GÉP/ALGORITMUS ÁLTAL KISZÁMOLT FÜGGVÉNY

Definíció

Egy f függvényt kiszámít egy T Turing-gép, ha minden $\omega \in \Sigma^*$ esetén leáll a Turing-gép és a kiszámított érték $f(\omega)$.

TURING-GÉP/ALGORITMUS ÁLTAL KISZÁMOLT FÜGGVÉNY

Definíció

Egy f függvényt kiszámít egy T Turing-gép, ha minden $\omega \in \Sigma^*$ esetén leáll a Turing-gép és a kiszámított érték $f(\omega)$.

Másképpen is fogalmazhatunk.

TURING-GÉP/ALGORITMUS ÁLTAL KISZÁMOLT FÜGGVÉNY

Definíció

Egy f függvényt kiszámít egy T Turing-gép, ha minden $\omega \in \Sigma^*$ esetén leáll a Turing-gép és a kiszámított érték $f(\omega)$.

Másképpen is fogalmazhatunk.

Definíció

Minden T Turing-gép kiszámol egy $f_T : \Sigma^* \to \Sigma^* \cup \{\infty\}$ függvényt. $\omega \in \Sigma^*$ esetén $f_T(\omega) = \infty$, ha a futás nem véges, míg a kiszámolt Σ^*-beli érték, ha a futás véges ω-n.

TURING-GÉP/ALGORITMUS ÁLTAL KISZÁMOLT FÜGGVÉNY

Definíció

Egy f függvényt kiszámít egy T Turing-gép, ha minden $\omega \in \Sigma^*$ esetén leáll a Turing-gép és a kiszámított érték $f(\omega)$.

Másképpen is fogalmazhatunk.

Definíció

Minden T Turing-gép kiszámol egy $f_T : \Sigma^* \to \Sigma^* \cup \{\infty\}$ függvényt. $\omega \in \Sigma^*$ esetén $f_T(\omega) = \infty$, ha a futás nem véges, míg a kiszámolt Σ^*-beli érték, ha a futás véges ω-n.

T akkor számol ki egy f függvényt, ha $f = f_T$.

Kiszámítható függvények

Kiszámítható függvények

Definíció

Egy $f : \Sigma^* \to \Sigma^*$ függvény akkor *kiszámítható*, ha alkalmas T Turing-gépre $f = f_T$.

Kiszámítható függvények

Definíció

Egy $f : \Sigma^* \to \Sigma^*$ függvény akkor *kiszámítható*, ha alkalmas T Turing-gépre $f = f_T$.

Egy kiszámítható függvényhez sok megfelelő T Turing-gép/algoritmus létezik.

Kiszámítható függvények

Definíció

Egy $f : \Sigma^* \to \Sigma^*$ függvény akkor *kiszámítható*, ha alkalmas T Turing-gépre $f = f_T$.

Egy kiszámítható függvényhez sok megfelelő T Turing-gép/algoritmus létezik.

Előre megemlítjük, hogy vannak nem kiszámítható függvények.

ELDÖNTŐ TURING-GÉPEK

ELDÖNTŐ TURING-GÉPEK

Eldöntési feladat megoldására szolgáló Turing-gép esetén az outputszalag egyetlen bit leírására szolgál.

ELDÖNTŐ TURING-GÉPEK

Eldöntési feladat megoldására szolgáló Turing-gép esetén az outputszalag egyetlen bit leírására szolgál.

A definíció egyszerűsíthető.

ELDÖNTŐ TURING-GÉPEK

Eldöntési feladat megoldására szolgáló Turing-gép esetén az outputszalag egyetlen bit leírására szolgál.

A definíció egyszerűsíthető. Az outputszalag eldobható.

ELDÖNTŐ TURING-GÉPEK

Eldöntési feladat megoldására szolgáló Turing-gép esetén az outputszalag egyetlen bit leírására szolgál.

A definíció egyszerűsíthető. Az outputszalag eldobható. A STOP állapotot helyettesítsük ELFOGAD és ELVET állapotokkal.

Definíció

Döntési feladatoknál a fenti modellel dolgozunk, ezt *eldöntő Turing-gépnek* nevezzük.

ELDÖNTŐ TURING-GÉPEK

Eldöntési feladat megoldására szolgáló Turing-gép esetén az outputszalag egyetlen bit leírására szolgál.

A definíció egyszerűsíthető. Az outputszalag eldobható. A STOP állapotot helyettesítsük ELFOGAD és ELVET állapotokkal.

Definíció

Döntési feladatoknál a fenti modellel dolgozunk, ezt *eldöntő Turing-gépnek* nevezzük.

Ekkor a futás akkor és csak akkor áll le, ha ELVET vagy ELFOGAD állapotba kerül.

ELDÖNTŐ TURING-GÉPEK

Eldöntési feladat megoldására szolgáló Turing-gép esetén az outputszalag egyetlen bit leírására szolgál.

A definíció egyszerűsíthető. Az outputszalag eldobható. A STOP állapotot helyettesítsük ELFOGAD és ELVET állapotokkal.

Definíció

Döntési feladatoknál a fenti modellel dolgozunk, ezt *eldöntő Turing-gépnek* nevezzük.

Ekkor a futás akkor és csak akkor áll le, ha ELVET vagy ELFOGAD állapotba kerül.

Definíció

Egy eldöntő T Turing-gép eldönti az L nyelvet, ha minden $\omega \in L$ esetén ELFOGAD állapottal áll le a gép és minden $\omega \notin L$ esetén ELVET állapottal áll le a gép. (Speciálisan minden inputon leáll a gép.)

Eldönthető nyelvek

Eldönthető nyelvek

Definíció

Egy $L \subset \Sigma^*$ nyelv/döntési feladat eldönthető ha van olyan Turing-gép, ami eldönti.

Eldönthető nyelvek

Definíció

Egy $L \subset \Sigma^*$ nyelv/döntési feladat eldönthető ha van olyan Turing-gép, ami eldönti.

Az eldönthető nyelvek halmaza legyen $\mathcal{D}(\Sigma)$.

Felsorolható nyelvek

Felsorolható nyelvek

Megemlítünk egy további lehetőséget az elfogadás/elvetés „kódolására”.

