Bevezetés a számításelméletbe

egyetemi jegyzet

Gazdag Zsolt

Eötvös Loránd Tudományegyetem, Informatikai Kar,
Algoritmusok és Alkalmazásaik Tanszék

Lektorálta:
Dr. Németh L. Zoltán
egyetemi adjunktus

A jegyzet az
ELTE IK 2015. évi Jegyzettámogatási pályázat
támogatásával készült

Budapest, 2015

Előszó

Ennek a jegyzetnek az a fő célja, hogy segítséget nyújtson az ELTE Informatikai Karon oktatott „Logika és számításelmélet” című tárgy számításelmélettel kapcsolatos részének könnyebb megértéséhez. A bevezetőn kívül két fő fejezetből áll. A kiszámíthatósággal kapcsolatos részben azt vizsgáljuk, hogy mit is jelent pontosan az, hogy egy probléma algoritmikusan megoldható. A bonyolultságelmélet alapjait ismertető fejezetben főként azzal foglalkozunk, hogy az algoritmikusan megoldható problémák közül melyek a nehéz problémák, vagyis azok, melyek megoldására az idő- vagy tárigény szempontjából nem ismert hatékony algoritmus.

A jegyzet igyekszik a fenti témaköröket olvasmányosan, az olvasó számára minél könnyebben érthetően tárgyalni. Mivel a témakör szorosan kapcsolódik a formális nyelvek elméletéhez és a matematikai logikához, a bevezetőben ismertetjük ezen témakörök alapjait is. Bár a szükséges matematikai fogalmak is ismertetésre kerülnek, feltételezzük, hogy az olvasó rendelkezik alapvető diszkrét matematikai, formális nyelvi és logikai ismeretekkel.

Mivel, ahogy azt már említettük, a számításelmélet oktatása az ELTE Informatikai Karán szoros kapcsolatban van a logika alapjainak oktatásával, a jegyzet mind az eldönthetetlenséggel, mind a bonyolultságelmélettel kapcsolatos részében különös figyelmet szentelünk a vizsgált kérdések logikai vonatkozásainak is.

A számításelmélet témakör gazdag idegen és magyar nyelvű irodalommal rendelkezik, melyek közül számosat megtalál az olvasó az irodalomjegyzékben. A jegyzet sokszor ezen műveket követve ismerteti a számítás- és bonyolultságelmélet klasszikus eredményeit és azok bizonyításait.

Budapest, 2015. november

Gazdag Zsolt

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. Miért érdekes a számításelmélet?

Az hogy egy adott probléma vajon megoldható-e vagy sem, és ha igen, akkor könnyű-e megoldani, alapvető kérdés egy informatikus számára. Jól ismert például, hogy egy irányított gráfban hatékonyan kereshető adott két csúcs között út. Másrészt viszont, olyan út keresése, ami minden csúcsot pontosan egyszer érint már nem tűnik könnyű feladatnak. Nem ismert ugyanis ezen problémát megoldó olyan algoritmus, melynek lépésszáma ne lenne kezelhetetlenül nagy a gráf méretének növekedésével. Mindazonáltal ez a probléma még mindig megoldható. Ismerünk-e vajon olyan problémát, ami nem is oldható meg? Erre már nem biztos, hogy mindenki számára egyértelmű a válasz. Pedig az alábbi egyszerű probléma megoldhatatlan: adott egy p egész együtthatós, többváltozós polinom. Keressük meg ennek a polinomnak a gyökeit az egész számok halmazán. Persze itt nem egy konkrét polinomra kell gondolni, hanem arra, hogy nincs olyan algoritmus, ami tetszőleges polinomra megválaszolja a fenti kérdést. Másképpen fogalmazva, bármely algoritmus ami a fenti problémát hivatott megoldani, bizonyos bemenetekre rossz választ ad vagy végtelen sokáig számol.

A jegyzetben először a fenti kérdéseket feszegetjük majd. Ehhez bevezetünk egy jól ismert algoritmusmodellt, a Turing-gépet. Bár a Turing-gép egyike a legáltalánosabb algoritmusmodelleknek, látni fogjuk, hogy bizonyos problémák nem oldhatók meg vele. Végül a Turing-gépek idő- és tárigénye kapcsán megismerjük a bonyolultságelmélet nevezetes problémaosztályait, úgy mint a **P** és **NP** idő-, valamint az **L**, **NL** és **PSPACE** tárbonyolultsági osztályokat. Azt is megnézzük, hogy milyen nevezetes problémák számítanak nehéznek a fent említett osztályokban.

A számításelmélet szorosan kapcsolódik a matematikai logikához és a formális

nyelvek elméletéhez, ezért a matematikai alapfogalmak ismertetése után röviden áttekintjük ezen témaköröknek a könyvben használt alapfogalmait is.

1.2. Matematikai alapfogalmak

Tetszőleges H halmaz esetén $\mathcal{P}(H)$-val jelöljük a H részhalmazainak halmazát, azaz H *hatványhalmazát*. Egy tetszőleges U univerzumra és $H \subseteq U$ halmazra, $\overline{H}$ jelöli a H *komplementerét*, vagyis az $\{u \in U \mid u \notin H\}$ halmazt. Véges H esetén $|H|$-val jelöljük a H elemeinek számát. A természetes számok halmazát $\mathbb{N}$ jelöli, és tetszőleges $n \in \mathbb{N}$ számra $[n]$ jelöli az $\{1, 2, \ldots, n\}$ halmazt (speciálisan, ha $n = 0$, akkor $[n]$ az üres halmazt, azaz $\emptyset$-t jelöli). Egy n valós számra $\lfloor n \rfloor$ illetve $\lceil n \rceil$ jelöli az n *alsó* illetve *felső egész részét*.

Tetszőleges $n \geq 1$ számra és H halmazra H^n jelöli a H halmaz *n-szeres Descartes-szorzatát*. Egy *n változós H-feletti reláció* alatt a H^n halmaz egy részhalmazát értjük. Ha ρ egy H-feletti n-változós reláció, akkor ρ-t gyakran egy olyan $\hat{\rho} : H^n \to \{igaz, hamis\}$ leképezésnek tekintjük, melyre a következő teljesül. Minden $a_1, \ldots, a_n \in H$ esetén, $\hat{\rho}(a_1, \ldots, a_n) = igaz$ akkor és csak akkor, ha $(a_1, \ldots, a_n) \in \rho$. Ha nem okoz félreértést, akkor $\hat{\rho}$ helyett gyakran ρ-t írunk.

Különösen fontosak lesznek számunkra a *kétváltozós relációk*. Legyen $\rho \subseteq H \times H$ egy kétváltozós reláció és tegyük fel, hogy $(a, b) \in \rho$. Ekkor használni fogjuk az $a\rho b$ jelölést is. Továbbá azt mondjuk, hogy ρ *reflexív* ha minden $a \in H$-ra, $a\rho a$ és *tranzitív*, ha minden $a, b, c \in H$-ra, $a\rho b$ és $b\rho c$ következménye, hogy $a\rho c$. Végezetül, a ρ *reflexív és tranzitív lezártja* alatt azt a legszűkebb reflexív és tranzitív ρ^* relációt értjük, amelyre $\rho \subseteq \rho^*$.

Legyen H egy halmaz és $A \subseteq H$. Az $f_A : H \to \{0, 1\}$ függvényt az A halmaz *karakterisztikus függvényének* nevezzük, ha minden $a \in H$ esetén, $a \in A$ akkor és csak akkor, ha $f(a) = 1$.

1.2.1. Gráfok

A jegyzetben számos gráfokkal kapcsolatos problémát vizsgálunk majd. Egy *G gráf csúcsok V* és *élek E* halmazából áll. Az élek lehetnek címkézettek valamilyen objektumhalmaz elemeivel, de alapesetben ezt nem követeljük meg. A *G irányítatlan gráf*, ha az E halmaz V-beli elempárok rendezetlen halmaza, azaz $E \subseteq \{\{a, b\} \mid a, b \in V\}$. A *$G$ irányított gráf*, ha az E V-beli elempárok rendezett halmaza, azaz $E \subseteq V \times V$ (vagyis ekkor E egy kétváltozós V-feletti reláció).

Legyen $G = (V, E)$ egy irányítatlan gráf és $u = a_1 a_2 \ldots a_n$ ($n \geq 2, a_i \in V, i \in [n]$). Azt mondjuk, hogy u egy *G-beli séta*, ha minden $i \in [n-1]$-re $\{a_i, a_{i+1}\} \in E$. Továbbá egy u sétát *útnak* nevezünk, ha u-ban a csúcsok páronként különböznek. Végül u egy *kör*, ha $a_1 = a_n$ és az $a_1, \ldots, a_{n-1}$ csúcsok

páronként különböznek. Egy G-beli utat illetve kört *Hamilton-útnak* illetve *Hamilton-körnek* nevezünk, ha az tartalmazza a G összes csúcsát. G-t *körmentesnek* nevezzük, ha nem tartalmaz kört. Irányított gráfokban a fenti fogalmak hasonlóan definiálhatók.

Legyen $G = (V, E)$ egy irányítatlan gráf. G *összefüggő*, ha bármely két csúcsa között vezet út. G *fa*, ha összefüggő és körmentes. Egy $G' = (V', E')$ fát a G *feszítő fájának* nevezünk, ha $V' = V$ és $E' \subseteq E$. Legyen $u \in V$. Az u *foka* alatt a következő számot értjük: $|\{v \in V \mid \{u, v\} \in E\}|$.

1.2.2. Matematikai logikai alapok

A számításelmélet szorosan kapcsolódik a matematikai logikához. A jegyzetben is számos olyan eredményt mutatunk meg, melyek valamilyen matematikai logikai kérdés eldöntésével kapcsolatosak. Bevezetőnk következő részeként megismerkedünk a matematikai logika egyes fejezeteinek alapfogalmaival.

1.2.2.1. Ítéletkalkulus

Az ítéletkalkulus a matematikai logika egy ága, mely formális kereteket biztosít olyan következtetések helyességének eldöntésére, melyek elemi állításokból (ítéletekből) épülnek fel. Az ítéletek fontos jellemzője, hogy igazságértékük egyértelműen eldönthető. Ítéletek például a „Süt a nap" vagy a „Lemegyek a térre", de nem tekinthető ítéletnek a „Laci magas" (mihez képest?), „Lejössz a térre?" (kérdő mondat) vagy „Bárcsak itt lennél" (óhajtó mondat).

Az összetett állítások könnyebb vizsgálatához az ítéleteket ítéletváltozókkal, a hétköznapi nyelv olyan szófordulatait, mint a „nem", „és", „vagy" és „ha ... akkor" pedig műveleti jelekkel helyettesítjük. Ennek megfelelően az ítéletkalkulus szintaxisát a következőképpen definiáljuk.

Ítéletváltozónak nevezünk egy olyan x változót, melynek értéke *igaz* vagy *hamis* lehet. Legyen $Var = \{x_1, x_2, \ldots\}$ ítéletváltozók egy megszámlálhatóan végtelen halmaza. Az *ítéletkalkulusbeli formulák* $Form$ halmaza a legszűkebb olyan halmaz amire a következők teljesülnek:

- Minden $x \in Var$ esetén $x \in Form$,
- ha $\varphi \in Form$, akkor $\neg\varphi \in Form$, és
- ha $\varphi_1, \varphi_2 \in Form$, akkor $(\varphi_1 \circ \varphi_2) \in Form$, ahol $\circ \in \{\wedge, \vee, \rightarrow\}$.

A $\neg, \wedge, \vee$ és $\rightarrow$ *műveleti jelek* nevei rendre *negáció*, *és*, *vagy* és *implikáció*. A továbbiakban, ha ez nem okoz félreértést, az ítéletkalkulusbeli jelzőt elhagyjuk a formulák elől. Tetszőleges $n \in \mathbb{N}$ esetén jelölje Var_n az $\{x_1, x_2, \ldots, x_n\}$ halmazt, $Form_n$ pedig azon $Form$-beli formulák halmazát, melyekben legfeljebb csak Var_n-beli változók szerepelnek.

