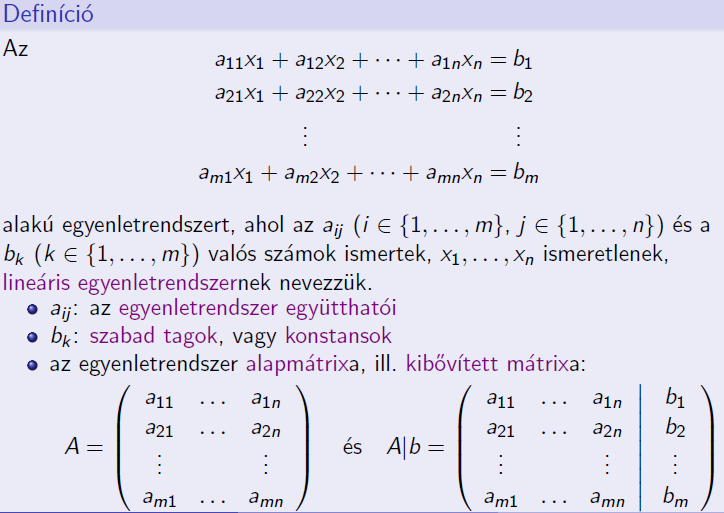
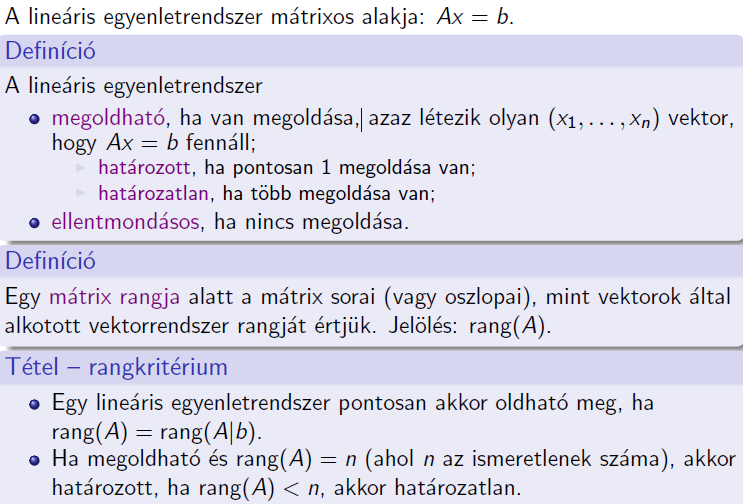
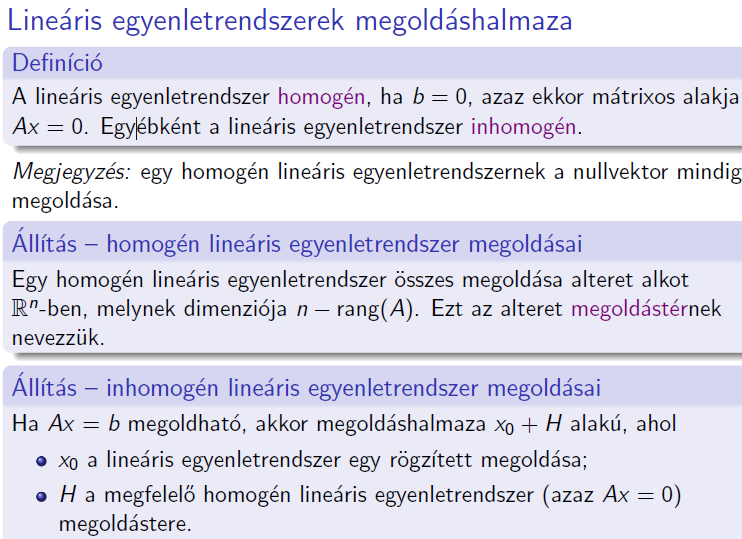
**7. tétel**

**Lineáris egyenletrendszer**



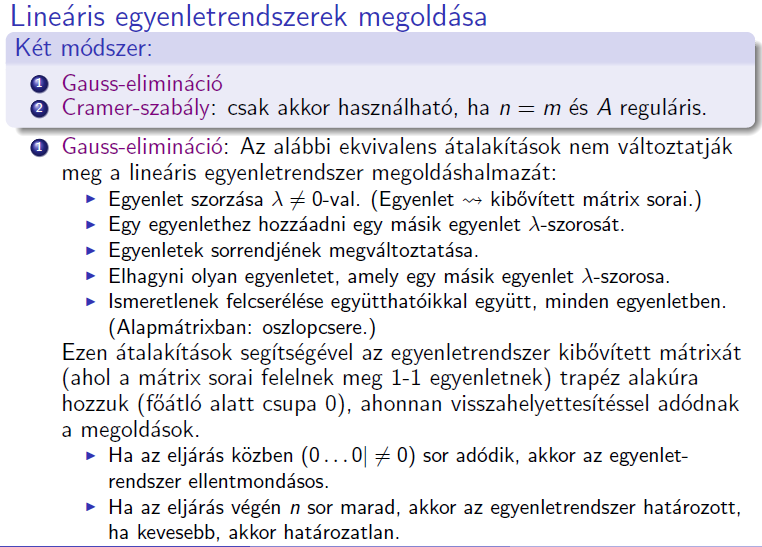


A lineáris egyenlet mátrixos alakja , egy mátrix rangja, megoldható, határozott, határozatlan, ellentmondásos, Tétel - rangkritérium



Lineáris egyenletrendszerek megoldáshalmaza, homogén, inhomogén, homogén lineáris egyenletrendszerek megoldásai, inhomogén egyenletrendszerek megoldásai

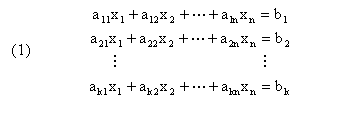
Gauss elimináció



Lineáris egyenletrendszerek megoldása, Gauss-eliminánció

A **lineáris egyenletrendszer** olyan többismeretlenes [egyenletrendszer](https://hu.wikipedia.org/wiki/Egyenletrendszer), ahol minden ismeretlen elsőfokon (azaz első [hatványon](https://hu.wikipedia.org/wiki/Hatv%C3%A1ny)) szerepel.