Felsorolható nyelvek

Megemlítünk egy további lehetőséget az elfogadás/elvetés „kódolására”.

Definíció

Egy eldöntő T Turing-gép *felsorolja* az L nyelvet, ha minden $\omega \in L$ esetén ELFOGAD állapottal áll le a gép és minden $\omega \notin L$ esetén nem áll le.

Felsorolható nyelvek

Megemlítünk egy további lehetőséget az elfogadás/elvetés „kódolására”.

Definíció

Egy eldöntő T Turing-gép *felsorolja* az L nyelvet, ha minden $\omega \in L$ esetén ELFOGAD állapottal áll le a gép és minden $\omega \notin L$ esetén nem áll le.

Definíció

Egy $L \subset \Sigma^*$ nyelv felsorolható ha van olyan Turing-gép, ami felsorolja. Az eldönthető nyelvek halmaza legyen $\mathcal{S}(\Sigma)$.

Felsorolható nyelvek

Megemlítünk egy további lehetőséget az elfogadás/elvetés „kódolására”.

Definíció

Egy eldöntő T Turing-gép *felsorolja* az L nyelvet, ha minden $\omega \in L$ esetén ELFOGAD állapottal áll le a gép és minden $\omega \notin L$ esetén nem áll le.

Definíció

Egy $L \subset \Sigma^*$ nyelv felsorolható ha van olyan Turing-gép, ami felsorolja. Az eldönthető nyelvek halmaza legyen $\mathcal{S}(\Sigma)$.

Adott Σ ábécé esetén nyilván

$$\mathcal{D}(\Sigma) \subset \mathcal{S}(\Sigma) \subset \mathcal{P}(\Sigma^*).$$

Szünet

Alapmodell

Alapmodell

A fent leírt modell esetleges, hiszen például az, hogy egy irányban végtelen szalagokat használunk továbbá, hogy éppen három szalagon dolgozunk önkényes választások. Így természetesen léteznek más változatok is.

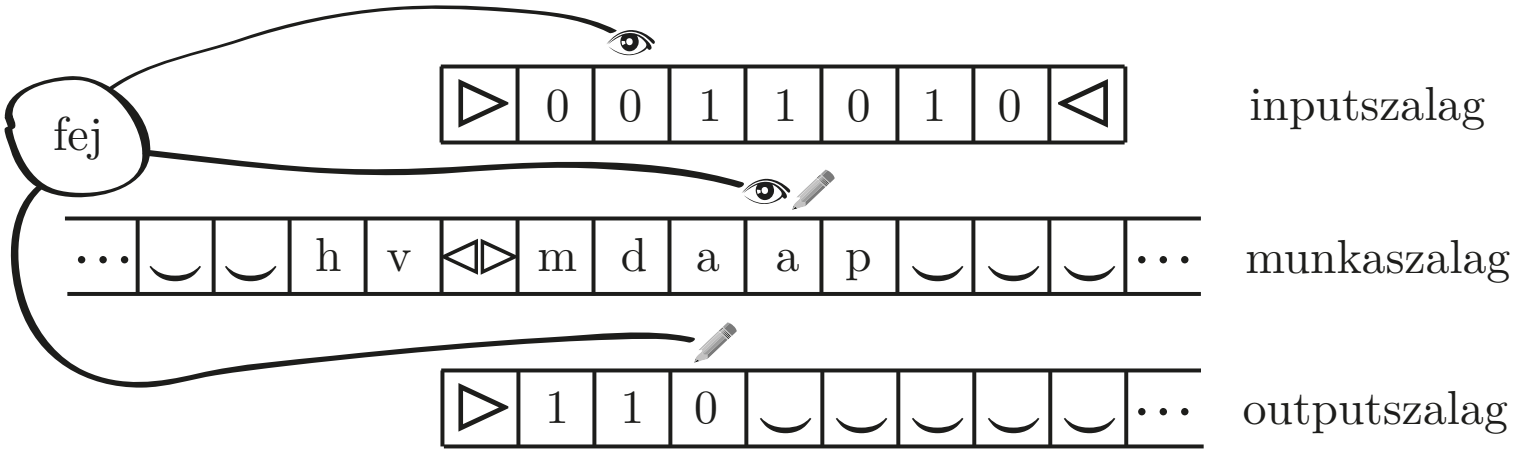
Változatok: Két irányban végtelen munkaszalag

Változatok: Két irányban végtelen munkaszalag

A munkaszalag bal és jobb irányban is végtelen és nem tartalmaz „szalag eleje” jelet, csupán egy kezdeti mezőt kell megadni.

Változatok: Két irányban végtelen munkaszalag

A munkaszalag bal és jobb irányban is végtelen és nem tartalmaz „szalag eleje” jelet, csupán egy kezdeti mezőt kell megadni.



A két irányban végtelen munkaszalag.