A $Form_n$-beli formulákban szereplő műveletek szemantikáját a következő módon definiáljuk. Legyen $\varphi \in Form_n$ és $I : Var_n \to \{igaz, hamis\}$ egy függvény, amit *interpretációnak* is nevezünk. Az I-t kiterjesztjük $Form$-ra a következő induktív definícióval. $I(\varphi) = igaz$ akkor és csak akkor, ha a következők egyike teljesül:

- $\varphi \in Var_n$ és $I(\varphi) = igaz$,
- $\varphi = \neg\psi$ és $I(\psi) = hamis$,
- $\varphi = (\varphi_1 \wedge \varphi_2)$ és $I(\varphi_1) = I(\varphi_2) = igaz$,
- $\varphi = (\varphi_1 \vee \varphi_2)$ és ($I(\varphi_1) = igaz$ vagy $I(\varphi_2) = igaz$),
- $\varphi = (\varphi_1 \to \varphi_2)$ és ($I(\varphi_1) = hamis$ vagy $I(\varphi_2) = igaz$).

Azt mondjuk, hogy az I *kielégíti az* φ*-t* (jele: $I \models \varphi$), ha $I(\varphi) = igaz$. Továbbá a φ

- *kielégíthető*, ha van olyan I, hogy $I \models \varphi$,
- *kielégíthetetlen*, ha nem kielégíthető, és
- *tautológia* vagy *érvényes*, ha minden I-re $I \models \varphi$.

Legyen F egy formulahalmaz. Egy I *interpretáció kielégíti* F*-et* (jele: $I \models F$), ha kielégíti az összes F-beli formulát. Ha F nem kielégíthető, akkor azt mondjuk, hogy *kielégíthetetlen*. Legyen φ egy formula. A φ *az* F *logikai következménye* (jele: $F \models \varphi$), ha a következő teljesül. Minden I interpretációra, ha $I \models F$, akkor $I \models \varphi$. Végezetül a φ_1 és φ_2 formulákra azt mondjuk, hogy *ekvivalensek*, ha minden I interpretációra $I \models \varphi_1$ akkor és csak akkor, ha $I \models \varphi_2$.

A definíciók egyszerű következményei az alábbi állítások, melyek helyességének belátását az olvasóra bízzuk.

1.1. Tétel

Legyen F egy formulahalmaz és φ egy további formula. Akkor a következők teljesülnek.

- φ akkor és csak akkor kielégíthetetlen, ha $\neg\varphi$ tautológia.
- $F \models \varphi$ akkor és csak akkor, ha $F \cup \{\neg\varphi\}$ kielégíthetetlen.

Most nézzünk egy példát az eddig tanultak alkalmazására!

1.2. Példa

Mutassuk meg ítéletkalkulusbeli formulák segítségével, hogy az alábbi következtetés helyes.

1. Vagy túlórázok, vagy kirándulni megyek.
2. Csak akkor nem javítom meg az autóm, ha túlórázok vagy nem kapok fizetésemelést.
3. Ha az autómat javítom meg, akkor nem tudok kirándulni menni.
4. Következik, hogy ha kirándulni megyek, akkor nem kapok fizetésemelést.

A fenti állításokat reprezentáló formuláinkban az elemi állításokat jelölő ítéletváltozók, rendhagyó módon, az elemi állítások kezdőbetűi lesznek. Ezek szerint t, k, m, f ítéletváltozók a következő állításokat jelölik. t : *túlórázok*, k : *kirándulni megyek*, m : *megjavítom az autómat* és f : *fizetésemelést kapok*. Ezek után a fenti állításoknak megfelelő formulák a következők.

1. $\varphi_1 = (t \vee k) \wedge \neg(t \wedge k)$ (az első állításunkban szereplő „vagy" valójában egy „kizáró vagy", azaz nem engedi meg, hogy a két elemi állítás egyszerre teljesüljön),
2. $\varphi_2 = \neg m \to (t \vee \neg f)$ (gondoljuk végig: a „Csak akkor" utáni állítás teljesülése maga után kell vonja a „ha" utáni rész teljesülését, hisz éppen azt az esetet zárjuk ki az állításunkkal, hogy a „Csak akkor" utáni rész teljesül, de a „ha" utáni rész nem),
3. $\varphi_3 = m \to \neg k$,
4. $\varphi_4 = k \to \neg f$.

Az 1.1. tétel alapján a következtetés helyességének belátásához elég megmutatni, hogy az $F = \{\varphi_1, \varphi_2, \varphi_3, \neg\varphi_4\}$ halmaz kielégíthetetlen. Nézzük meg, hogy milyen I interpretáció elégítheti ki összes F-beli formulát! Mivel $\neg\varphi_4$ ekvivalens a $k \wedge f$ formulával, kapjuk, hogy $I(k) = I(f) = igaz$. Ekkor $I(\neg k) = hamis$ miatt azt kapjuk, hogy $I \models \varphi_3$ csak akkor teljesül, ha $I(m) = hamis$. De akkor az $I(\neg m) = igaz$ miatt $I \models \varphi_2$ csak akkor, ha $I \models t \vee \neg f$. Tudjuk, hogy $I(\neg f) = hamis$, tehát $I \models t \vee \neg f$ csak akkor teljesül, ha $I(t) = igaz$. De akkor, mivel $I(k) = igaz$, $I \models t \wedge k$, azaz $I \not\models \neg(t \wedge k)$. Ebből azt kapjuk, hogy az egyetlen olyan interpretáció, ami kielégíti a φ_2, φ_3 és a $\neg\varphi_4$ formulát nem elégíti ki φ_1-et, azaz F kielégíthetetlen.

Szükségünk lesz az ítéletkalkulusbeli formulák egy speciális osztályára. Ehhez

bevezetjük a következő fogalmakat. *Literálnak* nevezünk egy x vagy $\neg x$ alakú formulát, ahol $x \in Var$. Egy *literál alapja* az az ítéletváltozó, ami a literálban szerepel. *Klóznak* hívunk egy $l_1 \vee l_2 \vee \ldots \vee l_n$ ($n \in \mathbb{N}$) alakú formulát, ahol $l_1, \ldots, l_n$ páronként különböző alapú literálok. Végül *konjunktív normálformának* (röviden KNF-nek) nevezünk egy $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ ($m \geq 1$) alakú formulát, ahol minden $i \in [n]$-re C_i egy klóz. Ismert, hogy minden ítéletkalkulusbeli formulához megadható egy ekvivalens KNF.

1.2.2.2. Elsőrendű logika

Tekintsük a következő állításokat: „Aki a virágot szereti, rossz ember nem lehet" és „Aki az állatokat szereti az jó ember". Az ítéletkalkulusban ezek egymástól független elemi állításoknak tekinthetők. Viszont érezhető, hogy ha egy-egy ítéletváltozóval formalizáljuk ezt a két állítást, akkor nem elég precízen formalizáltunk, hiszen ezen két állítás együtt azt jelenti, hogy „Aki a virágokat vagy az állatokat szereti, az jó ember". Az ilyen állítások formalizálására alkalmas az elsőrendű logika. Itt az állítások igazságértéke (például az, hogy „x szereti y-t") függ attól, hogy milyen paraméterekkel mondtuk ki az adott állítás. Például „Kata szereti a virágokat" lehet *igaz*, míg „Pali szereti az állatokat" lehet *hamis* egy-egy konkrét Kata és Pali esetén. Továbbá az az állítás, hogy „Van valaki aki szereti a virágokat" nyilvánvalóan *igaz*, ha az embereket halmazát tekintjük a „valaki" helyére írható „egyedek" halmazának. Láthatjuk tehát, hogy az ítéletek helyett paraméteres állításokat alkalmazva precízebben tudunk formalizálni.

1.3. Definíció

Egy $\mathcal{L}$ *elsőrendű nyelv* szimbólumhalmaza a következő részekből áll:

- a *predikátumszimbólumok* $Pred$,
- *függvényszimbólumok* $Func$ és
- a *konstansszimbólumok* $Const$ véges halmazai;
- az *egyedváltozók* $Ind = \{x_1, x_2, \ldots\}$ megszámlálhatóan végtelen halmaza;
- a *műveleti jelek* $\{\neg, \wedge, \vee, \rightarrow\}$ halmaza;
- az *univerzális kvantor* ($\forall$) és az *egzisztenciális kvantor* ($\exists$), és végül
- az „(", „)" és „ ," jelek (azaz a nyitó és záró zárójelek, valamint a vessző).

Feltesszük, hogy minden $s \in Const \cup Pred \cup Func$ szimbólum rendelkezik egy nemnegatív egész értékű aritással, amit $ar(s)$-el jelölünk. A konstansszimbólumok aritása mindig 0.

Ezen szimbólumok felhasználásával először az $\mathcal{L}$ nyelv *kifejezéseinek* vagy más szóval *termjeinek* $Term$ halmazát definiáljuk. Ez a legszűkebb olyan halmaz, melyre

- $Ind \cup Const \subseteq Term$, és
- ha $f \in Func$ és $t_1, \ldots, t_{ar(f)} \in Term$, akkor $f(t_1, \ldots, t_{ar(f)}) \in Term$.

Ezután az $\mathcal{L}$ *atomi formuláinak* $AForm$ halmaza az a legszűkebb halmaz, melyre a következő teljesül. Ha $p \in Pred$ és $t_1, \ldots, t_{ar(p)} \in Term$, akkor $p(t_1, \ldots, t_{ar(f)}) \in AForm$.

Végül az $\mathcal{L}$ elsőrendű nyelv formuláinak halmaza alatt a legszűkebb olyan $Form$ halmazt értjük, melyre

- $AForm \subseteq Form$,
- ha $\varphi \in Form$, akkor $\neg\varphi \in Form$,
- ha $\varphi_1, \varphi_2 \in Form$, akkor $(\varphi_1 \circ \varphi_2) \in Form$, ahol $\circ \in \{\wedge, \vee, \rightarrow\}$ és
- ha $x \in Ind$ és $\varphi \in Form$, akkor $\exists x\varphi \in Form$ és $\forall x\varphi \in Form$.

Legyenek φ és ψ elsőrendű logikai formulák. Azt mondjuk, hogy ψ a φ *részformulája*, ha ψ előfordul a φ fenti definíció szerinti rekurzív előállítsa során.

1.4. Példa

Tekintsük azt az elsőrendű $\mathcal{L}$ nyelvet, melyben $Pred = \{p, q\}$, $Func = \{f\}$ és $Const = \{a\}$. Továbbá $ar(p) = ar(q) = ar(f) = 2$. Ekkor az alábbiak mind $\mathcal{L}$-beli formulák:

1. $\varphi_1 = \forall x p(x, a)$
2. $\varphi_2 = \forall x \exists y q(f(x, y), a)$,
3. $\varphi_3 = \forall x(\forall y q(f(y, x), y) \rightarrow p(x, a))$.

Amint az várható, az interpretáció fogalma az elsőrendű logikában jóval általánosabb lesz mint az ítéletkalkulusban.

1.5. Definíció

Az $\mathcal{L}$ nyelv egy interpretációja egy olyan $I = \langle U, I_{Pred}, I_{Func}, I_{Const}\rangle$ struktúra, melyre a következők teljesülnek.

- U tetszőleges nem üres halmaz,

- I_{Pred} olyan függvény, ami miden $p \in Pred$-hez hozzárendel egy $ar(p)$ változós, U-feletti p^I relációt.
- I_{Func} olyan függvény, ami minden $f \in Func$-hoz hozzárendel egy $f^I : U^{ar(f)} \to U$ függvényt.
- I_{Const} minden $a \in Const$ konstansszimbólumhoz hozzárendel egy $a^I \in U$ elemet.

Egy $\kappa : Ind \to U$ függvényt *változókiértékelésnek* nevezünk. Legyen $x \in Ind$ és κ egy változókiértékelés. A κ egy x-variánsa egy olyan κ' változókiértékelés, melyre igaz, hogy minden $y \in Ind$, $y \neq x$ esetén, $\kappa'(y) = \kappa(y)$.

Legyen I egy interpretáció és κ egy változókiértékelés. Egy $t \in Term\ I$ és κ melletti értékét $|t|^{I,\kappa}$-val jelöljük és a következőképpen definiáljuk.