**Definíció.** Az



egyenletek halmazát, ahol xj valós számok (j = 1,...,n) az ismeretleneket, aij adott valós számok (i = 1,2,...,k, j = 1,2,...,n) az ismeretlenek együtthatóit jelentik és a bi -k adott valós számok, lineáris egyenletrendszernek nevezzük.

Megjegyzés.

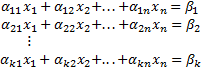
(1) Amennyiben b1 = b2 =...= bkk= 0, az egyenletrendszert homogénnak, ellenkező esetben inhomogénnak mondjuk.

(2) A c1,c2,...,cn számok az egyenletrendszer megoldását adják, ha az xi ismeretlenek helyére helyettesítve minden egyenletet kielégítenek.

Az általános lineáris egyenletrendszerek megoldására az egyik legtermészetesebben adódó, egyszerű és gyakorlati szempontból is jól alkalmazható eljárás a Gauss-féle kiküszöbölés, amelynek számos fontos elméleti következménye is van. Speciális egyenletrendszerekre vonatkozik a Cramer-szabály, amely a determinánsok segítségével ad képletet a megoldásra. A jelen fejezetben vezetjük be a lineáris függetlenséget és a mátrix rangját is, amelyek a későbbiekben is alapvető szerepet játszanak. Mindezek egyik alkalmazásaként visszatérünk a négyzetes mátrixok körében az invertálhatóság és a nullosztók kérdésére.

Gauss - kiküszöbölés

Egy *k* egyenletből álló *n* ismeretlenes *lineáris egyenletrendszer* általános alakja



ahol az αij együtthatók és a βi konstansok egy *T* kommutatív test elemei. Az egyenletek száma (*k*) és az ismeretlenek száma (*n*) egymástól függetlenül is tetszőleges lehet (tehát pl. semmiképpen sem szorítkozunk csak a *k*=*n* esetre).

Az egyenletrendszer egy *megoldás*án *T*-beli elemek egy olyan γ1,…,γn sorozatát értjük, amelyeket a megfelelő *x*i-k helyére beírva, valamennyi egyenletben egyenlőség teljesül.

Van olyan egyenletrendszer, amelynek nincs megoldása, van, amelyik egyértelműen oldható meg (azaz pontosan egy megoldása van) és van olyan, amelyet (egynél) több megoldás is kielégít. (Ez utóbbi esetben elég óvatosan fogalmaztunk, annak ellenére, hogy például valós számokra vonatkozó egyenletrendszereknél megszoktuk, hogy egynél több megoldás esetén a megoldásszám végtelen. Látni fogjuk, hogy végtelen test esetén ez valóban mindig így van. Azonban véges, mondjuk *t* elemű test esetén az összes szóba jövő *x*1,…,*x*n-re is csak *t*n lehetőségünk van, tehát eleve nem lehet végtelen sok megoldás.)

Az alábbi kérdésekre keressük a választ: (a) mi a feltétele annak, hogy egy egyenletrendszer megoldható legyen; (b) (megoldhatóság esetén) hány megoldás van; (c) hogyan lehet az összes megoldást áttekinteni; (d) milyen módszerrel juthatunk el (egy vagy az összes) megoldáshoz.

Ebben a pontban a fenti kérdésekre a *Gauss-féle kiküszöbölés* (röviden *Gauss-kiküszöbölés* vagy latinosan *Gauss-elimináció*) segítségével adjuk meg a választ.

Az eljárás során az alábbi lépéseket fogjuk végezni, amelyek valamennyien az eredetivel ekvivalens egyenletrendszerekhez vezetnek (azaz olyanokhoz,

amelyeknek pontosan ugyanazok a megoldásai, mint az eredetinek):

E1. Valamelyik egyenletet egy nullától különböző *T*-beli elemmel (a továbiakban: *skalár*ral) végigszorozzuk.

E2. Valamelyik egyenlethez egy másik egyenlet skalárszorosát hozzáadjuk.

E3. Két egyenletet felcserélünk.

E4. Az olyan egyenleteket, ahol valamennyi együttható és minden jobb oldali konstans is 0, elhagyjuk.

Ezeket a lépéseket *elemi ekvivalens átalakítások*nak nevezzük.

Az elemi ekvivalens átalakítások segítségével az egyenletrendszerből az alább részletezett módon egymás után ki fogjuk *küszöbölni* az ismeretleneket.

Tegyük fel, hogy α11≠0. Az első egyenletet osszuk végig α11-gyel (azaz alkalmazzuk E1-et az α11 reciprokával), majd minden *i*>1-re az *i*-edik egyenletből vonjuk ki az első egyenlet αi1-szeresét. Ezzel a többi egyenletből *kiküszöböl*tük *x*1-et.

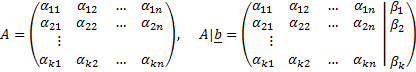
Tegyük fel, hogy az így kapott egyenletrendszerben az új α22≠0. Ekkor az előző eljárást megismételhetjük: a második egyenletet végigosztjuk α22-vel, majd minden *i*>2-re az *i*-edik egyenletből kivonjuk a második egyenlet αi2-szeresét stb.

Ha valamikor megakadtunk, pl. az előbb α22=0 volt, de mondjuk α52≠0, akkor a második és az ötödik egyenletet felcseréljük, és így haladunk tovább.

Ha ez sem megy, azaz minden *i*≥2 esetén αi2=0, akkor a harmadik ismeretlenre térünk át, vagyis α23-at vizsgáljuk stb.

Nemsokára néhány konkrét példán keresztül illusztráljuk, hogyan fest mindez a gyakorlatban és hogyan juthatunk el így az egyenletrendszer megoldásához. Előtte azonban érdemes némi technikai egyszerűsítést bevezetni.

Vegyük észre, hogy a fenti lépések nyomon követéséhez elég csak az együtthatók és a jobb oldali konstansok változását figyelni, az *x*i,+ és = „jeleket” fölösleges mindig újra leírni. Ezért az egyenletrendszert egyszerűbben jellemezhetjük mátrixok segítségével: az αij együtthatókból képezett *k*×*n*-es *A* mátrixot az egyenletrendszer *együtthatómátrix*ának nevezzük, a jobb oldali konstansokkal kibővített *k*×(*n*+1)-es mátrixot pedig az egyenletrendszer *kibővített mátrix*ának nevezzük és A|b-vel jelöljük, azaz



A kibővített mátrixban az együtthatók alkotta rész és a jobb oldali konstansok közé iktatott függőleges vonallal jelezzük, hogy a kétféle típusú elemek eltérő szerepet játszanak az egyenletrendszerben. (Az A|b felírásnál b a jobb oldalon álló βi konstansokból képezett „vektort” jelöli, erről bővebben ennek a pontnak a végén lesz szó.)