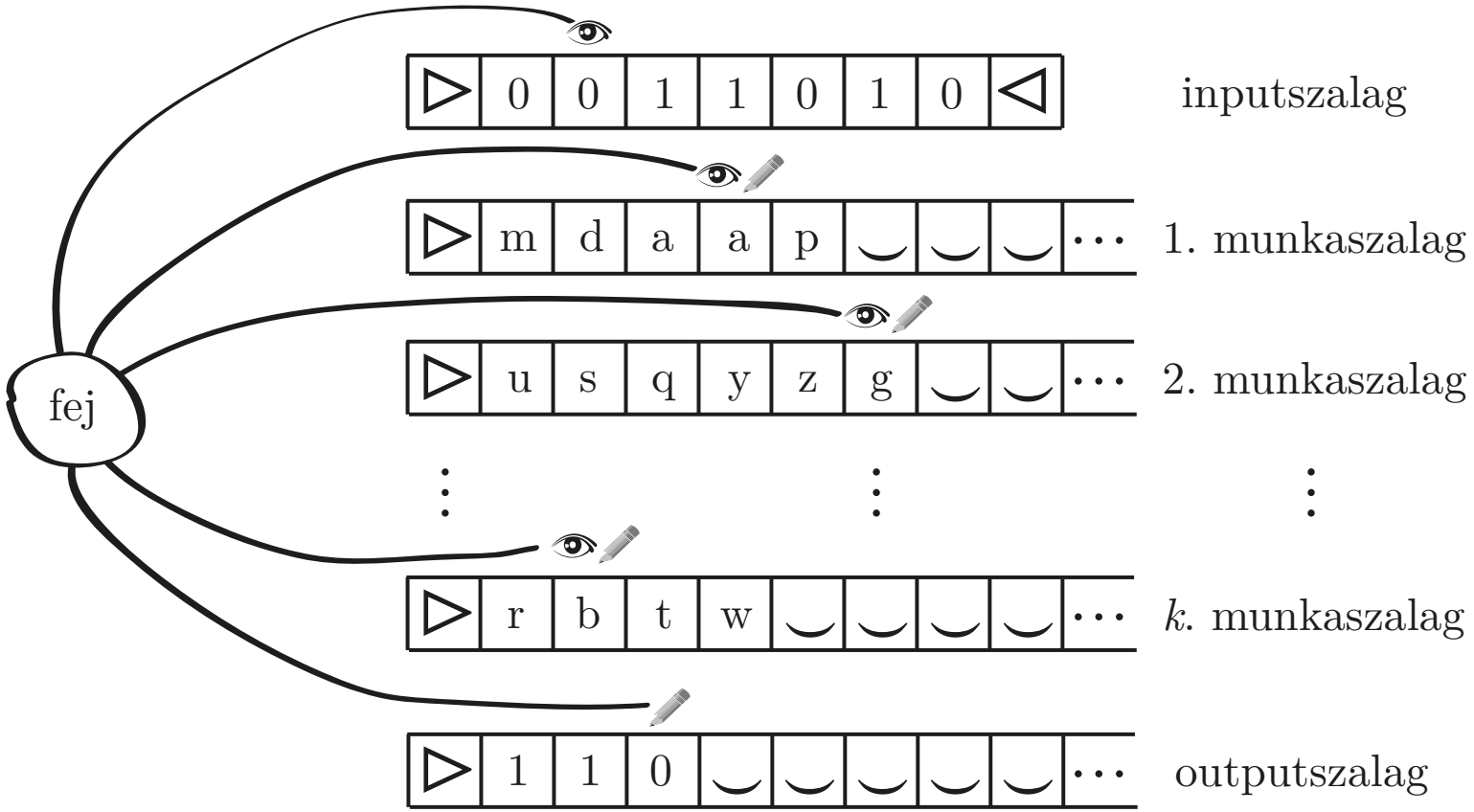
Változatok: k-szalagos modell

Változatok: k-szalagos modell

Rögzített $k \in \mathbb{N}$ számú munkaszalagot használunk, és minden egyes munkaszalaghoz tartozik egy szem/kéz páros. Fontos megjegyezni, hogy k nem függ az input méretétől.

Változatok: k-szalagos modell

Rögzített $k \in \mathbb{N}$ számú munkaszalagot használunk, és minden egyes munkaszalaghoz tartozik egy szem/kéz páros. Fontos megjegyezni, hogy k nem függ az input méretétől.



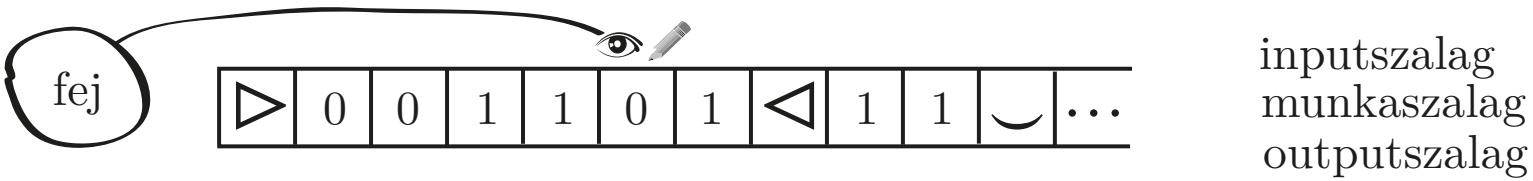
Változatok: Egyszalagos modell

Változatok: Egyszalagos modell

Minden művelet egyetlen szalagon történik, amely így egyszerre input-, munka-, és outputszalag. A szalag jobbra végtelen és a kezdeti mezőket az input foglalja el $\triangleright$,$\triangleleft$ jelek között. Ebben a modellben az input is felülírható és az output a szalag tartalma a $STOP \in S$ állapot elérésekor.

Változatok: Egyszalagos modell

Minden művelet egyetlen szalagon történik, amely így egyszerre input-, munka-, és outputszalag. A szalag jobbra végtelen és a kezdeti mezőket az input foglalja el $\triangleright$,$\triangleleft$ jelek között. Ebben a modellben az input is felülírható és az output a szalag tartalma a $STOP \in S$ állapot elérésekor.



Az egyszalagos modell.

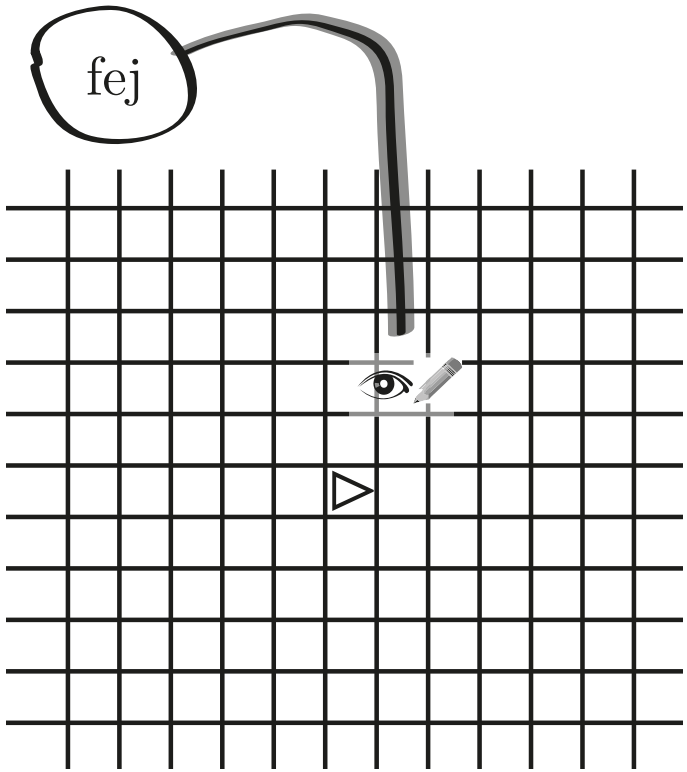
Változatok: Kétdimenziós munkaszalag

Változatok: Kétdimenziós munkaszalag

Érdemes a munkaszalagot úgy elképzelni, mint egy végtelen négyzethálós füzetlapot (ez lehet a teljes síkot lefedő háló vagy csupán egy negyedsíknak megfelelő). Ebben az esetben is ki kell jelölnünk egy kezdeti mezőt.