- Ha $t = a$ valamely a konstansszimbólumra, akkor $|t|^{I,\kappa} = a^I$,
- ha $t = x$ valamely x egyedváltozóra, akkor $|t|^{I,\kappa} = \kappa(x)$,
- ha $t = f(t_1, \ldots, t_n)$ valamely $f \in Func$, $ar(f) = n$, $t_1, \ldots, t_n \in Term$ esetén, akkor $|t|^{I,\kappa} = f^I(|t_1|^{I,\kappa}, \ldots, |t_n|^{I,\kappa})$.

Ezek után egy $\varphi \in Form$ formula értéke I-ben κ mellet (jele: $|\varphi|^{I,\kappa}$) *igaz* vagy *hamis* a következők szerint:

- ha $\varphi = p(t_1, \ldots, t_n)$ valamely $p \in Pred$, $ar(p) = n$, $t_1, \ldots, t_n \in Term$ esetén, akkor $|\varphi|^{I,\kappa} = igaz \Leftrightarrow (|t_1|^{I,\kappa}, \ldots, |t_n|^{I,\kappa}) \in p^I$,
- ha $\varphi = \neg\psi$ vagy $\varphi = \varphi_1 \circ \varphi_2$ ($\circ \in \{\wedge, \vee, \to\}$), akkor $|\varphi|^{I,\kappa}$ értéke a $|\psi|^{I,\kappa}$, $|\varphi_1|^{I,\kappa}$ és $|\varphi_2|^{I,\kappa}$ értékek alapján az ítéletkalkulusban látott módon számolható ki,
- ha $\varphi = \exists x\psi$, akkor $|\varphi|^{I,\kappa} = igaz$ akkor és csak akkor, ha van olyan κ' x-variánsa a κ-nak, hogy $|\psi|^{I,\kappa'} = igaz$, végül
- ha $\varphi = \forall x\psi$, akkor $|\varphi|^{I,\kappa} = igaz$ akkor és csak akkor, ha κ minden κ' x-variánsára $|\psi|^{I,\kappa'} = igaz$.

Egy formula vagy formulahalmaz kielégíthetősége és kielégíthetetlensége, egy formula érvényessége és a logikai következmény fogalma az ítéletkalkulusban látott módon definiálható.

Legyen φ egy formula, és tekintsük $x \in Ind$ egy előfordulását φ-ben. Azt mondjuk, hogy x ezen előfordulása *kötött*, ha x a φ egy $\exists x\psi$ vagy $\forall x\psi$ alakú

részformulájában szerepel. Ellenkező esetben x ezen előfordulása *szabad*. Ha φ-ben minden egyedváltozó minden előfordulása kötött, akkor φ-t *zárt formulának* vagy *mondatnak* nevezzük. Egyébként φ *nyitott*.

Legyen φ egy formula és I egy interpretáció. Világos, hogy ha φ zárt, akkor $|\varphi|^{I,\kappa}$ értéke független κ-tól. Ezért ebben az eseteben $|\varphi|^{I,\kappa}$ helyett $|\varphi|^{I}$-t írunk.

1.6. Példa

Tekintsük az 1.4. példában definiált elsőrendű nyelvet és a példában szereplő φ_1, φ_2 és φ_3 formulákat. Legyen továbbá $I = \langle U, I_{Pred}, I_{Func}, I_{Const}\rangle$ a következő interpretáció:

- $U = \mathbb{N}$,
- $p^I = \{(m,n) \mid m \geq n\}$, $q^I = \{(m,n) \mid m = n\}$ (azaz p-t és q-t rendre $\geq$ és $=$ relációkkal azonosítjuk),
- $\forall m, n \in \mathbb{N} : f^I(m,n) = m + n$ és
- $a^I = 0$.

Ebben az interpretációban a fenti formulák jelentése a következő:

1. φ_1 : minden m természetes számra, $m \geq 0$;
2. φ_2 : minden m természetes számhoz van olyan n természetes szám, hogy $m + n = 0$ és
3. φ_3 : minden m természetes számra, ha minden n természetes számra igaz, hogy $n + m = n$, akkor $m = 0$. Másképpen fogalmazva: ha egy m szám teljesíti $\mathbb{N}$-ben a zéruselem tulajdonságát, akkor az m megegyezik a 0-val.

Ezek alapján belátható, hogy $|\varphi_1|^I = |\varphi_3|^I = igaz$, de $|\varphi_2|^I = hamis$. Legyen I' az az interpretáció, ami megegyezik I-vel kivéve, hogy I' alaphalmaza az egész számok halmaza. Könnyen látható, hogy ekkor már mindhárom formula igaz I'-ben.

1.3. Formális nyelvi alapok

Akárcsak a természetes nyelvek, a formális nyelvek is szavakból épülnek fel, a szavak pedig egy véges, nemüres szimbólumhalmaz elemeiből állnak. Ezen szimbólumhalmazt *ábécének*, elemeit pedig *betűknek* nevezzük. Lényeges különbség azonban a formális nyelvi ábécék és a természetes nyelvek ábécéi között az, hogy az előbbiek tetszőleges szimbólumokból állhatnak. Így ábécének lehet te-

kinteni egy programozási nyelv tokenjeinek egy részhalmazát, például a $\Sigma_1 = \{if, then, else, do\}$ halmazt, vagy például azt a $\Sigma_2 = \{a, b, ..., x, y, z, 1, 2, ..., 0\}$ halmazt, melynek elemeiből felépíthető egy programozási nyelv azonosítóinak a halmaza. A fenti Σ_1 és Σ_2 ábécék uniója felett (ami persze szintén ábécé) megadható például az *if valtozo1 then do* szó (ami igazából már egy mondat, szokás is az irodalomban egy formális nyelv szavait mondatoknak nevezni).

Legyen Σ ábécé. Egy *Σ-feletti szón* a Σ betűinek egy tetszőleges véges (akár üres) sorozatát értjük. Azt a szót, ami nem tartalmaz egyetlen betűt sem, *üres szónak* nevezzük és ε-nal jelöljük. Jelölje Σ^* az összes Σ-feletti szót és Σ^+ a $\Sigma^* - \{\varepsilon\}$ halmazt. Jelölje továbbá $l(u)$ az $u \in \Sigma^*$ szó *hosszát*, és minden $a \in \Sigma$ esetén $l_a(u)$ az *u-beli a betűk számát*. Egy *Σ-feletti formális nyelven* Σ^* egy részhalmazát értjük. A továbbiakban, ha mást nem mondunk, nyelv alatt mindig formális nyelvet fogunk érteni. Lássunk néhány példát formális nyelvre!

1.7. Példa

Legyen $\Sigma = \{0, 1\}$. Ekkor $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$, az alábbiak pedig mind Σ-feletti nyelvek:

$$\emptyset, \quad \{\varepsilon\}, \quad \{\varepsilon, 0, 1, 10, 11\}, \quad \{0, 111, 0^5, 1^{10}\}, \quad L_1 = \{(01)^n \mid n \geq 0\},$$

$$L_2 = \{0^n 1^n \mid n \geq 0\}, \quad L_3 = \{u \in \Sigma^* \mid l_0(u) = l_1(u)\}.$$

A fenti példában szerepel néhány nyelv, melyek *szavak n-ik hatványát* tartalmazzák valamely n természetes számra. Egy tetszőleges u szóra és $n \in \mathbb{N}$ számra, $u^n = \varepsilon$, ha $n = 0$ és $u^n = uu^{n-1}$ egyébként.

Érdemes megjegyezni, hogy a fenti példában szereplő $\emptyset$ és $\{\varepsilon\}$ nyelvek természetesen nem egyenlőek: amíg $\emptyset$ az üres nyelv, mely nem tartalmaz egyetlen egy szót sem, addig $\{\varepsilon\}$ pontosan egy szót tartalmaz, mégpedig az üres szót.

1.3.1. Műveletek nyelveken

Az alábbiakban bemutatunk néhány alapvető, nyelveken értelmezett műveletet. Ezek közül az első csoport, melyeket *Boole-féle műveleteknek* nevezünk, a szokásos halmazelméleti műveletek: az *unió*, a *metszet* és a *komplementer*. Világos, hogy ha az L_1 és L_2 rendre Σ_1 illetve Σ_2-feletti formális nyelvek, akkor $L_1 \cup L_2$ és $L_1 \cap L_2$ is $\Sigma_1 \cup \Sigma_2$-feletti nyelvek, továbbá ha a L egy Σ-feletti formális nyelv, akkor a komplementere, azaz $\overline{L} = \Sigma^* - L$ is az. A másik műveletcsoport az úgynevezett *reguláris műveletek*, melyek az *unió*, a *konkatenáció* és a *Kleene-iteráció*. Látható, hogy az unió Boole-féle művelet valamint reguláris művelet is egyszerre. A további két műveletet az alábbi módon definiáljuk. Legyen L_1 és L_2 két nyelv és legyen $u \in L_1, v \in L_2$. Az *u és v szavak konkatenációját* úgy kapjuk, hogy a v-t az u után írjuk, azaz ha $\cdot$ jelöli a szavakon értelme-

zett konkatenáció műveletet, akkor $u \cdot v$ jelöli az u és v konkatenációját. A $\cdot$ műveleti jelet azonban általában nem írjuk ki, vagyis egyszerűen csak uv-t írunk. Ezek után az L_1 *és az* L_2 *konkatenációja* a következőképpen adódik: $L_1 \cdot L_2 = \{uv \mid u \in L_1, v \in L_2\}$. A $\cdot$ jelet általában itt is elhagyjuk, azaz $L_1 \cdot L_2$ helyett L_1L_2-t írunk.

Legyen L egy formális nyelv. Akkor az L *Kleene-iteráltja*, amit L^*-gal jelölünk, a következőképpen adódik: $L^* = \{u_1 \dots u_n \mid n \geq 0, u_1, \dots, u_n \in L\}$ (az $n = 0$ eset azt eredményezi, hogy $\varepsilon \in L^*$). A Kleene-iteráltat egy alternatív módon is definiálhatjuk. Ehhez először induktív módon definiáljuk az L *n-ik hatványát*, vagyis az L^n nyelvet ($n \in \mathbb{N}$). Ha $n = 0$, akkor legyen $L^n = \{\varepsilon\}$. Ha $n \geq 1$, akkor $L^n = L^{n-1}L$. Ezek után $L^* = \bigcup_{n=0}^{\infty} L^n$.

1.8. Példa

Legyen $L_1 = \{\varepsilon, a, b\}$ és $L_2 = \{0, 1\}$. Ekkor $L_1L_2 = \{0, 1, a0, a1, b0, b1\}$. Figyeljük meg, hogy mivel L_1 tartalmazza ε-t, az L_2 részhalmaza az L_1L_2-nek. Továbbá $L_1^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \dots\}$ és $L_2^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$. Vajon megegyezik-e $(L_1L_2)^*$ az $(L_1)^*(L_2)^*$ nyelvvel? Könnyen látható, hogy a két nyelv nem egyezik meg, hisz az $a0b1a1$ eleme az $(L_1L_2)^*$ nyelvnek, de nem eleme $(L_1)^*(L_2)^*$-nek.

Az 1.7. példában szereplő nyelvek közül az L_1-el, L_2-vel és L_3-mal jelöltek végtelen sok szót tartalmaznak, a többi nyelv véges. Egy véges nyelvet könnyen meg lehet adni az elemeinek felsorolásával, de hogyan lehet reprezentálni egy végtelen formális nyelvet? Nyilvánvaló, hogy ebben az esetben a szóban forgó nyelvnek egy véges reprezentációját kell megadnunk. Ilyen például a később ismertetett Turing-gép, a különböző véges automata modellek vagy a generatív grammatikák.

1.3.2. Generatív grammatikák

A *generatív grammatikákat* Noam Chomsky amerikai nyelvész vezette be 1956-ban. Célja az volt, hogy a segítségükkel modellezze a természetes nyelvek szintaxisát. A generatív grammatikák egy kezdőszimbólumból kiindulva egy véges szabályhalmaz segítségével szavakat generálnak. Az összes kezdőszimbólumból generálható szó halmaza a grammatika által generált nyelv. A generált nyelv bonyolultsága nagyban függ a grammatika által használt szabályok alakjától.