Azonnal adódik, hogy az egyenletekkel végzett E1–E4 elemi ekvivalens átalakításoknak a kibővített mátrixnál a sorokkal végzett hasonló változtatások felelnek meg:

M1. Valamelyik sort egy nullától különböző skalárral végigszorozzuk.

M2. Valamelyik sorhoz egy másik sor skalárszorosát hozzáadjuk.

M3. Két sort felcserélünk.

M4. A csupa 0-ból álló sorokat elhagyjuk.

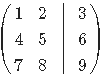
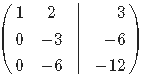
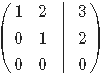
A kibővített mátrixon végzett fenti lépéseket *elemi sorekvivalens átalakítások*nak nevezzük.

(Az ekvivalens lépések „visszacsinálhatósága” érdekében formailag teljesebb, ha E4-nél, illetve M4-nél az ilyen egyenletek, illetve sorok hozzávételét is megengedjük, de ennek gyakorlati alkalmazására nyilván sosincs szükség.)

Most három, valós számokra vonatkozó egyenletrendszeren mutatjuk be a kiküszöbölési eljárást.

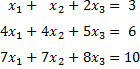
**P1 példa**:



Ennek kibővített mátrixa  A jelzett kiküszöbölési eljárásnak megfelelően vonjuk ki a második sorból az első sor 4-szeresét, a harmadik sorból pedig az első sor 7-szeresét. Így az  mátrixhoz jutunk. Osszuk el a második sort –3-mal, majd adjuk hozzá ennek 6-szorosát a harmadik sorhoz. Az így kapott mátrix  Itt a csupa nulla sor elhagyható, tehát marad az  mátrix. Itt a második sorból azonnal leolvasható, hogy *x*2=2 (hiszen *x*2 „ki van fejezve”). Ezt visszahelyettesíthetjük az első sornak megfelelő egyenletbe: *x*1+2*x*2=*x*1+2·2=3, tehát *x*1=–1. Azonban ez a lépés is „automatizálható”. Ha a legutolsó mátrixnál az első sor második elemét kiejtjük, akkor *x*1 is „ki lesz fejezve”. Vonjuk ki ezért az első sorból a második sor 2-szeresét, ekkor az  mátrixot kapjuk, ahonnan valóban *x*1=–1 is közvetlenül leolvasható.

Az egyenletrendszernek tehát egyetlen megoldása van: *x*1=–1, *x*2=2.

**P2 példa**:



A kiküszöbölés során a kibővített mátrix a következőképpen változik:



Mivel az utolsó mátrix harmadik sora a lehetetlen 0*x*1+0*x*2+0*x*3=1 egyenletnek felel meg, ezért ennek az egyenletrendszernek nincs megoldása.

### **3.1.1 Tétel**

I. Egy lineáris egyenletrendszer kibővített mátrixa elemi sorekvivalens átalakításokkal redukált lépcsős alakra hozható.

II. Az egyenletrendszer akkor és csak akkor oldható meg, ha a (redukált) lépcsős alakban nincs tilos sor.

III. Az egyenletrendszernek akkor és csak akkor egyértelmű a megoldása, ha (nincs tilos sor és) a vezéregyesek száma megegyezik az ismeretlenek számával.

IV. Ha több megoldás van, akkor a vezéregyest nem tartalmazó oszlopoknak megfelelő ismeretlenek szabad paraméterek (tetszőlegesen megválaszthatók), a többi ismeretlen pedig ezekkel egyértelműen kifejezhető. A megoldásszám ekkor végtelen test esetén végtelen, *t* elemű test esetén pedig *t*s, ahol *s* a szabad paraméterek száma, és a(z összes) megoldás közvetlenül leolvasható a redukált lépcsős alakból.

### **3.1.2 Tétel**

Ha egy *k* egyenletből álló *n* ismeretlenes lineáris egyenletrendszernek egyetlen megoldása van, akkor *n*≤*k*.❶

*Bizonyítás*: Egyértelmű megoldás esetén az RLA-ban a vezéregyesek száma *n*, másrészt a vezéregyesek különböző sorokban helyezkednek el, tehát számuk legfeljebb *k*. Innen valóban *n*≤*k*.❷

Ennek az észrevételnek egy egyszerű, de fontos következményét a későbbiekben sokszor fel fogjuk használni. Ehhez előbb bevezetjük a homogén lineáris egyenletrendszer fogalmát:

### **3.1.3 Definíció**

Egy lineáris egyenletrendszert *homogén*nek nevezünk, ha a jobb oldali konstansok mindegyike nulla.❶

Egy homogén egyenletrendszer biztosan megoldható, hiszen *x*1=…=*x*n=0 mindig megoldás. Ezt *triviális megoldás*nak nevezzük. Így itt az az érdekes kérdés, hogy mikor létezik nemtriviális megoldás. Erre elégséges feltételt ad az alábbi

### **3.1.4 Tétel**

Ha egy homogén lineáris egyenletrendszerben az ismeretlenek száma nagyobb, mint az egyenletek száma, akkor az egyenletrendszernek biztosan létezik nemtriviális megoldása.❶

*Bizonyítás*: Indirekt, tegyük fel, hogy a triviálison kívül nincs más megoldás. Ekkor az egyenletrendszernek egyetlen megoldása van, tehát a 3.1.2 Tétel szerint az ismeretlenek száma nem lehet nagyobb az egyenletek számánál, ami ellentmond a feltételnek. ❷

A továbbiak előkészületeként az egyenletrendszerek két másik felírási módjával ismerkedünk meg. Ehhez szükségünk lesz az (oszlop)vektorok fogalmára.