Változatok: Kétdimenziós munkaszalag

Érdemes a munkaszalagot úgy elképzelni, mint egy végtelen négyzethálós füzetlapot (ez lehet a teljes síkot lefedő háló vagy csupán egy negyedsíknak megfelelő). Ebben az esetben is ki kell jelölnünk egy kezdeti mezőt.



Robosztusság

Robosztusság

Nyilvánvaló, hogy a különböző modellekhez más látható részek tartoznak, ebből adódóan pedig eltérőek a modellek átmeneti függvényei. A szükséges módosítások formalizálása nem nehéz, de időigényes; ezt az érdeklődő hallgatókra bízzuk

Robosztusság

Nyilvánvaló, hogy a különböző modellekhez más látható részek tartoznak, ebből adódóan pedig eltérőek a modellek átmeneti függvényei. A szükséges módosítások formalizálása nem nehéz, de időigényes; ezt az érdeklődő hallgatókra bízzuk

Fontos kérdés, hogy az aprónak tűnő változtatások nem változtatják-e meg a kiszámíthatóságot.

Robosztusság

Nyilvánvaló, hogy a különböző modellekhez más látható részek tartoznak, ebből adódóan pedig eltérőek a modellek átmeneti függvényei. A szükséges módosítások formalizálása nem nehéz, de időigényes; ezt az érdeklődő hallgatókra bízzuk

Fontos kérdés, hogy az aprónak tűnő változtatások nem változtatják-e meg a kiszámíthatóságot.

Fontos tétel: NEM.

Robosztusság

Nyilvánvaló, hogy a különböző modellekhez más látható részek tartoznak, ebből adódóan pedig eltérőek a modellek átmeneti függvényei. A szükséges módosítások formalizálása nem nehéz, de időigényes; ezt az érdeklődő hallgatókra bízzuk

Fontos kérdés, hogy az aprónak tűnő változtatások nem változtatják-e meg a kiszámíthatóságot.

Fontos tétel: NEM.

Igazából nem is nehéz: bármelyik változatra alapítunk egy algoritmust/Turing-gépet az általa elvégzett feladat megoldható egy másik modellben (szimulálható). Összegezve: A korábban bevezetett függvény-/nyelv-osztályok nem függnek a fenti típusú módosításoktól.

Standard modell

Standard modell

A bonyolultságelmélet „feladata” a $\mathcal{D}$ nyelvosztály finomítása, az ide tartozó nyelvek vizsgálata. $\mathcal{D}$-n kívüli nyelvek összehasonlítása és vizsgálata inkább matematikai logika jellegű kérdésekhez vezet.

Standard modell

A bonyolultságelmélet „feladata” a $\mathcal{D}$ nyelvosztály finomítása, az ide tartozó nyelvek vizsgálata. $\mathcal{D}$-n kívüli nyelvek összehasonlítása és vizsgálata inkább matematikai logika jellegű kérdésekhez vezet.

A matematikai kérdésekhez jó, ha megállapodunk a modellben.

Standard modell

A bonyolultságelmélet „feladata” a $\mathcal{D}$ nyelvosztály finomítása, az ide tartozó nyelvek vizsgálata. $\mathcal{D}$-n kívüli nyelvek összehasonlítása és vizsgálata inkább matematikai logika jellegű kérdésekhez vezet.

A matematikai kérdésekhez jó, ha megállapodunk a modellben.

Számunkra az elkülönített munkaszalag fontos lesz. Ez alapján tudunk majd beszélni „logaritmikus tárban” kiszámítható függvényről.

Standard modell

A bonyolultságelmélet „feladata” a $\mathcal{D}$ nyelvosztály finomítása, az ide tartozó nyelvek vizsgálata. $\mathcal{D}$-n kívüli nyelvek összehasonlítása és vizsgálata inkább matematikai logika jellegű kérdésekhez vezet.

A matematikai kérdésekhez jó, ha megállapodunk a modellben.

Számunkra az elkülönített munkaszalag fontos lesz. Ez alapján tudunk majd beszélni „logaritmikus tárban” kiszámítható függvényről.

A több (k darab) munkaszalag léte kényelmes azok számára, akik valójában megadnak Turing-gépeket.

Standard modell

A bonyolultságelmélet „feladata” a $\mathcal{D}$ nyelvosztály finomítása, az ide tartozó nyelvek vizsgálata. $\mathcal{D}$-n kívüli nyelvek összehasonlítása és vizsgálata inkább matematikai logika jellegű kérdésekhez vezet.

A matematikai kérdésekhez jó, ha megállapodunk a modellben.

Számunkra az elkülönített munkaszalag fontos lesz. Ez alapján tudunk majd beszélni „logaritmikus tárban” kiszámítható függvényről.

A több (k darab) munkaszalag léte kényelmes azok számára, akik valójában megadnak Turing-gépeket.

Definíció

A k-szalagos Turing-gép modelljét standard modellnek hívjuk.

Standard modell

A bonyolultságelmélet „feladata” a $\mathcal{D}$ nyelvosztály finomítása, az ide tartozó nyelvek vizsgálata. $\mathcal{D}$-n kívüli nyelvek összehasonlítása és vizsgálata inkább matematikai logika jellegű kérdésekhez vezet.

A matematikai kérdésekhez jó, ha megállapodunk a modellben.

Számunkra az elkülönített munkaszalag fontos lesz. Ez alapján tudunk majd beszélni „logaritmikus tárban” kiszámítható függvényről.

A több (k darab) munkaszalag léte kényelmes azok számára, akik valójában megadnak Turing-gépeket.