A szabályokra alkalmazott megszorítások alapján Chomsky megalkotta a generatív grammatikák egy hierarchiáját, amit ma a tiszteletére Chomsky hierarchiának nevezünk. Mint később kiderült, a különböző típusú grammatikák közül az egyik, az úgynevezett 2-es típusú vagy környezetfüggetlen grammatikák, alkalmasak a programozási nyelvek szintaxisának megadására is (lásd a

Backus–Naur-formát a keretes megjegyzésben).

1.9. Definíció

Generatív grammatikának (vagy röviden *grammatikának*) nevezünk egy olyan $G = (V, \Sigma, R, S)$ rendszert, ahol

- V az úgynevezett *nemterminálisok* ábécéje,
- Σ a *terminálisok* ábécéje (feltesszük, hogy $V \cap \Sigma = \emptyset$),
- S egy kitüntetett szimbólum V-ből, amit *kezdőszimbólumnak* nevezünk,
- R pedig $u \to v$ alakú *átírási szabályok* (röviden *szabályok*) véges halmaza, ahol $u, v \in (V \cup \Sigma)^*$ és u-ban van legalább egy nemterminális.

Ha egy grammatikának az α bal oldalú szabályai rendre $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$ valamely $n \geq 2$-re, akkor ezt gyakran így fogjuk jelölni: $\alpha \to \beta_1 \mid \ldots \mid \beta_n$.

A továbbiakban megnézzük, hogyan lehet a grammatikákat szavak generálására felhasználni. Ha ezt tudjuk, akkor definiálni tudjuk azt is, hogy mit értünk egy grammatika által generált nyelven.

Legyen $G = (V, \Sigma, R, S)$ grammatika és $u, v \in (V \cup \Sigma)^*$. Azt mondjuk, hogy a v *egy lépésben levezethető* u*-ból* a G-ben (jele $u \Rightarrow_G v$), ha u és v felírható $u = \alpha\gamma\beta$ és $v = \alpha\gamma'\beta$ alakban ($\alpha, \beta, \gamma, \gamma' \in (V \cup \Sigma)^*$) úgy, hogy R-ben szerepel a $\gamma \to \gamma'$ szabály. Az így kapott relációt a G által meghatározott *közvetlen levezetési relációnak* nevezzük. Amikor egyértelmű, hogy milyen grammatikáról van szó, akkor $\Rightarrow_G$-ben a G indexet általában elhagyjuk.

A Backus–Naur-forma

A BNF-ként is rövidített metanyelvet John Backus hozta létre az ALGOL nyelvtanának leírására. 1959-ben, a Párizsban megtartott „World Computer Congress” elnevezésű konferencián mutatta be. Később Peter Naur egyszerűsítette a BNF eredeti formalizmusát, ezért Donald Knuth javaslatára az ő neve is bekerült a normálforma elnevezésébe.
Egy példa BNF-ben megadott szabályra:

$$< \text{integer} > ::= < \text{digit} > \mid < \text{integer} >< \text{digit} >,$$

ahol a $<$ integer $>$ és $<$ digit $>$ felelnek meg a nemterminálisoknak, $::=$ pedig a $\to$ jelnek (lásd az 1.9. definíciót).

A G által meghatározott *levezetési reláció* (jele $\Rightarrow_G^*$) a következőképpen adódik (a G indexet itt is elhagyhatjuk, amennyiben ez nem okoz félreértést): $u \Rightarrow^* v$ pontosan akkor teljesül, ha u megkapható v-ből a közvetlen levezetési reláció

véges sok (azaz akár nulla) alkalmazásával. Formálisan, $u \Rightarrow^* v$, ha léteznek olyan $n \geq 0$ és $w_0, w_1, \ldots, w_n \in (V \cup \Sigma)^*$ szavak, hogy $u = w_0$, minden $0 \leq i \leq n-1$-re $w_i \Rightarrow w_{i+1}$ és $w_n = v$. Megjegyezzük, hogy $\Rightarrow^*$ a $\Rightarrow$ reláció reflexív, tranzitív lezártja.

A G *által generált nyelv* (jele $L(G)$) azon Σ^*-beli szavak halmaza, melyek megkaphatók (levezethetők) a kezdőszimbólumból a levezetési reláció alkalmazásával: $L(G) = \{u \in \Sigma^* \mid S \Rightarrow^* u\}$.

Egy generatív grammatika szabályaira adott megszorítások alapján a Chomsky-hierarchia alábbi osztályait különböztetjük meg.

1.10. Definíció

Legyen $G = (V, \Sigma, R, S)$ grammatika.

- G *0-típusú* vagy *általános grammatika*, ha a szabályaira nincsen semmilyen megkötés.
- G *1-típusú* vagy *környezetfüggő grammatika*, ha minden szabálya $\alpha A \beta \to \alpha \gamma \beta$ alakú, ahol $A \in V$, $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$, $\gamma \neq \varepsilon$ (kivéve az $S \to \varepsilon$ szabályt, de ha ezt a szabályt G tartalmazza, akkor S nem fordulhat elő szabály jobb oldalán).
- G *2-típusú* vagy *környezetfüggetlen (röviden CF) grammatika*, ha minden szabálya $A \to v$ alakú, ahol $A \in V$ és $v \in (V \cup \Sigma)^*$.
- G *3-típusú* vagy *reguláris grammatika*, ha minden szabálya $A \to vB$ vagy $A \to v$ alakú, ahol $A \in V$ és $u \in \Sigma^*$.

Egy L nyelvet i-típusúnak nevezünk, ha van olyan G i-típusú grammatika, hogy $L(G) = L$. A jegyzet további részében csak CF grammatikákat fogunk vizsgálni. A 2-típusú nyelveket *környezetfüggetlennek* (röviden CF-nek) is nevezzük.

1.11. Példa

Az alábbiakban mutatunk néhány példát környezetfüggetlen grammatikákra. A grammatikák által generált nyelvekkel már találkoztunk az 1.7. példában.

1. Legyen $G_1 = (V, \Sigma, R, S)$, ahol $V = \{S\}$, $\Sigma = \{0, 1\}$ és $R = \{S \to 01S, S \to \varepsilon\}$ (vegyük észre, hogy G_1 3-as típusú is). G_1-ben elvégezhető például a következő levezetés: $S \Rightarrow 01S \Rightarrow 0101S \Rightarrow 0101$, vagyis $S \Rightarrow^* 0101$. Tehát $0101 \in L(G_1)$, továbbá a G_1 által generált nyelv: $L(G_1) = \{(01)^n \mid n \geq 0\}$.
2. Legyen $G_2 = (V, \Sigma, R, S)$, ahol $V = \{S\}$, $\Sigma = \{0, 1\}$ és $R = \{S \to$

$0S1, S \to \varepsilon\}$. Ekkor igaz, hogy $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0011$, vagyis $S \Rightarrow^* 0011$. Tehát $0011 \in L(G_2)$ és $L(G_2) = \{0^n 1^n \mid n \geq 0\}$.

1.3.2.1. Egyértelmű környezetfüggetlen nyelvtanok

Tekintsük az alábbi, egyszerű aritmetikai kifejezéseket generáló CF grammatikát.

$$G_{Ar} = (\{E\}, \{(,), *, +, x, y, z\}, R, E),$$

ahol R a következő szabályokat tartalmazza:

$$E \to E + E \mid E * E \mid (E) \mid x \mid y \mid z.$$

Ebben a grammatikában le lehet vezetni például az $x + y * z$ szót a következőképpen:

$$E \Rightarrow E + E \Rightarrow x + E \Rightarrow x + E * E \Rightarrow x + y * E \Rightarrow x + y * z.$$

A fenti levezetés olyan, hogy mindig az aktuális mondatforma legelső nemterminálisát írjuk át. Az ilyen levezetéseket nevezzük *baloldali levezetéseknek*. Felmerülhet a kérdés, hogy az $x + y * z$ szónak van-e a fentitől különböző baloldali levezetése. A válasz igen, tekintsük például a $E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow x + E * E \Rightarrow x + y * E \Rightarrow x + y * z$ levezetést. Ez azért jelent problémát, mert az első levezetés szerint $y * z$, a második szerint viszont $x + y$ alkot részkifejezést, ezért más-más jelentést fogunk a későbbiekben ugyanahhoz az $x + y * z$ kifejezéshez társítani.

Ez a probléma nem merül fel egy olyan G grammatika esetén, melyben minden $u \in L(G)$ szónak pontosan egy baloldali levezetése van. Az ezzel a tulajdonsággal rendelkező grammatikákat nevezzük *egyértelmű grammatikáknak*. A fenti G_{Ar} tehát nem egyértelmű grammatika. Egy nyelvet *egyértelműnek* nevezünk, ha van olyan egyértelmű grammatika ami L-et generálja. Bár a G_{Ar} nyelvtan nem egyértelmű, az általa generált L_{Ar} nyelv az. Egy az L_{Ar}-t generáló egyértelmű grammatika például az, melynek szabályai a következők:

$E \to E + T \mid T$, $T \to T * N \mid N$, $N \to (E) \mid x \mid y \mid z$.

Vajon minden CF nyelv egyértelmű?

Érdekes kérdés, hogy vajon minden CF nyelvhez megadható-e őt generáló egyértelmű nyelvtan. Sajnos a válasz az, hogy nem. Az $L = \{a^n b^n c^m d^m \mid n, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n, m \geq 1\}$ nyelv például nem egyértelmű CF nyelv. Ennek bizonyítása megtalálható a [4] könyvben is, ezért itt csak a bizonyítás alapötletét közöljük. Legyen G egy L-et generáló CF grammatika. Viszonylag könnyen meggondolható, hogy G-ben kell hogy legyen két diszjunkt szabályhalmaz,

Itt az $x + y * z$ szónak csak egy baloldali levezetése van, ez pedig a következő:

$$\begin{aligned} E \Rightarrow & E + T \Rightarrow N + T \Rightarrow x + T \Rightarrow \\ \Rightarrow & x + T * N \Rightarrow x + N * N \Rightarrow \\ \Rightarrow & x + y * N \Rightarrow x + y * z. \end{aligned}$$

Mivel a CF grammatikák felhasználhatók programozási nyelvek szintaxisának megadására, és várhatóan egy egyértelmű CF grammatikákhoz hatékonyabb elemző algoritmus konstruálható, mint egy nem egyértelműhöz, érdekes kérdés az, hogy vajon el lehet-e dönteni egy tetszőleges CF grammatikáról, hogy egyértelmű-e. Később látni fogjuk, hogy sajnálatos módon ez egy olyan probléma, amit nem lehet algoritmikusan eldönteni.

melyek lényegében rendre az $L_1 = \{a^n b^n c^m d^m \mid n, m \geq 1\}$ és az $L_2 = \{a^n b^m c^m d^n \mid n, m \geq 1\}$ nyelveket generálják. Ha ezek a szabályhalmazok nem lennének diszjunktak, akkor olyan szavak is levezethetők lennének, melyek sem az L_1-be, sem az L_2-be nem tartoznak. Kell hogy legyenek tehát az L-ben olyan $a^n b^n c^n d^n$ $(n \geq 1)$ alakú szavak, melyeket mindkét szabályhalmaz segítségével le lehet vezetni, vagyis G nem lehet egyértelmű.

2. fejezet

A kiszámíthatóságelmélet alapjai

Először áttekintjük, hogy milyen események vezettek odáig, hogy kialakult az algoritmus matematikailag is precíz fogalma és bizonyossá vált, léteznek algoritmikusan eldönthetetlen problémák.

2.1. A kiszámíthatóságelmélet rövid története

1900-ban, a századforduló alkalmából, David Hilbert német matematikus 23 addig megválaszolatlan kérdést intézett a kor matematikusaihoz. Mint ahogy az később kiderült, ezek közül jó néhány nagy hatással volt a huszadik századi matematika és különösen a kiszámíthatóságelmélet fejlődésére.