### **3.1.5 Definíció**

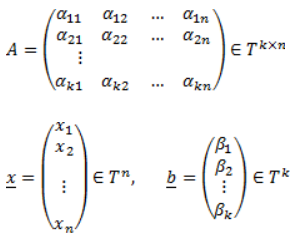
Az egy oszlopból álló mátrixokat *oszlopvektor*oknak nevezzük. Egy ilyen mátrix (egyetlen oszlopának) elemeit a vektor *komponensei*nek vagy *koordinátái*nak hívjuk. A *T* test elemeiből képzett *q* komponensű vektorok összességét (*T*q×1 helyett röviden) *T*q-val jelöljük.❶

Ez a fogalom a sík-, illetve térvektorok (valós) számpárokként, illetve számhármasokként történő felírási módjának az általánosítása. Később még sokkal általánosabb értelemben fogjuk használni a „vektor” szót (lásd a 4.1 pontot).

*T* q-ban — az általános mátrixműveleteknek megfelelően — beszélhetünk két vektor összegéről, illetve egy vektor skalárszorosáról (azaz *T*-beli elemmel vett szorzatáról), ezeket úgy kapjuk, hogy a megfelelő komponenseket összeadjuk, illetve a komponenseket a skalárral végigszorozzuk.

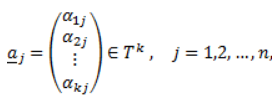
A vektorokat aláhúzott latin kisbetűkkel fogjuk jelölni.

Most rátérünk az egyenletrendszer egyik átírási módjára. Legyen



azaz *A* az együtthatómátrix, x−−x\_ az ismeretlenekből képezett vektor és a (már említett) b–b\_ a jobb oldali konstansokból álló vektor. Ekkor a mátrixszorzás definíciójának megfelelően az egyenletrendszer felírható Ax=b alakban.

A másik átírási módhoz legyen



tehát az aj\_-k az *A* együtthatómátrix oszlopvektorai. Ekkor az egyenletrendszer a következőképpen írható fel: x1a1+x2a2+...+xnan=b

**Tétel.** A mondatszerkezetű nyelvek osztálya egybeesik a Turing gépek által elfogadott nyelvek osztályával.

Amennyiben a mozgásfüggvény képhalmaza a *Q*×*V*×{*Bal*, *Jobb*, *Helyben*} halmaz (és nem annak részhalmazainak halmaza), akkor determinisztikus Turing gépről beszélünk, és érvényes a következő tétel.

**Tétel.**

A determinisztikus Turing gépek által elfogadott nyelvek osztálya egybeesik a nemdeterminisztikus Turing gépek által elfogadott nyelvek osztályával.

A Turing-gépeknek több változata is ismert, most ezek közül mutatunk be néhányat.

Van olyan definíció, ahol a fej a {*Bal*, *Jobb*} irányokba léphet, és nem maradhat helyben. Belátható, hogy egy ilyen Turing-gép szimulálni tudja az eddigiekben ismertetett változatnak a fejet helyben

hagyó lépéseit is: pl. egyet balra lép a fej, és egy olyan állapotba kerül az automata, amiben bármit is olvas a szalagon, azt nem változtatja meg, viszont jobbra visszalép és az eredeti automata állapotának megfelelelő állapotba kerül.

Ugyancsak szokásos a csak *egyirányban végtelen szalagú Turing-gép* használata, amelyik szintén képes az általános változat szimulációjára. Ekkor a szalag első karaktere egy speciális jel, amiből a gép rájön, hogy erre nem mehet tovább a fej. Ekkor egy olyan speciális állapotba kerül, aminek hatására jobbra lép, először leírja azt a jelet, ami eredetileg a speciális szimbólum helyére írt volna (ha a szalag mindkétirányban végtelen lenne), majd az itt olvasott karaktert eggyel jobbra, és így tovább, vagyis a szalag teljes (értelmes) tartalmát eggyel jobbra másolja, ezután (észlelve a felhasznált tárterület jobb szélét), a fej vissza mozog a baloldalra, aholis a gép folytatja az eredetileg tervezett számítását.

Sokszor az egyszerűbb leírás kedvéért többszalagos Turing-gépet használunk:

*TMk*=(*k*, *Q*, *T*, *V*, *q*0, #, *d*, *F*) *k*- szalagos Turing-gép, ahol *k*∈ ℕ természetes szám, ennyi szalagja van aTuring-gépnek; és a *d* átmenetfüggvény alakja a következő: *d*:*Q*×*Vk* → *Q*×*Vk*×{*Bal*, *Jobb*, *Helyben*}*k*.

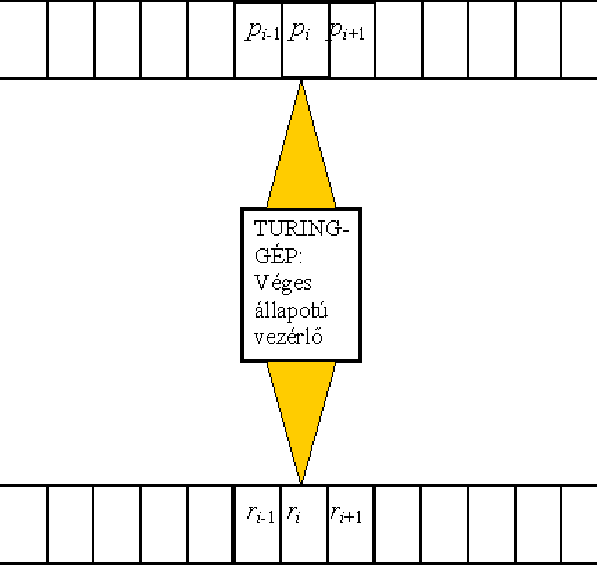
Működését tekintve a többszalagos Turing-gép egy lépésben olvashat/írhat egyszerre több szalagra is. Kezdő konfigurációban az egyik szalagon (input-szalag) van a feldolgozandó adat, a többi szalag pedig üres. Többszalagos gépek esetén szokás egy szalagot az outputnak is fenntartani, ekkor a számítás végén azon a szalagon olvasható az eredmény, illetve sokszor az input szalag csak olvasható.

Minden többszalagos Turing-gép működése szimulálható egyszalagos Turing-géppel, vagyis egyszalagos Turing-gép is el tudja végezni azt a számítást amit egy többszalagos Turing-gép.

Megkülönböztethetünk kiszámító és eldöntő Turing-gépeket a következő definíció alapján.