Definíció

A k-szalagos Turing-gép modelljét standard modellnek hívjuk.

Ha azt mondjuk, hogy egy L nyelv eldönthető, akkor ez alatt azt értjük, hogy van alkalmas $k \in \mathbb{N}$ és alkalmas k-szalagos Turing-gép, amely eldönti L-et.

Algoritmus/Turing-gép időigénye és tárigénye

Algoritmus/Turing-gép időigénye és tárigénye

Definíció

Egy T *Turing-gép időigénye* egy ω inputon $\ell := \mathit{TIME}(\omega, T)$, ha futása $\{\kappa_i\}_{i=0}^{\ell}$, vagyis az ℓ-edik konfigurációban kerül először *STOP* állapotba, illetve ∞, ha futása végtelen (nem éri el a *STOP* állapotot).

Algoritmus/Turing-gép időigénye és tárigénye

Definíció

Egy T *Turing-gép időigénye* egy ω inputon $\ell := \mathit{TIME}(\omega, T)$, ha futása $\{\kappa_i\}_{i=0}^{\ell}$, vagyis az ℓ-edik konfigurációban kerül először *STOP* állapotba, illetve ∞, ha futása végtelen (nem éri el a *STOP* állapotot).

Természtesen döntési feladatokra specializált Turing-gépek esetén is használni fogjuk ezt a fogalmat, de a *STOP* szerepét *ELFOGAD/ELVET* veszi át.

Algoritmus/Turing-gép időigénye és tárigénye

Definíció

Egy T *Turing-gép időigénye* egy ω inputon $\ell := \mathit{TIME}(\omega, T)$, ha futása $\{\kappa_i\}_{i=0}^{\ell}$, vagyis az ℓ-edik konfigurációban kerül először *STOP* állapotba, illetve ∞, ha futása végtelen (nem éri el a *STOP* állapotot).

Természtesen döntési feladatokra specializált Turing-gépek esetén is használni fogjuk ezt a fogalmat, de a *STOP* szerepét *ELFOGAD/ELVET* veszi át.

Definíció

Egy T *Turing-gép tárigénye* egy ω inputon $s := \mathit{SPACE}(\omega, T)$, ha futása során a munkaszem/kéz alatti mező legnagyobb indexe s, illetve ∞, ha a munkaszem/kéz tetszőleges messze elmozdul a munkaszalag baloldali határától.

Észrevétel

Észrevétel

Mivel a Turing-gép legfeljebb annyi mezőt látogathat meg a munkaszalagon, amennyit mozog, ezért ezen jellemzők között fennáll a nyilvánvaló

$$SPACE(\omega, T) \leq TIME(\omega, T)$$

összefüggés.

Géposztályok

Géposztályok

A futás idő- és tárigényét nyilván az input hosszától való függése alapján ítélhetjük meg. A következő definícióval ezen függést tudjuk kifejezni.

Géposztályok

A futás idő- és tárigényét nyilván az input hosszától való függése alapján ítélhetjük meg. A következő definícióval ezen függést tudjuk kifejezni.

Definíció

Legyen $t\colon \mathbb{N} \to \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, egy T Turing-gép eleme a $\mathtt{TIME(t(n))}$ halmaznak, ha minden $\omega \in \Sigma^*$ esetén

$$TIME(\omega, T) \leq t(|\omega|).$$

Géposztályok

A futás idő- és tárigényét nyilván az input hosszától való függése alapján ítélhetjük meg. A következő definícióval ezen függést tudjuk kifejezni.

Definíció

Legyen $t\colon \mathbb{N} \to \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, egy T Turing-gép eleme a $\mathtt{TIME(t(n))}$ halmaznak, ha minden $\omega \in \Sigma^*$ esetén

$$TIME(\omega, T) \leq t(|\omega|).$$

Definíció

Legyen $s\colon \mathbb{N} \to \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, egy T Turing-gép eleme a $\mathtt{SPACE(s(n))}$ halmaznak, ha minden $\omega \in \Sigma^*$ esetén

$$SPACE(\omega, T) \leq s(|\omega|).$$

Nyelvosztályok

Nyelvosztályok

Definíció

Legyen $t : \mathbb{N} \to \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, hogy egy L nyelv eleme a $\mathcal{TIME}(t(n))$ halmaznak, ha létezik olyan T Turing-gép, amely

(i) eldönti L-et, és

(ii) $T \in \mathtt{TIME(t(n))}$, azaz $\mathit{TIME}(\omega, T) \leq t(|\omega|)$.

Nyelvosztályok

Definíció

Legyen $t\colon \mathbb{N} \to \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, hogy egy L nyelv eleme a $\mathcal{TIME}(t(n))$ halmaznak, ha létezik olyan T Turing-gép, amely

(i) eldönti L-et, és

(ii) $T \in \mathtt{TIME(t(n))}$, azaz $\mathit{TIME}(\omega, T) \leq t(|\omega|)$.

Definíció

Legyen $s\colon \mathbb{N} \to \mathbb{R}$ egy tetszőleges függvény. Azt mondjuk, hogy egy L nyelv eleme a $\mathcal{SPACE}(s(n))$ halmaznak, ha létezik olyan T Turing-gép, amely

(i) eldönti L-et, és

(ii) $T \in \mathtt{SPACE(s(n))}$, azaz $\mathit{SPACE}(\omega, T) \leq s(|\omega|)$.

$\mathcal{P}$

Az imént bevezetett $\mathcal{TIME}(t(n))/\mathcal{SPACE}(s(n))$ jelöléseknél hasznosabbak azok az osztályok ahol az idő, illetve tár korlátozást nem egy függvénnyel, hanem egy „nagyságrenddel” írjuk elő.

Definíció: Polinomiális időben eldönthető nyelvek

$$\mathcal{P} := \bigcup_{p \in \mathbb{R}[x]} \mathcal{TIME}(p(n)) = \bigcup_{a \in \mathbb{N}} \mathcal{TIME}(an^a + a).$$

$\mathcal{P}$

Az imént bevezetett $\mathcal{TIME}(t(n))/\mathcal{SPACE}(s(n))$ jelöléseknél hasznosabbak azok az osztályok ahol az idő, illetve tár korlátozást nem egy függvénnyel, hanem egy „nagyságrenddel” írjuk elő.