Akkoriban az algoritmus fogalmának csak egy intuitív definíciója létezett:

> *Az algoritmus utasítások egy jól definiált, véges sorozata, melyeket végrehajtva megoldható egy adott feladat (probléma).*

Hilbert 10-ik problémája

Adott egy p egész együtthatós többváltozós polinom. A feladat annak megválaszolása, hogy lehet-e p változóiba olyan egész számokat helyettesíteni, hogy p értéke 0 legyen. Ha például $p = 3xy - 2x^2 + 2z$, akkor x helyébe 2-t, y és z helyébe pedig 1-et helyettesítve p értéke 0 lesz. Tehát ezen konkrét polinom esetében *igen* a válasz a kérdésre. Hilbert olyan algoritmust keresett, ami tetszőleges polinom esetén helyes választ ad.

Hilbert úgy gondolta, hogy nincsenek megoldhatatlan problémák és meg volt

győződve róla, hogy például a 10-es probléma is megoldható (lásd a kertes megjegyzésben). Ez azt jelenti, hogy hitt egy olyan algoritmus létezésében, ami a probléma tetszőleges bemenete esetén helyes választ ad. Az, hogy létezik-e ilyen algoritmus nyitott kérdés maradt 1970-ig. Ekkor Yuri Matiyasevich, a témában született korábbi eredményeket felhasználva megmutatta, hogy a 10-ik probléma algoritmikusan eldönthetetlen. Ehhez azonban nem volt elég az algoritmus intuitív fogalma, annak matematikailag is precíz definíciójára volt szükség.

Hilbert az 1920-as években meghirdette nagyra törő programját, melynek során formalizálni és axiomatizálni szerette volna a matematika összes elméletét egy végesen reprezentálható, teljes és konzisztens axiómarendszerrel. A program része volt egy olyan algoritmus megadása, nevezzük Mindent Megoldó Algoritmusnak (röviden MMA-nak), mely a matematika összes állításáról képes eldönteni, hogy az igaz vagy hamis.

Azt, hogy Hilbert programja alapvetően megvalósíthatatlan, Kurt Gödel mutatta meg azzal, hogy az úgynevezett első nemteljességi tételével bebizonyította, hogy minden olyan effektíven kiszámítható elmélet, ami tartalmazza a természetes számok elméletét, nem lehet egyszerre helyes és teljes. Az MMA létezésének cáfolásához viszont várni kellett az algoritmus pontos definíciójának megjelenéséig.

1934-ben Gödel definiálta az úgynevezett *rekurzív függvényeket*. Az 1930-as években Alonso Church tanítványaival megalkotta a *λ-kalkulust*, egy formális rendszert, ami a függvény fogalmán és a függvényeknek a változók értékeire való alkalmazásán alapszik. 1936-ban Alan Turing definiálta a később róla elnevezett *Turing-gépeket*. Mindannyian azt gondolták, hogy az általuk megalkotott formalizmus reprezentálja a legáltalánosabban az algoritmus intuitív fogalmát. Később kiderült, hogy ezekkel az eszközökkel ugyanazon függvényeket lehet kiszámítani. Ezután számos más formális rendszert is definiáltak, de mindről kiderült, hogy a fenti eszközökkel azonos számítási erővel bírnak. Ezek az eredmények támasztják alá a ma már széles körben elfogadott *Church-Turing tézist:*

> *A kiszámíthatóság különböző matematikai modelljei mind az effektíven kiszámítható függvények osztályát definiálják.*

Visszatérve az MMA létezésének cáfolására, Church ezt úgy bizonyította, hogy megmutatta, nincs olyan effektíven kiszámítható módszer, ami két λ-kalkulusbeli kifejezésről eldönti, hogy azok ekvivalensek-e. Turing a következőképpen gondolkodott. Először megmutatta, hogy nincs olyan módszer, ami eldöntené a Turing-gépek úgynevezett *megállási problémáját*. Ezek után pedig megfogalmazta a problémát matematikai állításként. Ebből már következett, hogy nem létezhet a Mindent Megoldó Algoritmus.

2.2. Problémák mint formális nyelvek

Kiszámítási problémának nevezünk egy olyan, a matematika nyelvén megfogalmazott kérdést, amire egy algoritmussal szeretnénk megadni a választ. A gyakorlati élet szinte minden problémájához rendelhető, megfelelő absztrakciót használva, egy kiszámítási probléma (lásd például a *körpakolás problémát* a keretes megjegyzésben).

Egy probléma egy konkrét bemenetét szokás a probléma egy *példányának* is nevezni. A körpakolás probléma egy példánya például az, amikor megadjuk a lefedendő alakzat és a lefedésre használható körök paramétereit.

A körpakolás probléma

Ez a probléma egy ismert matematikai probléma: azt vizsgáljuk, hogyan lehet egy adott alakzat minél nagyobb részét lefedni azonos vagy akár különböző sugarú körökkel. Világos, hogy ez a probléma egy absztrakt leírása a következő, valós életből vett problémának. Tegyük fel, hogy van több, azonos magasságú, de különböző méretű hordónk, melyeket el szeretnénk szállítani valahova. Adódik a kérdés: hogyan helyezzük el a teherautónkon a hordóinkat úgy, hogy minél nagyobb legyen a hordók együttes űrtartalma.

Azt mondjuk, hogy egy P_1 probléma a P_2 egy *speciális esete*, ha P_1 minden példánya a P_2 egy példánya is egyben. A körpakolás probléma egy speciális esete például az a probléma, amikor a lefedendő alakzat csak téglalap lehet.

Egy P kiszámítási probléma reprezentálható egy $f_P : A \to B$ függvénnyel. Az A halmaz tartalmazza a probléma egyes bemeneteit, jellemzően egy megfelelő ábécé feletti szavakkal kódolva, míg a B halmaz tartalmazza a bemenetekre adott válaszokat, szintén valamely alkalmas ábécé feletti szavakkal kódolva.

Egy $f : A \to B$ függvényt *kiszámíthatónak* nevezünk, ha minden $x \in A$ elemre az $f(x) \in B$ érték kiszámítható valamilyen algoritmus modellel. Azt mondjuk, hogy egy P probléma *megoldható*, ha f_P kiszámítható.

Speciális kiszámítási problémák az *eldöntési problémák*. Ilyenkor a problémával kapcsolatos kérdés egy eldöntendő kérdés, tehát a probléma egy példányára a válasz *igen* vagy *nem* lesz. Értelemszerűen, ha P egy eldöntési probléma, akkor az f_P értékkészlete egy két elemű halmaz: {`igen, nem`}, $\{1, 0\}$, stb. A megoldható eldöntési problémákat *eldönthető problémáknak* nevezzük.

Az egyik legismertebb eldöntési probléma az úgynevezett Sat probléma, amit a következőképpen definiálunk. Adott egy φ ítéletkalkulusbeli konjunktív normálforma. A kérdés az, hogy kielégíthető-e φ. Tehát a problémára a válasz *igen*, ha φ kielégíthető, és *nem* egyébként.

A Sat probléma eldönthető, hiszen könnyen adható olyan algoritmus, ami eldönti azt, hogy egy φ formula kielégíthető-e. Ez az algoritmus nem csinál mást, mint a φ-ben szereplő változóknak logikai értéket ad az összes lehetséges mó-

don, majd rendre kiértékeli a formulát. Ez az algoritmus a bemenet méretében exponenciális időigényű, és nem ismert ennél nagyságrendileg jobb algoritmus a probléma megoldására.

Egy eldöntési probléma egy bemenetét szokás *igen példánynak* vagy *pozitív bemenetnek* (rendre, *nem példánynak* vagy *negatív bemenetnek*) nevezni, ha a bemenet olyan, hogy a válasz rá *igen* (rendre, *nem*).

Egy eldöntési probléma tekinthető úgy is mint egy formális nyelv. A probléma példányait kódoljuk egy megfelelő ábécé feletti szavakkal. Ezek után magát a problémát azonosítjuk azzal a formális nyelvvel, mely azokat a szavakat tartalmazza, melyek a probléma pozitív bemeneteit kódolják.

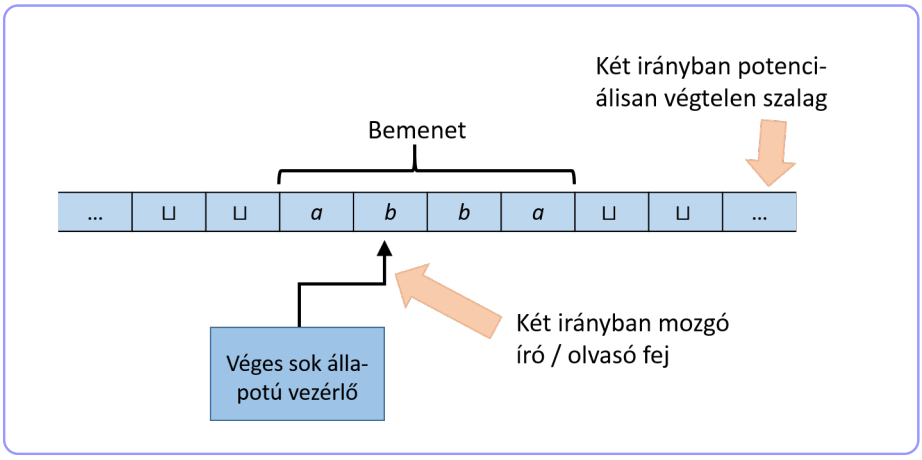
Az így kapott formális nyelvet általában ugyanúgy nevezzük, mint magát a problémát. Tehát például a SAT jelentheti a fent definiált eldöntési problémát és azt a formális nyelvet is, amely szavai a kielégíthető ítéletkalkulusbeli KNF-eket kódolják.

A továbbiakban egy tetszőleges D objektumra $\langle D \rangle$ jelöli a D egy megfelelő ábécé feletti szóval való *kódolását*.

A jegyzetben jellemzően eldöntési problémákkal foglalkozunk majd. Ez nem jelenti azonban az általánosság megszorítását. Tekintsünk ugyanis egy P kiszámítási problémát, és legyen $f_P : A \to B$ a P által meghatározott függvény. Ekkor P-hez megadható egy P' eldöntési probléma úgy, hogy P' pontosan akkor dönthető el, ha P kiszámítható. Állítsuk párba ugyanis minden $a \in A$ elemre az a-t és az $f_P(a)$ értéket, és kódoljuk az így kapott párokat egy-egy szóval. Ezek után legyen P' az így kapott szavakból képzett formális nyelv. Nyilvánvaló, hogy ha minden $a \in A$ és $b \in B$ elemre az $(a, b) \in P'$ tartalmazás eldönthető, vagyis P' eldönthető, akkor P kiszámítható. Továbbá az is igaz, hogy ha P kiszámítható, akkor P' eldönthető.

Hány probléma van?

Mivel az eldöntési problémák megfeleltethetők formális nyelveknek, ugyanannyi eldöntési probléma van, mint amennyi formális nyelv. Formális nyelvből legalább megszámlálhatatlanul sok van a következők miatt. A $\{a\}^*$ számossága megegyezik az $\mathbb{N}$ számosságával, és $\{a\}^*$-feletti formális nyelv annyi van amennyi a $\mathcal{P}(\mathbb{N})$ számossága, azaz megszámlálhatatlanul végtelen sok.
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.



2.1. ábra. Egy Turing-gép

2.3. Turing-gépek

Amint arról a bevezetésben szó volt, a Turing-gép az egyik legáltalánosabb algoritmusmodell. Ebben a részben megvizsgáljuk a Turing-gépek különböző változatait. Ezután megmutatjuk, hogy van olyan formális nyelv, melyről egyetlen Turing-gép sem képes eldönteni teljes biztonsággal, hogy vajon egy szó eleme-e ennek a nyelvnek vagy sem.

A *Turing-gép* olyan véges sok állapottal rendelkező eszköz, ami egy *két irányban végtelen szalagon* dolgozik. A szalag *cellákra* van osztva, tulajdonképpen ez a gép (korlátlan) memóriája. Kezdetben a szalagon csak a bemenő szó van, minden cellán egy betű. A szalag többi cellája egy úgynevezett *üres* ($\sqcup$) szimbólummal van feltöltve. Egy ilyen gép sematikus rajza a 2.1. ábrán látható.

Kezdetben a gép úgynevezett *író-olvasó feje* a bemenő szó első betűjén áll és a gép a kezdőállapotában van.