Amennyiben a Turing-gép célja adott függvény kiszámítása a megadott bemenő értékekkel, akkor a gép a megállásakor az (output)szalagon a megfelelő eredményt hagyja. Ezzel szemben vannak olyan számítások, amikor a választ egy igen-nem kérdésre keressük, ezesetben eldöntő Turing-gépről beszélünk. Az eldöntő Turing-gépekkel lehet pl. egy *L* nyelvet elfogadtatni a következőképpen: bemenet egy *w*∈*T*\* szó, a Turing-gép számításának eredménye pontosan akkor "igen" ha *w*∈*L* teljesül. (Ugyanez a hatás érhető el, ha csak akkor engedjük végállapotba jutni a gépet, ha elfogad.)

A következő ábrán egy kétszalagos Turing-gép vázlata látható.



A továbbiakban, ha mást nem mondunk, Turing-gép alatt determinisztikus Turing-gépet értünk. A példákban a fej mozgását jelentő szavakat azok kezdőbetűivel rövidítjük.

## Lineárisan korlátozott automata

Az automata egy véges vezérlővel rendelkezik és egy szalaggal, amelyen kezdetben az input szó áll. Az automata a működése során két lényeges eltérést mutat az eddig tárgyalt automatákhoz képest: az egyik, hogy a szalagon a fej előre és hátra is mozoghat, a másik, még lényegesebb, hogy nem csak olvashatja a szalagot, de annak tartalmát át is írhatja, ennek megfelelően nem olvasó, hanem író-olvasó fejről beszélünk.

**Definíció.** Az *LBA*=(*Q*, *T*, *V*, *q*0, #, *d*, *F*)- t (nemdetrminisztikus) lineárisan korlátozott automatának hívjuk, ha *Q* az állapotok véges halmaza, *T* az inputábécé, *V*⊇*T* a szalagábécé, *q*0 a kezdőállapot, #∈*V*\*T* speciális jel a szalag azon részén, ahol nincs input (tulajdonképpen a szóköz jelnek feleltethető meg), *d* az átmenetfüggvény, *F*⊆*Q* pedig a végállapotok halmaza. A *d* átmenetfüggvény (leképezés) alakja a következő: a (*Q*×(*V* ∖{#})) halmazból képez a (*Q*×*V*×{*Bal*, *Jobb*, *Helyben*}) részhalmazaiba, illetve (*Q*×{#}) halmazból képez a (*Q*×#×{*Bal*, *Jobb*, *Helyben*}) részhalmazaiba. ★

Az automata egy konfigurációját (*u*, *q*, *av*) –ként írhatjuk le, ahol *u*, *v*∈*V*\* a szalagon levő információ a fejtől balra, illetve jobbra, *q*∈*Q* az aktuális állapot, *a*∈*V* pedig a fej által éppen látott szimbólum a szalagon. Kezdetben az input # jelek között szerepel az input szalagon és a fej az input első betűjére van pozícionálva, vagyis a kezdeti konfiguráció (*λ*, *q*0, *aw* ') ahol az input *w*=*aw* '. (Üresszó input esetén (*λ*, *q*0, #) a konfiguráció kezdetben.)

Az automata vezérlője (a *d* leképezés által leírt módon) az aktuális állapot és az éppen olvasott szimbólum alapján nemdeterminisztikusan választ a lehetséges átmenetek közül: (*q* ', *b*, *m*)∈*d*(*q*, *a*)

jelentése: ha az automata *q* állapotban van és *a* szimbólumot lát a szalagon, akkor átmehet *q* ' állapotba, miközben a szalagon *a* helyére *b*- t ír és a fej az *m*∈{*Bal*, *Jobb*, *Helyben*} által megadott irányba lép egyet a szalagon (vagy értelemszerűen *Helyben* esetén helyben marad). Akkor mondjuk, hogy az automata elfogadott egy input szót, ha van az átmeneteknek olyan véges sorozata a szó feldolgozása során, hogy az automata végállapotba jut. Az elfogadott szavak halmaza adja az elfogadott (vagy felismert) nyelvet.

A definícióból látható, hogy a szalagon levő # jelek nem írhatóak felül, ennek megfelelően az automata csak az eredeti input által elfoglalt területet használhatja számolásra.

**Tétel.** A lineárisan korlátozott automatákkal elfogadott nyelvek osztálya megegyezik a környezetfüggő nyelvek osztályával.

A bizonyítás ötletét röviden mutatjuk be: Ha egy nyelv környezetfüggő, akkor monoton nyelvtannal generálható, vagyis a mondatforma hossza a levezetés egyik lépésében sem haladja meg a levezett szó hosszát. Az adott nyelvtan alapján megszerkeszthető egy olyan lineárisan korlátozott automata, amely éppen a szabályok alkalmazását szimulálja visszafelé. Ha adott egy lineárisan korlátozott automata, akkor készíthető egy olyan analitikus nyelvtan, amely pont az automatát szimulálja, a nemterminális éppen a fej helyét jelöli, és tárolja az aktuális állapotot. Ez alapján a duális, generatív nyelvtan is elkészíthető, ami monoton. ∎

Itt jegyezzük meg, hogy a nemdeterminisztikus lineárisan korlátozott automata az amely a környezetfüggő nyelvosztály elfogadására alkalmas, az pedig, hogy a determinisztikus lineárisan korlátozott automaták által felismert nyelvek osztálya valódi részhalmaza-e ennek egy e jegyzet megírása idején is fennálló nevezetes megoldatlan probléma.

Igazolható, hogy az ismertetett modellel ekvivalensek, vagyis ugyanezt a nyelvosztályt fogadják el azok a lineárisan korlátozott automaták, ahol az automatának egy előre rögzített *k* konstansszor annyi szalagpozíció (tárhely) áll rendelkezésre működése során, mint az input hossza (lásd [Tárhely](#_bookmark152) [tétel](#_bookmark152) [[223]](#_bookmark152)). Továbbá az is ismert (Savitch tétele (ejtsd: szévics)), hogy determinisztikus automatával, négyzetesen korlátolt tárral (vagyis, ahol az inputszó hosszának négyzetével arányos a megengedetten felhasználható szalagterület) minden környezetfüggő nyelv felismerhető.