Definíció: Polinomiális időben eldönthető nyelvek

$$\mathcal{P} := \bigcup_{p \in \mathbb{R}[x]} \mathcal{TIME}(p(n)) = \bigcup_{a \in \mathbb{N}} \mathcal{TIME}(an^a + a).$$

A második uniós alak csupán egy speciális polinom sorozatra ritkítja ki az első uniót. A speciális polinom sorozat tudja, hogy bármely $p \in \mathbb{R}[x]$ polinomhoz létezik olyan $a \in \mathbb{N}$ természetes szám, melyre $p(n) \leq an^a + a$, ahol $n \in \mathbb{N}$. Ezért a két unió egyenlősége nyilvánvaló. A két unió csak két alternetíva a megfogalmazásra, nem része a definíciónak.

$\mathcal{EXP}, \mathcal{PSPACE}$

$\mathcal{EXP}, \mathcal{PSPACE}$

Exponenciális időben eldönthető nyelvek

$$\mathcal{EXP} := \bigcup_{a \in \mathbb{N}} \mathcal{TIME}(2^{an^a+a}).$$

$\mathcal{EXP}, \mathcal{PSPACE}$

Exponenciális időben eldönthető nyelvek

$$\mathcal{EXP} := \bigcup_{a \in \mathbb{N}} \mathcal{TIME}(2^{an^a + a}).$$

Polinomiális tárral eldönthető nyelvek

$$\mathcal{PSPACE} := \bigcup_{a \in \mathbb{N}} \mathcal{SPACE}(an^a + a).$$

$\mathcal{EXPSPACE}, \mathcal{L}$

$\mathcal{EXPSPACE}, \mathcal{L}$

Exponenciális tárral eldönthető nyelvek

$$\mathcal{EXPSPACE} := \bigcup_{a \in \mathbb{N}} \mathcal{SPACE}(2^{an^a+a}).$$

$\mathcal{EXPSPACE}, \mathcal{L}$

Exponenciális tárral eldönthető nyelvek

$$\mathcal{EXPSPACE} := \bigcup_{a \in \mathbb{N}} \mathcal{SPACE}(2^{an^a+a}).$$

Logaritmikus tárral eldönthető nyelvek

$$\mathcal{L} := \bigcup_{a \in \mathbb{N}} \mathcal{SPACE}(a \log n).$$

Tartalmazások

Tartalmazások

Könnyen belátható, hogy a nyelvosztályok között az alábbi tartalmazások teljesülnek:

Tartalmazások

Könnyen belátható, hogy a nyelvosztályok között az alábbi tartalmazások teljesülnek:

$$\begin{array}{ccccccccccc} \mathcal{L} & \subseteq & \mathcal{PSPACE} & \subseteq & \mathcal{EXPSPACE} & \subseteq & \mathcal{D} & \subseteq & \mathcal{S} & \subseteq & \mathcal{P}(\Sigma^*) \\ & & \cup| & & \cup| & & & & & & \\ & & \mathcal{P} & \subseteq & \mathcal{EXP} & & & & & & \end{array}$$

Tartalmazások

Könnyen belátható, hogy a nyelvosztályok között az alábbi tartalmazások teljesülnek:

$$\begin{array}{ccccccccccc} \mathcal{L} & \subseteq & \mathcal{PSPACE} & \subseteq & \mathcal{EXPSPACE} & \subseteq & \mathcal{D} & \subseteq & \mathcal{S} & \subseteq & \mathcal{P}(\Sigma^*) \\ & & \cup| & & \cup| & & & & & & \\ & & \mathcal{P} & \subseteq & \mathcal{EXP} & & & & & & \end{array}$$

Természetesen további kapcsolatok is vannak, ezekről a későbbiek során esik szó.

Tartalmazások

Könnyen belátható, hogy a nyelvosztályok között az alábbi tartalmazások teljesülnek:

$$\begin{array}{ccccccccccc} \mathcal{L} & \subseteq & \mathcal{PSPACE} & \subseteq & \mathcal{EXPSPACE} & \subseteq & \mathcal{D} & \subseteq & \mathcal{S} & \subseteq & \mathcal{P}(\Sigma^*) \\ & & \cup| & & \cup| & & & & & & \\ & & \mathcal{P} & \subseteq & \mathcal{EXP} & & & & & & \end{array}$$

Természetesen további kapcsolatok is vannak, ezekről a későbbiek során esik szó.

Nyelvosztályokból is jelentősen több létezik, mint amit itt megadtunk, a részletekért lásd a Qwiki oldalát: http://qwiki.stanford.edu/index.php/Complexity_Zoo.

A Lemma

A Lemma

Célunk, hogy belássuk:

$$\mathcal{L} \subseteq \mathcal{P} \subseteq \mathcal{PSPACE} \subseteq \mathcal{EXP} \subseteq \mathcal{EXPSPACE} \subseteq \mathcal{D} \subseteq \mathcal{S} \subseteq \mathcal{P}(\Sigma^*).$$

A Lemma

Célunk, hogy belássuk:

$$\mathcal{L} \subseteq \mathcal{P} \subseteq \mathcal{PSPACE} \subseteq \mathcal{EXP} \subseteq \mathcal{EXPSPACE} \subseteq \mathcal{D} \subseteq \mathcal{S} \subseteq \mathcal{P}(\Sigma^*).$$

A fenti tartalmazások jó része nyilvánvaló. A teljes tartalmazás sorozat a következő lemmából nyilvánvaló.

A Lemma

Célunk, hogy belássuk:

$$\mathcal{L} \subseteq \mathcal{P} \subseteq \mathcal{PSPACE} \subseteq \mathcal{EXP} \subseteq \mathcal{EXPSPACE} \subseteq \mathcal{D} \subseteq \mathcal{S} \subseteq \mathcal{P}(\Sigma^*).$$

A fenti tartalmazások jó része nyilvánvaló. A teljes tartalmazás sorozat a következő lemmából nyilvánvaló.