A gép az író-olvasó fejet tetszőlegesen képes mozgatni a szalagon. Képes továbbá a fej pozíciójában a szalag tartalmát kiolvasni és átírni. A gépnek van két kitüntetett állapota, a q_i és a q_n állapotok. Ha ezekbe az állapotokba kerül, akkor rendre elfogadja illetve elutasítja a bemenő szót. Formálisan a Turing-gépet a következő módon definiáljuk.

2.1. Definíció

A *Turing-gép* olyan $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ rendszer, ahol

- Q az *állapotok* véges, nemüres halmaza,

- $q_0, q_i, q_n \in Q$, q_0 a *kezdő-*, q_i az *elfogadó* és q_n az *elutasító állapot*,
- Σ és Γ rendre a *bemenő jelek* és a *szalagszimbólumok ábécéje* úgy, hogy $\Sigma \subseteq \Gamma$ és $\Gamma - \Sigma$ tartalmaz egy speciális $\sqcup$ szimbólumot,
- $\delta : (Q - \{q_i, q_n\}) \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$ az *átmenetfüggvény*.

Egy M Turing-gép működésének fázisait konfigurációkkal írhatjuk le. Az M *konfigurációja* egy olyan uqv szó, ahol $q \in Q$ és $u, v \in \Gamma^*$ úgy, hogy $v \neq \varepsilon$. Ez a konfiguráció az M azon állapotát tükrözi amikor a szalag tartalma uv (uv előtt és után a szalagon már csak $\sqcup$ van), a gép a q állapotban van, és az író-olvasó fej a v első betűjére mutat. A gép *kezdőkonfigurációja* egy olyan $q_0u\sqcup$ szó, ahol u csak Σ-beli betűket tartalmaz. Az M összes konfigurációinak a halmazát $\mathcal{C}_M$-mel jelöljük. Az M *konfiguráció-átmenete* egy olyan $\vdash \subseteq \mathcal{C}_M \times \mathcal{C}_M$ reláció, amit a következőképpen definiálunk. Legyen $uqav$ egy konfiguráció, ahol $a \in \Gamma$ és $u, v \in \Gamma^*$. A következő három esetet különböztetjük meg.

1. Ha $\delta(q, a) = (r, b, S)$, akkor $uqav \vdash urbv$.
2. Ha $\delta(q, a) = (r, b, R)$, akkor $uqav \vdash ubrv'$, ahol ha $v \neq \varepsilon$, akkor $v' = v$, és $v' = \sqcup$ egyébként.
3. Ha $\delta(q, a) = (r, b, L)$, akkor $uqav \vdash u'rcbv$ ahol ha $u \neq \varepsilon$, akkor $u'c = u$ ($u' \in \Gamma^*$ és $c \in \Gamma$), és $u' = \varepsilon$ valamint $c = \sqcup$ egyébként.

Azt mondjuk, hogy M véges sok lépésben eljut a C konfigurációból a C' konfigurációba (jele $C \vdash^* C'$), ha van olyan $n \geq 0$ és $C_1, \ldots, C_n$ konfigurációsorozat, hogy $C_1 = C$, $C' = C_n$ és minden $i \in [n]$-re, $C_i \vdash C_{i+1}$. Tehát a $\vdash^*$ reláció a $\vdash$ reláció reflexív, tranzitív lezártja.

Ha $q \in \{q_i, q_n\}$, akkor azt mondjuk, hogy az uqv konfiguráció *megállási konfiguráció*. Továbbá $q = q_i$ esetében *elfogadó*, míg $q = q_n$ esetében *elutasító* konfigurációról beszélünk.

Az M *által felismert nyelv* (amit $L(M)$-mel jelölünk) azoknak az $u \in \Sigma^*$ szavaknak a halmaza, melyekre igaz, hogy $q_0u\sqcup \vdash^* xq_iy$ valamely $x, y \in \Gamma^*, y \neq \varepsilon$ szavakra. Érdemes megjegyezni, hogy ha M nem fogad el egy u bemenetet (azaz $u \notin L(M)$), akkor szükségszerűen vagy elutasítja azt (véges sok lépés után elutasító konfigurációba lép), vagy nem áll meg rajta.

2.2. Példa

Tekintsük az $L = \{uu^{-1} \mid u \in \{a, b\}^*\}$ nyelvet. Az L felismerhető egy olyan M Turing-géppel, mely adott u bemeneten a következő algoritmus szerint működik.

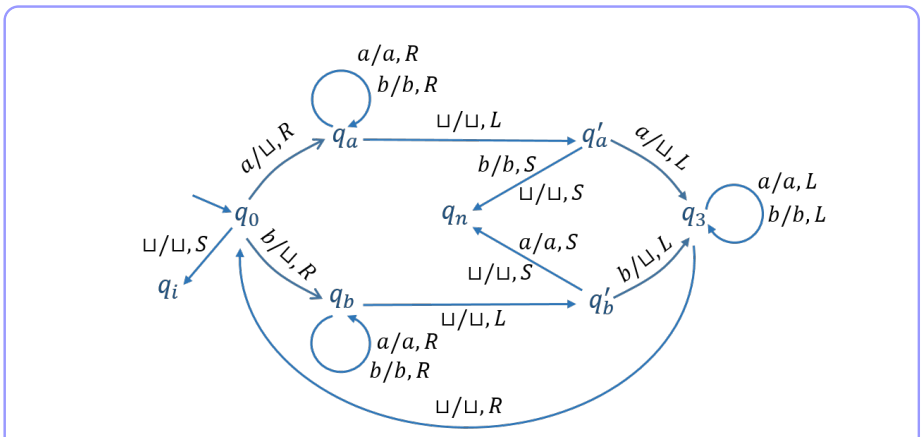
1. q_0-ban elolvassa a szó első betűjét.
2. Ha az első betű $\sqcup$, akkor megáll q_i-ben.
3. Ha az első betű a, akkor letörli és q_a-ba lép (az állapotában megjegyzi mit törölt le);
4. Ha az első betű b, akkor letörli és q_b-ba lép (az állapotában megjegyzi mit törölt le);
5. q_a-ban illetve q_b-ben elmegy a szó végére.
6. Megnézi, hogy a szó végi betű megegyezik-e az állapotban megjegyzett (azaz a szó elejéről letörölt) betűvel.
7. Ha nem, akkor megáll q_n-ben.
8. Ha igen, akkor
   (a) letörli a szalagon lévő szó utolsó betűjét és q_3-ba lép;
   (b) q_3-ban visszamegy a szó elejére; ha azt elérte q_0-ba lép, és folytatja az 1. ponttal.

A 2.2. ábrán egy a fenti algoritmust megvalósító M Turing-gép látható. A gráf csúcsai a gép állapotai, a gép kezdőállapotát, azaz q_0-t egy kiindulási csúcs nélküli él jelzi, $\Sigma = \{a, b\}$ és $\Gamma = \{a, b, \sqcup\}$. Az átmenetfüggvényt a következő módon lehet kiolvasni a gráfból. Legyen q és p a gép két állapota, a, b szalagszimbólumok, D pedig egy $\{L, R, S\}$-beli irány. Ekkor az ábrán egy q-ból p-be vezető él „$a/b, D$" címkéje azt jelenti, hogy $\delta(q, a) = (p, b, D)$.
M számítása az *abba* szón végigkövethető a 2.2. ábrán, ahol néhány konfigurációátmenetet összevontunk (azt, hogy mennyit a $\vdash$ felső indexébe írt szám jelöli). Tehát M, helyesen, elfogadja az *abba* szót.

Most a Turing-gép segítségével definiáljuk a Turing-felismerhető és az eldönthető nyelveket.

2.3. Definíció

Egy $L \in \Sigma^*$ nyelv *Turing-felismerhető*, ha $L = L(M)$ valamely M Turing-gépre. Továbbá egy $L \subseteq \Sigma^*$ nyelv *eldönthető*, ha létezik olyan M Turing-gép, ami felismeri L-et és minden bemeneten megállási konfigurációba jut. A Turing-felismerhető nyelveket szokás *rekurzívan felsorolhatónak*, az eldönthető nyelveket pedig *rekurzívnak* is nevezni. Az összes



2.2. ábra. A 2.2. példabeli nyelvet felismerő egyszalagos Turing-gép

Turing-felismerhető nyelv osztályát **RE**-vel az összes rekurzív nyelvét pedig **R**-rel fogjuk jelölni.

Megjegyezzük, hogy a fenti példában látható Turing-gép nem csak felismeri, hanem el is dönti az L nyelvet.

Felmerülhet a kérdés, hogy mennyi lépesre van szüksége egy Turing-gépnek a bemenet hosszának a függvényében ahhoz, hogy megálljon a bemeneten (ha megáll egyáltalán). Ezen kérdés vizsgálatához definiálnunk kell a Turing-gépek időigényét.

Tekintsünk egy $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ Turing-gépet és annak egy $u \in \Sigma^*$ bemenő szavát. Azt mondjuk, hogy M *időigénye az u szón n* ($n \in \mathbb{N}$), ha M a $q_0u\sqcup$ kezdőkonfigurációból n lépésben egy megállási konfigurációba jut. Ha nincs ilyen szám, akkor M időigénye az u-n végtelen.

Legyen $f : \mathbb{N} \to \mathbb{N}$ egy függvény. Azt mondjuk, hogy M *időigénye* $f(n)$ (vagy azt, hogy M egy $f(n)$ *időkorlátos gép*), ha minden $u \in \Sigma^*$ input szóra, M időigénye az u szón legfeljebb $f(l(u))$.

Legtöbbször csak arra vagyunk kíváncsiak, hogy mi az általunk vizsgált Turing-gép időigényének nagyságrendje (azaz vajon lineáris, négyzetes, polinomiális

vagy exponenciális-e). Ehhez használhatjuk a jól ismert $\mathcal{O}$ jelölést. Legyen f és g két $\mathbb{N}$-ből a nem negatív valós számok halmazába képező függvény. Azt mondjuk, hogy az f *legfeljebb olyan gyorsan nő mint a* g (jele: $f(n) = \mathcal{O}(g(n))$) ha a következő teljesül. Vannak olyan $n_0 \in \mathbb{N}$ és c pozitív valós számok, hogy minden $n \geq n_0$ esetén $f(n) \leq c \cdot g(n)$. Könnyen látható például, hogy $5n^2 + 5n + 5 = \mathcal{O}(n^2)$.

2.4. Példa

Tekintsük újra a 2.2. példában látható L nyelvet, és az azt eldöntő Turing-gépet. A gép időigénye egy uu^{-1} szón, melynek hossza n, a következőképpen alakul.

- A szó elején és végén lévő betűk összehasonlítása legfeljebb $2n + 2$ lépés.
- A fenti összehasonlítást legfeljebb $\lfloor \frac{n}{2} \rfloor$ esetben kell végrehajtani.

Tehát a gép számítása az uu^{-1} szón legfeljebb $n^2 + n$ lépésből áll. Ha a gép a szalagján olyan n-hosszú u szót kap bemenetként, amely nem eleme L-nek, akkor belátható, hogy u elutasításra kerül legfeljebb ugyanennyi lépésben. Tehát a példában szereplő Turing-gép egy $\mathcal{O}(n^2)$ időkorlátos gép.

2.3.1. Különböző Turing-gép változatok

Ebben a fejezetben megvizsgálunk néhány, az eddigitől kissé eltérő Turing-gép változatot. Ezek, mint látni fogjuk, mind ekvivalensek lesznek az eredeti Turing-gép modellünkkel.

2.3.1.1. Többszalagos Turing-gépek

A többszalagos Turing-gépek, értelemszerűen, egynél több szalaggal is rendelkezhetnek. Mindegyik szalaghoz tartozik egy-egy író-olvasó fej, melyek egymástól függetlenül képesek mozogni a szalagon.