**Eldönthetetlen probléma**

Mennyi az eldöntési problémákat megoldó algoritmusok száma? Ha elfogadjuk azt, hogy van olyan eszköz melyben végesen reprezentálható minden algoritmus és a reprezentáció ráadásul kódolható egy rögzített ábécé feletti szavakkal, akkor azt kapjuk, hogy az összes algoritmus halmaza megszámlálhatóan végtelen kell, hogy legyen. Ez azt jelenti, hogy nagyságrendileg több probléma van, mint amennyi algoritmus. Következik, hogy biztosan van olyan probléma, amihez nem párosítható őt megoldó algoritmus. Azaz, kell, hogy legyen eldönthetetlen probléma.

**IDŐ ÉS TÁR KORLÁTOK NEM TALÁLTAM**

**Egyszalagos, determinisztikus Turing-gép**

**Először a legegyszerűbb változatát definiáljuk a Turing-gépnek, az egyszalagos Turing-gépet. Ekkor a gépnek egyetlen (egyik irányban végtelen) szalagja van, egy író-olvasó fejjel, mely mindkét irányba tud mozogni a szalagon. (Később tárgyalunk még további változatokat is.)**

**9.1. Definíció Egy Turing-gépet a következő T=(Q,Σ,Γ,q0,\_,F,δ) hetes ír le, ahol:**

Q egy véges, nem üres halmaz, ez a gép állapotainak halmaza

Σ egy véges, nem üres halmaz, ez a bemeneti ábécé

Γ egy véges, nem üres halmaz, ez a szalagábécé

q0∈Q a kezdőállapot

\_∈Γ∖Σ, a szalagon az üres jel

F⊆Q az elfogadó állapotok halmaza

δ az átmeneti függvény, δ:(q,a)→(q',b,D), ahol q,q'∈Q, a,b∈Γ és D∈{B,J,H} (azaz Balra, Jobbra vagy Helyben).

**A gép kezdetben a q0 állapotban van, a szalag elején a bemeneti szó található (ebben csak Σ-beli karakterek szerepelhetnek), a szalag többi része \_ szimbólumokkal van feltöltve és a fej a szalag első (bal szélső) mezőjén áll. Ez a gép kezdőhelyzete.**

A gép minden lépésben beolvassa a szalagon az aktuális karaktert, majd ennek, és az eddigi állapotnak a hatására változik az állapota, módosíthatja az író-olvasó fej alatt látott karaktert, majd lép a szalagon egyet jobbra vagy balra, esetleg helyben marad.

Az átmeneti függvény azt írja le, hogy egy lépés során (a gép belső állapotától és az olvasott karaktertől függően) mit tesz a gép: δ:(q,a)→(q',b,D) azt adja meg, hogy ha a gép q állapotban van, a betűt olvas a szalagról akkor q' állapotba kerül, az olvasott a-t felülírja b-vel, majd a szalagot olvasó és író fej D irányba lép.

A Turing-gép egy számítás során a kezdőhelyzetből indulva az átmeneti függvénynek megfelelő lépések sorozatát hajtja végre. Ha a szalag elején balra lépne (azaz "leesne" a szalagról az író-olvasó fej), akkor a gép hibával megáll. Egyébként a működés akkor ér véget, ha nem tud lépni, elakad, mert az adott belső állapot - olvasott karakter kombinációra nincs értelmezve az átmeneti függvény. A gép akkor fogadja el a bemeneti szót, ha ez az elakadás F-beli állapotban történik.

**Fontos megjegyezni, hogy semmi nem garantálja, hogy egy adott bemenettel a Turing-gép valaha is le fog állni. A következő lehetőségek vannak:**

a gép előbb-utóbb leáll elfogadó állapotban, tehát a szót a gép elfogadj

a gép előbb-utóbb leáll hibával ("leesik a szalagról") vagy leáll nem elfogadó állapotban

a gép az adott bemenet hatására soha sem áll meg.

A második és harmadik esetben a gép a bemenetet nem fogadja el. Ha hangsúlyozni szeretnénk, hogy a második eset történt (megállt a gép, de nem elfogadóan), akkor azt mondjuk, hogy a gép a szót elutasítja.

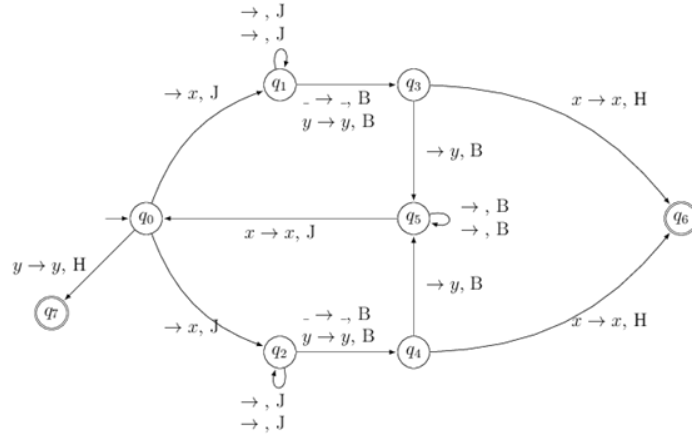
**9.2. Megjegyzés** A definíció a determinisztikus, hiányos automatákra emlékeztet. Fontos formai megkötés, hogy most az elfogadás feltétele más: a véges automatáknál és veremautomatáknál megköveteltük, hogy elfogadó számításnál a bemeneti szót végig kell olvasni, nem szabad, hogy a számítás elakadjon. Most viszont a végigolvasás nem szükséges, és elfogadni csak elakadás esetén lehetséges. Lehetne az elfogadási feltételt a korábbiakhoz hasonlóan megadni, ezzel is egy ekvivalens modellt kapnánk, de a jelen forma az általánosan elterjedt, egyszerűbben használható.

**9.3. Definíció Az M Turing-gép által elfogadott nyelv:**

L(M)={w∈Σ:M elfogadja w szt}

A Turing-gépeket is lehet gráffal ábrázolni, ilyenkor az átmeneteket jelző nyilakon szerepel, hogy milyen olvasott karakter hatására milyen karaktert ír a gép, illetve milyen irányba lép tovább.