Lemma

$$\mathcal{SPACE}(s(n)) \subseteq \cup_{c \in \mathbb{N}} \mathcal{TIME}(c^{s(n)+\log(n+1)}).$$

A Lemma bizonyítása

A Lemma bizonyítása

Legyen $L \in \mathcal{SPACE}(s(n))$. Ekkor megadható olyan T Turing-gép, amely eldönti L-et (speciálisan minden $\omega \in L$-en megáll), és a tárigénye legfeljebb $s(n)$.

A Lemma bizonyítása

Legyen $L \in \mathcal{SPACE}(s(n))$. Ekkor megadható olyan T Turing-gép, amely eldönti L-et (speciálisan minden $\omega \in L$-en megáll), és a tárigénye legfeljebb $s(n)$.

Jelölés

A fenti bekezdés egy karakterisztikus bizonyítás kezdés lesz. A továbbiakban ezt $L \in_T \mathcal{SPACE}(s(n))$ jelöléssel rövidítjük.

A Lemma bizonyítása

Legyen $L \in \mathcal{SPACE}(s(n))$. Ekkor megadható olyan T Turing-gép, amely eldönti L-et (speciálisan minden $\omega \in L$-en megáll), és a tárigénye legfeljebb $s(n)$.

Jelölés

A fenti bekezdés egy karakterisztikus bizonyítás kezdés lesz. A továbbiakban ezt $L \in_T \mathcal{SPACE}(s(n))$ jelöléssel rövidítjük.

Legyen $\kappa_0(\omega) \to \kappa_1(\omega) \to \kappa_2(\omega) \to \ldots \to \kappa_\ell(\omega)$ a futás ω-n. Azaz ez egy $\ell \geq 1$ hosszú, véges konfigurációsorozat, ahol az első konfiguráció ($\kappa_0(\omega)$) a kiinduló konfiguráció (ebben az állapot START) és az utolsó állapot ($\kappa_\ell(\omega)$) az első olyan konfigutáció a futás során, amelyben az állapot ELFOGAD/ELVET.

A Lemma bizonyítása (folytatás)

A Lemma bizonyítása (folytatás)

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül.

A Lemma bizonyítása (folytatás)

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül.

Hányféle konfiguráció léphet fel a fenti sorozatban rögzített ω esetén? Legyen $|\omega| = n$.

A Lemma bizonyítása (folytatás)

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül.

Hányféle konfiguráció léphet fel a fenti sorozatban rögzített ω esetén? Legyen $|\omega| = n$.

Egy felső becslés a kérdésre adandó válaszra (α_T, β_T konstansok T-től függnek):

$$(n+2)\cdot(s(n)+1)^k \cdot |\Gamma|^{k\cdot s(n))} \cdot |S| \leq \alpha_T(n+1)\alpha_T^{s(n)} \leq \beta_T^{s(n)+\log(n+1)},$$

A Lemma bizonyítása (folytatás)

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül.

Hányféle konfiguráció léphet fel a fenti sorozatban rögzített ω esetén? Legyen $|\omega| = n$.

Egy felső becslés a kérdésre adandó válaszra (α_T, β_T konstansok T-től függnek):

$$(n+2)\cdot(s(n)+1)^k \cdot |\Gamma|^{k\cdot s(n))} \cdot |S| \leq \alpha_T(n+1)\alpha_T^{s(n)} \leq \beta_T^{s(n)+\log(n+1)},$$

hiszen az input szem helyzete $n+2$-féle,

A Lemma bizonyítása (folytatás)

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül.

Hányféle konfiguráció léphet fel a fenti sorozatban rögzített ω esetén? Legyen $|\omega| = n$.

Egy felső becslés a kérdésre adandó válaszra (α_T, β_T konstansok T-től függnek):

$$(n+2)\cdot(s(n)+1)^k \cdot |\Gamma|^{k\cdot s(n))} \cdot |S| \leq \alpha_T(n+1)\alpha_T^{s(n)} \leq \beta_T^{s(n)+\log(n+1)},$$

hiszen az input szem helyzete $n+2$-féle, mind a k munkaszalag felett: a munka szem helyzete $s(n)+1$-féle,

A Lemma bizonyítása (folytatás)

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül.

Hányféle konfiguráció léphet fel a fenti sorozatban rögzített ω esetén? Legyen $|\omega| = n$.

Egy felső becslés a kérdésre adandó válaszra (α_T, β_T konstansok T-től függnek):

$$(n+2)\cdot(s(n)+1)^k \cdot |\Gamma|^{k\cdot s(n))} \cdot |S| \leq \alpha_T(n+1)\alpha_T^{s(n)} \leq \beta_T^{s(n)+\log(n+1)},$$

hiszen az input szem helyzete $n+2$-féle, mind a k munkaszalag felett: a munka szem helyzete $s(n)+1$-féle, a munkaszalag tartalma $(|\Gamma|+1)^{s(n)}$-féle (az $s(n)$-nél nagyobb indexű munka-mezők szükségszerűen az érintetlen karaktert tartalmazzák),

A Lemma bizonyítása (folytatás)

Könnyű látni, hogy a futás során nem ismétlődhet konfiguráció, azaz $i \neq j$ esetén $\kappa_i \neq \kappa_j$ teljesül.

Hányféle konfiguráció léphet fel a fenti sorozatban rögzített ω esetén? Legyen $|\omega| = n$.

Egy felső becslés a kérdésre adandó válaszra (α_T, β_T konstansok T-től függnek):

$$(n+2)\cdot(s(n)+1)^k \cdot |\Gamma|^{k\cdot s(n))} \cdot |S| \leq \alpha_T(n+1)\alpha_T^{s(n)} \leq \beta_T^{s(n)+\log(n+1)},$$

hiszen az input szem helyzete $n+2$-féle, mind a k munkaszalag felett: a munka szem helyzete $s(n)+1$-féle, a munkaszalag tartalma $(|\Gamma|+1)^{s(n)}$-féle (az $s(n)$-nél nagyobb indexű munka-mezők szükségszerűen az érintetlen karaktert tartalmazzák), továbbá az állapot $|S|$-féle lehet.

A Lemma bizonyítása (folytatás)

A Lemma bizonyítása (folytatás)

Összefoglalva: Ha a futási idő $\beta_T{}^{s(n)+\log n}$-nél hosszabb lenne, akkor a futás során a konfigurációk ismétlődnének, így a futás végtelen lenne.