2.5. Definíció

Legyen $k \geq 1$. A *k-szalagos Turing-gép* olyan $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ rendszer, ahol a komponensek a δ kivételével megegyeznek az egyszalagos Turing-gép komponenseivel, δ pedig a következőképpen adódik: $\delta : (Q - \{q_i, q_n\}) \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$. Legyenek

$q, p \in Q$, $a_1, \ldots, a_k, b_1, \ldots, b_k \in \Gamma$ és $D_1, \ldots, D_k \in \{L, R, S\}$. Ha $\delta(q, a_1, \ldots, a_k) = (p, b_1, \ldots, b_k, D_1, \ldots, D_k)$, akkor a gép a q állapotból, ha a szalagjain rendre az $a_1, \ldots, a_k$ betűket olvassa, át tud menni a p állapotba, miközben az $a_1, \ldots, a_k$ betűket átírja a $b_1, \ldots, b_k$ betűkre, és a szalagokon a fejeket a $D_1, \ldots, D_k$ irányokba mozgatja.

A többszalagos Turing-gép konfigurációja, a konfigurációátmenet valamint a felismert és az elöntött nyelv definíciója az egyszalagos eset értelemszerű általánosítása. A többszalagos Turing-gép időigényét is az egyszalagoshoz hasonlóan definiáljuk.

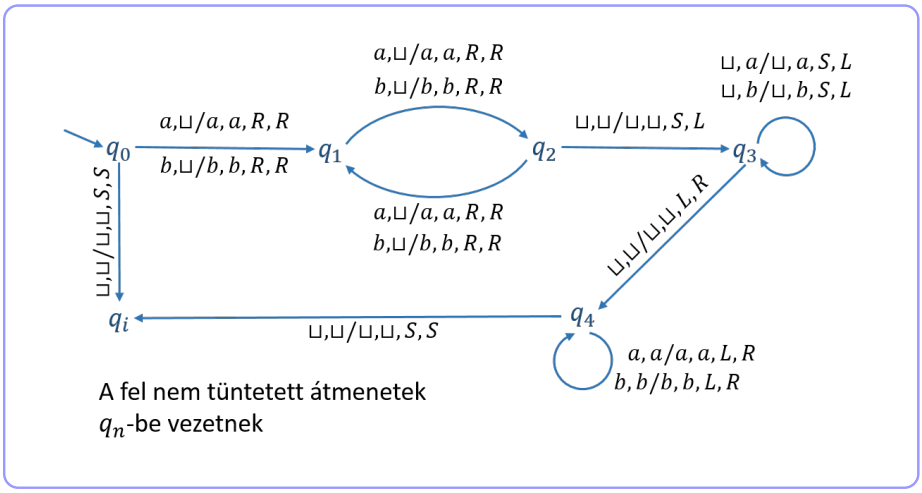
A továbbiakban egy L nyelvet $f(n)$ *időben eldönthetőnek* nevezünk, ha eldönthető egy $f(n)$ időkorlátos (akár többszalagos) Turing-géppel.

2.6. Példa

Tekintsük újra az $L = \{uu^{-1} \mid u \in \{a, b\}^*\}$ nyelvet. A 2.3. ábrán látható kétszalagos Turing-gép az alábbi algoritmust alkalmazva dönti el L-et.

1. Ha üres a bemenet, akkor q_i-be lép (helyesen, hisz az üres szó is eleme a nyelvnek).
2. Ha a bemenet nem üres, akkor q_1 és q_2 állapotok között lépkedve átmásolja a bemenetet a második szalagra.
3. Ha a másolás q_1-ben ér véget, akkor q_n-be lép (ezt helyesen teszi, hisz ekkor a szó páratlan hosszú és nem eleme a nyelvnek).
4. Ha a másolás q_2-ben ér véget, akkor a második szalagon visszamegy a szó elejére (q_3 állapot); ha elérte a szó elejét, akkor q_4-be lép.
5. Az első szalagon balra, a másodikon pedig jobbra lépkedve összehasonlítja a szalagon lévő betűket.
6. Ha egyszerre látja a szó elejét az első és a szó végét a második szalagon, akkor q_i-be, egyébként q_n-be lép.

A gép δ átmenetfüggvénye a következő módon olvasható ki az ábrából. Legyen q és p a gép két állapota, a_1, a_2, b_1 és b_2 szalagszimbólumok, D_1 és D_2 pedig $\{L, R, S\}$-beli irányok. Egy q-ból p-be vezető él „$a_1, a_2/b_1, b_2, D_1, D_2$” címkéje azt jelenti, hogy $\delta(q, a_1, a_2) = (p, b_1, b_2, D_1, D_2)$. A gép be nem rajzolt átmenetei itt is a q_n állapotba vezetnek.
A gép időigényét egy n hosszú szón könnyű kiszámolni, az legfeljebb $3n + 3$ lesz, tehát L egy $\mathcal{O}(n)$, azaz lineáris időben eldönthető nyelv.



2.3. ábra. Az 2.2. példabeli nyelvet felismerő kétszalagos Turing-gép

Bár a Turing-gép több szalaggal általában egyszerűbben képes felismerni egyes nyelveket, az általános számítási ereje nem nő, ahogy ezt az alábbi tételben látni fogjuk. Értelemszerűen, két Turing-gépet *ekvivalensnek* nevezünk, ha azok ugyanazt a nyelvet ismerik fel.

2.7. Tétel

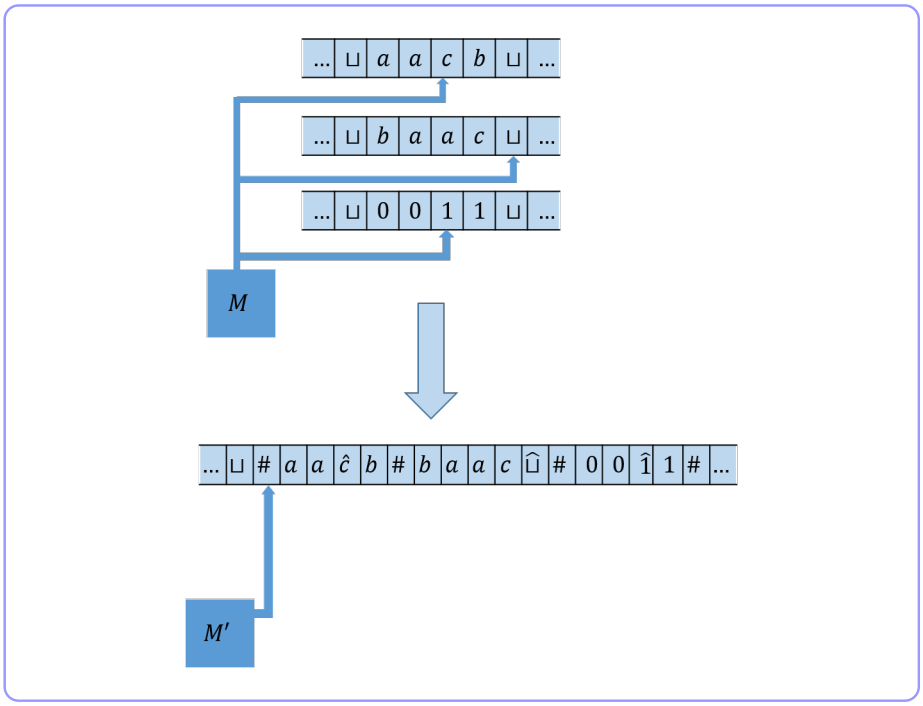
Minden k-szalagos Turing-géphez van vele ekvivalens egyszalagos Turing-gép.

Bizonyítás. Legyen M k-szalagos Turing-gép. Ha $k = 1$ akkor készen vagyunk, tegyük fel tehát, hogy $k \geq 2$. Megadunk egy M' egyszalagos Turing-gépet, ami képes szimulálni M működését. A szimuláció alapötlete a 2.7. ábrán látható.

M' egymás után tárolja a szalagján M szalagjainak a tartalmát. A különböző szalagokat # jellel választja el egymástól. Azt, hogy M szalagjain a fejek mely szimbólumokra mutatnak, M' úgy tartja számon, hogy a kérdéses szimbólumokat megjelöli ˆ jellel. Tehát M' szalagszimbólumai között, az M szalagszimbólumai mellett, ott van még a # szimbólum, valamint M szalagszimbólumainak ˆ-pal megjelölt változatai.

A szimuláció lépései a következők. Legyen $w = a_1 \dots a_n$ az M bemenő szava.

1. M' először előállítja a szalagján M kezdőkonfigurációját, ami így néz ki: $\#\hat{a}_1 a_2 \dots a_n \# \hat{\sqcup} \# \hat{\sqcup} \dots \#$.
2. M egy lépésének szimulálásához M' végigolvassa a szalagját az első #-tól



2.4. ábra. Egy háromszalagos Turing-gépet szimuláló egyszalagos Turing-gép

kezdve az utolsó (azaz a k+1-ik) #-ig. Eközben az állapotában eltárolja a $\hat{}$ jellel megjelölt szimbólumok $\hat{}$ nélküli változatait (M' állapotai úgy vannak definiálva, hogy mindegyik képes tárolni M k darab szalagszimbólumát).

3. M' ezután az állapotában eltárolt adatok és M átmenetfüggvénye alapján (az M átmenetfüggvényét be lehet ágyazni M' átmenetfüggvényébe) végrehajtja a saját szalagján azokat a módosításokat, melyeket M végez a szalagjain.

4. Ha M valamelyik szalagján az utolsó nem $\sqcup$ szimbólum olvasása után jobbra lép, akkor M' a megfelelő #-tól kezdve jobbra mozgatja egy pozícióval a szalagjának tartalmát és a felszabadult helyre beszúr egy $\hat{\sqcup}$ szimbólumot. Hasonlóan jár el M', ha M a legelső nem $\sqcup$ szimbólum olvasása után balra lép.

5. Ha M valamilyen (elfogadó vagy elutasító) megállási konfigurációba kerül, akkor M' is a megfelelő megállási konfigurációba lép. Egyébként M' folytatja M lépéseinek szimulálását a 2. ponttal.

Látható, hogy M' pontosan akkor lép elfogadó állapotba amikor M, tehát $L(M) = L(M')$, vagyis a két Turing-gép ekvivalens. □

2.3.1.2. Turing-gép egy irányban végtelen szalaggal

A Turing-gép definiálható úgy is, hogy egy balra zárt, jobbra végtelenen szalagon dolgozik. Ebben a fejezetben az ilyen Turing-gépet *egyirányú Turing-gépnek* nevezzük (az eredeti modellre pedig *többirányú Turing-gépként* hivatkozunk majd). Egyirányú Turing-gépnél a Turing-gép definícióját úgy módosítjuk, hogy az író olvasó fej nem „eshet le" a szalag bal oldalán (az ilyen gépek pontos definíciója megtalálható például a [8] könyvben). Természetesen az egyirányú Turing-gépnek is létezik többszalagos verziója.

Az világos, hogy egy egyirányú Turing-gép szimulálható egy többirányúval. Egyszerűen megjelöljük a többirányú Turing-gép szalagjain a fejek kezdőpozícióját egy speciális csak erre használt szimbólummal és ügyelünk rá, hogy a Turing-gépünk ne mozdítsa a fejet ezen pozíciótól balra. A fordított irányú szimuláció egy kicsit bonyolultabb.

2.8. Tétel

Minden többirányú Turing-géphez megadható vele ekvivalens egyirányú Turing-gép.

Bizonyítás (vázlat). Legyen M többirányú Turing-gép. Megkonstruálunk egy M-mel ekvivalens, kétszalagos M' egyirányú Turing-gépet. Ezután M'-

höz, a 2.7. tételben látott bizonyításhoz hasonló módon, megkonstruálható egy ekvivalens egyszalagos, egyirányú Turing-gép.