**9.4. Példa Az ábrán látható** Turing-gép a Σ={a,b} feletti palindromokat fogadja el.



A gép működésének lényege, hogy az első karaktertől kezdve sorban ellenőrzi, hogy a szó i-edik karaktere megegyezik-e a hátulról nézve i-edik karakterrel. Egy ellenőrzés során beolvassa az első, még nem ellenőrzött karaktert a szó elejéről, belső állapotában megjegyzi, hogy mit olvasott, majd elbaktat a szó végére a megfelelő karakterhez és ellenőrzi, hogy egyezik-e az állapotában megjegyzett karakterrel. Kicsit pontosabban a gép működése a következő:

q0 állapotban olvassa be a következő ellenőrzendő karaktert a szó első feléből. Ha ez a, akkor a felső ágon (q1,q3,q5) halad majd tovább, ha b, akkor az alsón (q2,q4,q5), így jegyezve meg belső állapotában, hogy mit olvasott. Továbblépés előtt az a (vagy b) karaktert x-re cseréli, ezzel jelezve, hogy ezt a karaktert már ellenőrzés alá vonta. A továbbiakban az a esetet írjuk le, a b eset teljesen hasonló.

q1 állapotban olvas és lép jobbra, amíg el nem éri azt a karaktert, amivel az összehasonlítást el kell végeznie. Ez a karakter vagy a szó végén van (utána már \_ karakter következik) vagy az első olyan karakter előtt, amit már egyezőnek találtunk (az egyezett helyeket y-nal jelöli a gép). Ha tehát ezen két eset valamelyikét ( \_ vagy y) látja, akkor q3 állapotba kerül és a szalagon visszalép balra egyet, így pontosan arra a karakterre érkezve, amivel az összehasonlítást el kell végeznie.

A tényleges összehasonlítás q3-ban történik: ha itt a-t talál, akkor ezt a karaktert is ellenőrzöttnek jelöli azzal, hogy átírja y-ra, majd q5 állapotban visszasétál a szalagon balra, amíg x-et nem lát (ekkor elérte a szó elejének ellenőrzött részét). Ilyenkor q0-ba kerül és jobbra lép, készen állva a következő ellenőrzésre. Ha q3-ban b-t talál (nincs egyezés), akkor elakad, a szót nem fogadja el, hiszen q3 nem elfogadó.

A gép kétféleképpen juthat elfogadó állapotba. Ha páros hosszú palindromunk van, akkor az utolsó ellenőrzés (a szó közepén levő aa vagy bb megtalálása) után már nem talál a q0 állapotban újabb ellenőrizendő karaktert, csak y-t, ekkor az elfogadó q7 állapotba megy, ott elakad, tehát elfogad. Ha páratlan hosszú palindromunk van, akkor a szó közepi a vagy bx-re cserélése után rögtön egy y következik a szalagon (ennek felel meg az az eset, amikor q3-ban x-et olvas a gép), ekkor a q6 elfogadó állapotba lép és elakad.

A továbbiakban fontos szerepet kapnak azok a nyelvek, melyekhez létezik a nyelvet elfogadó Turing-gép. Különösen érdekesek lesznek azok a nyelvek, melyekhez olyan Turing-gép is szerkeszthető, ami nem csak elfogadja a nyelvet, hanem még az a szép tulaj-donsága is megvan, hogy a gép mindig megáll. Ez fontos tulajdonsága a gépnek, hiszen ha egy nyelv esetén szeretnénk tetszőleges szóról eldönteni, hogy az eleme-e a nyelvnek vagy sem, akkor egy olyan gép nem nagy segítség, ami nem biztos, hogy valaha is megáll és választ ad.

A fenti két fogalmat ragadja meg a következő két definíció.

**9.5. Definíció** Az L nyelv rekurzívan felsorolható, ha van olyan M Turing-gép, hogy L(M)=L. Azt is szoktuk mondani, hogy az L nyelv ekkor felismerhető. A rekurzívan felsorolható nyelvek halmazát RE-vel jelöljük.

**9.6. Definíció** Az L nyelv rekurzív, ha van olyan M Turing-gép, hogy L(M)=L és M minden bemenet esetén véges sok lépésben megáll. Azt is szoktuk mondani, hogy az L nyelv ekkor eldönthető. A rekurzív nyelvek halmazát R-rel jelöljük.

Ha egy nyelvre van őt eldöntő (a definíció szerint mindig megálló) Turing-gép, azaz a nyelv rekurzív, akkor ezen nyelv esetén a nyelvbe tartozás problémája tetszőleges szóra megoldható: a Turing-gépet lefuttatjuk a szóval és mivel mindig megáll, biztosan választ kapunk arra, hogy a szó eleme-e a nyelvnek.

Ha egy nyelvről csak azt tudjuk, hogy rekurzívan felsorolható, azaz csak olyan Turing-gépet ismerünk, ami elfogadja ugyan a nyelvet (a nyelv szavaira megáll elfogadóban), de nem feltétlenül áll meg minden inputon (a nyelven kívüli szavakra vagy elutasítóban áll meg vagy sosem áll meg), akkor erre a nyelvre ezt a Turing-gépet nem tudjuk a nyelvbe tartozás problémájának megoldására használni: ha beadunk a gépnek egy szót és sokáig nem kapunk választ (a gép nem áll meg), akkor nem tudhatjuk, hogy ez azért van, mert ez egy kivételesen hosszú számítás, aminek nemsokára vége lesz vagy pedig a gép sosem fog leállni.

A definíciókból azonnal adódik, hogy R⊆RE. Nemsokára látni fogjuk, hogy a tartalmazás valódi, mutatni fogunk rekurzívan felsorolható, de nem rekurzív nyelveket.

Látunk majd példát olyan nyelvre is, ami még csak nem is rekurzívan felsorolható, de azt már most is be tudjuk bizonyítani (konkrét nyelv megadása nélkül), hogy ilyennek léteznie kell.

**9.7. Lemma Tetszőleges Σ ábécé esetén van olyan L⊆Σ∗ nyelv, amelyre L∉RE.**

Bizonyítás. Az állítás, a 4.17 tételhez hasonló számossági megfontolásból következik. Egy Turing-gép leírás véges hosszú (például egy szónak tekinthető a gép formális leírása azzal a hetessel, ami definiálja), azt pedig tudjuk, hogy véges hosszú szóból egy véges ábécé felett megszámlálhatóan végtelen sok van. Ezért a Turing-gépek is megszámlálhatóan végtelen sokan vannak, így az általuk felismerhető nyelvek száma is megszámlálhatóan végtelen.