A Lemma bizonyítása (folytatás)

Összefoglalva: Ha a futási idő $\beta_T{}^{s(n)+\log n}$-nél hosszabb lenne, akkor a futás során a konfigurációk ismétlődnének, így a futás végtelen lenne.

Tudjuk, hogy nincs így. Tehát kaptuk, hogy T időigénye automatikusan megfelel az észrevételben szereplőkkel. Azaz a kezdeti T igazolja az állítást.

Robosztusság

Robosztusság

A kiszámíthatóságnál/eldönthetőségnél már szó volt arról, hogy ezek a fogalamak a Turing-gép definícióján alapulnak.

Robosztusság

A kiszámíthatóságnál/eldönthetőségnél már szó volt arról, hogy ezek a fogalamak a Turing-gép definícióján alapulnak. A Turing-gép matematikai definíció sok apróságot tartalmaznak, amik bizonyos szemszögből hasznosak.

Robosztusság

A kiszámíthatóságnál/eldönthetőségnél már szó volt arról, hogy ezek a fogalamak a Turing-gép definícióján alapulnak. A Turing-gép matematikai definíció sok apróságot tartalmaznak, amik bizonyos szemszögből hasznosak. Az apróságokban azonban nem remélhető egységesítsés. Sokan sokféleképpen írhatják le a számukra legszimpatikusabb változatát a Turing-gép fogalmának.

Robosztusság

A kiszámíthatóságnál/eldönthetőségnél már szó volt arról, hogy ezek a fogalamak a Turing-gép definícióján alapulnak. A Turing-gép matematikai definíció sok apróságot tartalmaznak, amik bizonyos szemszögből hasznosak. Az apróságokban azonban nem remélhető egységesítsés. Sokan sokféleképpen írhatják le a számukra legszimpatikusabb változatát a Turing-gép fogalmának.

Ennek ellenére nem hatnak ki a kiszámíthatóság/eldönthetőség fogalmára.

Robosztusság

A kiszámíthatóságnál/eldönthetőségnél már szó volt arról, hogy ezek a fogalamak a Turing-gép definícióján alapulnak. A Turing-gép matematikai definíció sok apróságot tartalmaznak, amik bizonyos szemszögből hasznosak. Az apróságokban azonban nem remélhető egységesítsés. Sokan sokféleképpen írhatják le a számukra legszimpatikusabb változatát a Turing-gép fogalmának.

Ennek ellenére nem hatnak ki a kiszámíthatóság/eldönthetőség fogalmára.

Hasonló megjegyzés igaz a fent bevezetett nagyságrendi korlátokkal definiált nyelvosztályokra. $\mathcal{P}$ a polinom időben eldönthető nyelvek osztálya ugyanaz marad, ha más modellel dolgozunk.

Robosztusság

A kiszámíthatóságnál/eldönthetőségnél már szó volt arról, hogy ezek a fogalamak a Turing-gép definícióján alapulnak. A Turing-gép matematikai definíció sok apróságot tartalmaznak, amik bizonyos szemszögből hasznosak. Az apróságokban azonban nem remélhető egységesítsés. Sokan sokféleképpen írhatják le a számukra legszimpatikusabb változatát a Turing-gép fogalmának.

Ennek ellenére nem hatnak ki a kiszámíthatóság/eldönthetőség fogalmára.

Hasonló megjegyzés igaz a fent bevezetett nagyságrendi korlátokkal definiált nyelvosztályokra. $\mathcal{P}$ a polinom időben eldönthető nyelvek osztálya ugyanaz marad, ha más modellel dolgozunk.

$\mathcal{P}$ egy robusztus nyelvosztály.

Veszélyek

Veszélyek

Jóval nagyobb problémát okozna, ha definiálnánk a $\mathcal{LINEAR} := \{L(T) : T \in \cup_{a \in \mathbb{N}} \texttt{Time}(an + a)\}$ nyelvosztályt.

Veszélyek

Jóval nagyobb problémát okozna, ha definiálnánk a $\mathcal{LINEAR} := \{L(T) : T \in \cup_{a \in \mathbb{N}} \texttt{Time}(an + a)\}$ nyelvosztályt.

Ez a nyelvosztály már nem annyira robusztus.

Veszélyek

Jóval nagyobb problémát okozna, ha definiálnánk a $\mathcal{LINEAR} := \{L(T) : T \in \cup_{a \in \mathbb{N}} \texttt{Time}(an + a)\}$ nyelvosztályt.

Ez a nyelvosztály már nem annyira robusztus.

Ha ilyennel talalkoznánk, akkor nagyon részletesen értsük meg mely eldönthetőség definiációra alapul az időkorlát.

Veszélyek

Jóval nagyobb problémát okozna, ha definiálnánk a $\mathcal{LINEAR} := \{L(T) : T \in \cup_{a\in\mathbb{N}}\texttt{Time}(an + a)\}$ nyelvosztályt.

Ez a nyelvosztály már nem annyira robusztus.

Ha ilyennel talalkoznánk, akkor nagyon részletesen értsük meg mely eldönthetőség definiációra alapul az időkorlát.

Hasonlóan komoly technikai problémakat vetne fel az $\mathcal{E} := \{L(T) : T \in \cup_{a\in\mathbb{N}}\texttt{Time}(a^{n+a})\}$ nyelvosztály.

Veszélyek

Jóval nagyobb problémát okozna, ha definiálnánk a $\mathcal{LINEAR} := \{L(T) : T \in \cup_{a\in\mathbb{N}}\texttt{Time}(an + a)\}$ nyelvosztályt.

Ez a nyelvosztály már nem annyira robusztus.

Ha ilyennel talalkoznánk, akkor nagyon részletesen értsük meg mely eldönthetőség definiációra alapul az időkorlát.

Hasonlóan komoly technikai problémakat vetne fel az $\mathcal{E} := \{L(T) : T \in \cup_{a\in\mathbb{N}}\texttt{Time}(a^{n+a})\}$ nyelvosztály.

Korábbi definícióink nem voltak véletlenek. A fenti két példával csak azok dolgozzanak, akik a bonyolultságelmélet alapjait már mélyen megértették és szembesültek a technikai nehézségekkel.

Vége van!

Köszönöm a figyelmet!