Az összes Σ feletti nyelv számossága azonban kontinuum, mert a nyelvek halmaza éppen a megszámlálhatóan végtelen Σ∗ halmaz hatványhalmaza. Mivel a kontinuum nagyobb számosság, mint a megszámlálhatóan végtelen, ezért biztosan van olyan nyelv, amely nem ismerhető fel Turing-géppel. [QED]

**9.8. Megjegyzés** Vegyük észre, hogy a Turing-gép általánosítása mind a véges automatának, mind a veremautomatának: mindkét automatát könnyen lehet szimulálni Turing-géppel, az állapotokat a Turing-gép állapotaiban őrizzük, a veremautomata vermét pedig a szalagon tároljuk.

A véges automatánál annyival van több lehetősége a Turing-gépnek, hogy a fej mindkét irányba mozoghat a szalagon és a fej írni is tud. Meg lehet mutatni, hogy ebből a két különbségből az utóbbi a lényeges, mert egy olyan véges automata is csak a reguláris nyelveket tudja elfogadni, ami mindkét irányban tud mozogni a szalagon.

### **Nemdeterminisztikus Turing-gép**

A következő Turing-gép változat, amit megvizsgálunk, a nemdeterminisztikus Turing-gép. Természetesen adódik a kérdés, hogy nagyobb-e ennek a verziónak a számítási ereje, mint a determinisztikusnak, azaz több nyelvet lehet-e elfogadni vele. Definiáljuk most az egyszalagos nemdeterminisztikus Turing-gépet. (A nemdeterminisztikus Turing-gépek esetén is definiálható többszalagos gép, amely szi-mu-lál-ha-tó egy egyszalagossal, ennek bizonyítása ugyanúgy megy, mint a determinisztikus esetben).

9.13. Definíció A nemdeterminisztikus Turing-gépeket egy T=(Q,Σ,Γ,q0,\_,F,δ)T=(Q,Σ,Γ,q0,\_,F,δ) hetessel írhatjuk le, ahol Q,Σ,Γ,q0,\_,FQ,Σ,Γ,q0,\_,F szerepe azonos a Turing-gépek esetén definiálttal, valamint:

δ(q,a)⊆Q×Γ×{J,B,H}δ(q,a)⊆Q×Γ×{J,B,H}

A fő különbség a determinisztikus változathoz képest tehát az, hogy most egy állapot és egy olvasott karakter hatására többféle műveletet is végezhet a Turing-gép. Ellentétben a véges automatával és a veremautomatával, a nemdeterminisztikus Turing-gép esetén nem kell külön εε átmenetekkel bajlódnunk. Ha meg akarjuk engedni, hogy a gép léphessen úgy, hogy a fej alatt álló karaktert nem olvassa el, akkor ezt megtehetjük egy olyan szabálycsoport létrehozásával, melyben a fej bármit is olvas, ugyanazt írja vissza és a fej helyben marad.

Egy adott bemeneti szó esetén a gép lehetséges számításait egy úgy nevezett számítási fával is leírhatjuk, mely-nek csúcsai a gép konfigurációit reprezentálják (mi van a szalagon, milyen állapotban van a gép, hol áll a fej), elágazásai pedig az adott konfigurációban lehetséges továbblépéseknek felelnek meg. A fa gyökere a kezdőkonfiguráció.

9.14. Definíció A nemdeterminisztikus Turing-gép akkor fogadja el a bemenetét, ha a számítási fában van elfogadó levél, azaz olyan konfiguráció, amiből nem lehetséges továbblépés és a hozzá tartozó belső állapot elfogadó.

Ez úgy is mondható, hogy akkor fogad el a nemdeterminisztikus Turing-gép, ha van elfogadó számítása. Az előfordulhat, hogy a többi számítás között vannak elutasító vagy esetleg soha véget nem érő változatok is.

9.15. Definíció Az M nemdeterminisztikus Turing-gép által elfogadott nyelv azon szavak halmaza, melyeket a gép elfogad, vagyis amely szavakra legalább egy elfogadó számítás létezik.

9.16. Tétel Bármely nemdeterminisztikus Turing-gép szimulálható determinisztikus Tu-ring--géppel.

Bizonyítás. Vázolni fogjuk, hogy hogyan lehet egy nemdeterminisztikus Turing-géphez olyan determinisztikus Turing-gépet készíteni, ami ugyanazt a nyelvet fogadja el.

Legyen M a nemdeterminisztikus Turing-gép, és tekintsük a számítási fáját egy tetszőleges bemeneten. Ebben a számítási fában minden csúcsnak (minden konfigurációnak) legfeljebb 3|Q||Γ| gyerek-konfigurációja van: |Q| darab lehetséges új állapot van, Γ féle új karaktert írhat le a fej, majd három lehetséges irányba léphet. Az M' Turing-gép a következőképpen fog működni az adott bemeneten:

* Szélességi bejárással végigmegy M számítási fáján: ehhez minden csúcsban nyilvántartja (MM átmeneti függvényének ismeretében), hogy mik a lehetséges tovább-lépések és ezek közül melyeket nézte már meg eddig.
* Ha valamelyik szinten elfogadó levelet talál, akkor megáll és elfogad.
* Ha végigment a teljes fán, és nem talált elfogadó levelet, akkor elutasít.
* Egyébként (azaz ha a fa végtelen és nincs elfogadó levele), nem áll meg.

[QED]

9.17. Megjegyzés Vegyük észre, hogy a mélységi bejárása itt nem használható - ha van egy véget nem érő számítási ág, akkor a bejárás nem jutna el a további ágakba.

Ha tehát egy nyelvről meg akarjuk mutatni, hogy rekurzívan felsorolható, akkor megtehetjük, hogy ennek belátására nemdeterminisztikus Turing-gépet konstruálunk rá, hiszen a determinisztikus és nemdeterminisztikus Turing-gépek ekvivalensek, amennyiben az általuk elfogadott nyelvek halmazát tekintjük. Fontos azonban látni azt, hogy a nemdeterminisztikus Turing-géppel ekvivalens determinisztikus változat sokkal több lépés után jut elfogadó állapotba, mint amekkora a legrövidebb nemdeterminisztikus elfogadó számítás hossza. Erről a különbségről rész-letesen fogunk majd beszélni a 11. fejezetben.

**Nevezetes osztályok**