Az M' működésének az alapötlete a következő. M' először megjelöli mindkét szalagján a fej kezdőpozícióját egy új, M szalagszimbólumai között nem szereplő \$ szimbólummal. M szalagjai közül rendre az első illetve a második szalag reprezentálja M szalagjának azon részét, amely M kezdőkonfigurációjában a fejtől jobbra illetve balra szerepel. Amikor M a fej kezdőpozíciójához képest jobbra dolgozik, akkor M' az első szalagon lemásolja M működését. Amikor M a fej kezdőpozíciójától egyet balra lép (ezt M' onnan tudja, hogy az első szalagján a fej \$-ra mutat), akkor M' a második szalagján kezd el dolgozni az alábbiak szerint. M minden egyes olyan lépését, mely a fej kezdőpozíciójától balra lévő szalagrészen történik M' egy a második szalagján történő ellentétes irányú lépéssel szimulálja (ezért M' második szalagján az M megfelelő szalagrészének a tükörképe lesz). Ha a fej egyszer csak újra a kezdőpozíciójától jobbra lévő szalagrészen kezd el dolgozni, akkor M' újra az első szalagján kezdi el szimulálni M működését. Mindeközben M' állapotai tárolják M állapotait plusz azt az információt, hogy M'-nek az első szalagon vagy a másodikon kell-e dolgoznia. □

2.3.1.3. Nemdeterminisztikus Turing-gépek

A Turing-gép átmeneti függvényét módosíthatjuk úgy, hogy a függvény értéke nem egy konkrét állapot-szalagszimbólum-irány hármas, hanem ezek egy véges halmaza. Az ilyen Turing-gépeket hívjuk nemdeterminisztikus Turing-gépeknek. Ebben az alfejezetben ezekkel a gépekkel ismerkedünk meg.

2.9. Definíció

A *nemdeterminisztikus Turing-gép* olyan $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ rendszer, ahol

- Q, Σ, Γ, q_0, q_i, q_n ugyanazok mint a 2.1. definícióban, és
- $\delta : (Q - \{q_i, q_n\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$.

Tehát egy M nemdeterminisztikus Turing-gép minden konfigurációjából néhány (esetleg nulla) különböző konfigurációba mehet át. Az M *konfigurációja* a deterministikus esettel megegyezően definiálható. A *konfigurációátmenet* pedig a determinisztikus eset értelemszerű kiterjesztése. Legyen például $uqav$ egy konfiguráció, ahol $a \in \Gamma$ és $u, v \in \Gamma^*$. Ekkor például $uqav \vdash urbv$, ha $(r, b, S) \in \delta(q, a)$.

Az M számítási sorozatai egy u szón legegyszerűbben egy fával reprezentálhatók. A fa csúcsa M kezdőkonfigurációja, a szögpontjai pedig M konfigurációi. A fa minden levele megfelel M azon számítási sorozatának az u-n, mely a gyökértől az adott levélig vezető úton előforduló konfigurációkat tartalmazza. M akkor fogadja el u-t, ha a fa valamelyik levele elfogadó konfiguráció. Az így definiált

fát nevezzük az M *nemdeterminisztikus számítási fájának az* u*-n.*

Az M által felismert nyelv a determinisztikus esethez hasonlóan definiálható, a gép által eldöntött nyelv pedig a következőképpen. Azt mondjuk, hogy egy nemdeterminisztikus Turing-gép *eldönti* az $L \subseteq \Sigma^*$ nyelvet ha felismeri, és minden $u \in \Sigma$ szóra M számítási sorozatai végesek, és elfogadási vagy elutasítási konfigurációba vezetnek. A nemdeterminisztikus Turing-gép definíciója értelemszerűen kiterjeszthető a többszalagos esetre is.

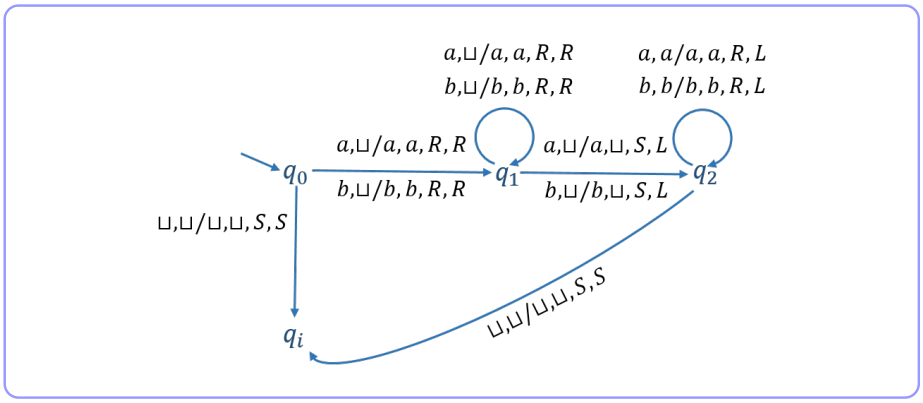
2.10. Példa

A 2.5. ábrán látható nemdeterminisztikus Turing-gép szintén az $L = \{uu^{-1} \mid u \in \{a,b\}^*\}$ nyelvet dönti el, de a nemdeterminisztikusságot felhasználva ennek a gépnek a felépítése a legegyszerűbb az eddig látott megoldások közül. Ez a Turing-gép a következőképpen működik.

1. Ha üres a bemenet, akkor q_i-be lép.
2. Ha nem üres a bemenet, akkor lemásolja az első betűt a második szalagra és q_1-be lép.
3. q_1-ben lemásol néhány betűt a második szalagra majd q_2-be lép (itt jelenik meg a nemdeterminizmus).
4. q_2-ben az első szalagon jobbra lépve a másodikon pedig balra megnézi, hogy ugyanazon betűk vannak-e a szalagokon.
5. Ha egyszerre látja a szó elejét jelző $\sqcup$-t a második és a szó végét jelző $\sqcup$-t az első szalagon, akkor q_i-be lép.

A be nem rajzolt átmenetek q_n-be vezetnek.
Látszik, hogy a gépnek, attól függően, hogy mikor hagyja abba a másolást, lehet számos nem elfogadó számítási sorozata. Másrészt viszont az is látható, hogy ha a bemenő szó eleme a nyelvnek, akkor lesz a gépnek egy olyan számítási sorozata, amely pont a szó felénél hagyja abba a szó másolását, és ekkor az első szalagon a fej pozíciójában kezdődő szó a második szalagon lévő szó tükörképe lesz. Tehát ebben az esetben a gép elfogadja a bemenetet, tekintet nélkül arra, hogy a szón az összes többi számítási sorozat sikertelen. Az is könnyen ellenőrizhető, hogy a nyelvbe nem tartozó szavakon sohasem kerülhet a gép elfogadó állapotba.

A nemdeterminisztikus Turing-gép időigényét a következő módon definiáljuk. Legyen $f : \mathbb{N} \to \mathbb{N}$ egy függvény és M egy nemdeterminisztikus Turing-gép. Azt mondjuk, hogy az M *időigénye* $f(n)$, ha egy n hosszú u bemeneten nincsenek M-nek $f(n)$-nél hosszabb számítási sorozatai, azaz M számítási fája az u-n legfeljebb $f(n)$ magas. A 2.10. példabeli Turing-gép $\mathcal{O}(n)$ időben dönti el az L nyelvet.



2.5. ábra. A 2.2. példában szereplő nyelvet eldöntő nemdeterminisztikus Turing-gép

Most belátjuk, hogy a nemdeterminizmus nem jelent plusz számítási erőt a Turing-gépek esetében.

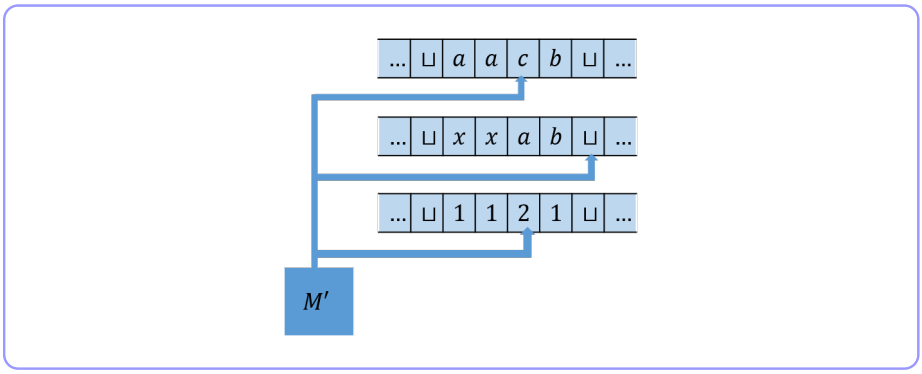
2.11. Tétel

Minden M nemdeterminisztikus Turing-géphez megadható egy vele ekvivalens M' determinisztikus Turing-gép.

Bizonyítás. Legyen M nemdeterminisztikus Turing-gép. Megadunk egy M' háromszalagos, determinisztikus Turing-gépet, ami ekvivalens M-mel. Azt már korábban láttuk, hogy M'-höz megadható egy ekvivalens egyszalagos Turing-gép.

M' első szalagja tartalmazza az u bemenő szót. A második szalagon történik az M számítási sorozatainak a szimulációja. Ez a szalag tartalmazza M egy konkrét számítási sorozatának a lépésenkénti eredményét. A harmadik szalagon lévő szó alapján szimulálja M' az M számítási sorozatait, egyesével véve sorra azokat. A szalagon a fej pozíciója mutatja azt, hogy hányadik lépését szimulálja éppen M' az M aktuális számítási sorozatának. A harmadik szalagon tulajdonképpen az u-t elfogadó konfigurációk egy szélességi keresése történik a nemdeterminisztikus számítási fában. M' egy konfigurációja a 2.6. ábrán látható.

Legyen d az M átmenetfüggvénye által megadott halmazok közül a legnagyobb elemszámúnak a számossága. Tegyük fel továbbá, hogy az átmenetfüggvény által megadott halmazokban az elemeknek van egy rögzített sorrendje. Világos, hogy az u számítási fájának minden szögpontjához egyértelműen hozzárendelhető egy $\Sigma = \{1, \ldots, d\}$ feletti szó. Nevezetesen az, amelyik megmutatja, hogy



2.6. ábra. Nemdeterminisztikus Turing-gép szimulálása determinisztikussal

a kezdőkonfigurációból az átmenetfüggvény nemdeterminisztikus lehetőségei közül mely választásokkal juthatunk el az adott szögpontban lévő konfigurációhoz.

Maga a szimuláció a következőképpen történik:

1. Kezdetben az 1-es szalag tartalmazza az u bemenő szót, a 2-es és 3-as szalagok üresek.
2. M' átmásolja az első szalag tartalmát a második szalagra.
3. M' szimulálja a második szalagon M egy számítási sorozatát. Az, hogy melyiket kell szimulálnia a harmadik szalagon lévő szótól függ. M' a szimuláció minden lépése előtt megnézi a 3-ik szalagján lévő szó következő betűjét, és e betű szerint (ami egy szám Σ-ból) választ M átmenetfüggvényének a lehetőségei közül. Ha nincs megfelelő sorszámú választási lehetőség, vagy a 3-ik szalagon már nincs több betű, akkor a szimuláció véget ér, és a 4-ik lépésre ugrunk.
   Ha a lépés szimulálása során M elutasító konfigurációba kerül, akkor szintén a 4-ik lépésre ugrunk.
   Végül, ha M' azt látja, hogy M elfogadó konfigurációba kerül, akkor M' is elfogadó állapotba lép és megáll.

Van-e hatékonyabb szimuláció?

Belátható, hogy a 2.11. tétel bizonyításában szereplő konstrukció exponenciális időigény romlást eredményez. Tehát a megkonstruált M' determinisztikus Turing-gép nagyságrendekkel lassabb, mint a szimulált nemdeterminisztikus M. Ez nem is meglepő: amíg M időigénye a számítási fa magasságával van meghatározva, addig M' időigénye a számítási fában lévő konfigurációk számával arányos.
Felmerül a kérdés, hogy létezik-e jelentősen hatékonyabb szimuláció. A válasz az, hogy ilyen szimulációt nem ismerünk, de azt sem tudjuk jelenleg bizonyítani, hogy nincs ilyen. A kérdésre még visszatérünk a bonyolultságelmélet résznél a **P** és **NP** osztályok kapcsán.

4. M' kicseréli a 3-ik szalagján lévő szót az azt lexikografikusan követő szóra, a fejet a szó elejére állítja és a 2-ik pontra ugorva újrakezdi M működésének szimulálását.

Látható, hogy M' pontosan akkor kerül elfogadó konfigurációba egy szón, ha M-nek van ezen a szón elfogadó konfigurációba vezető számítási sorozata. Következik tehát, hogy M és M' ekvivalensek. □

2.4. Eldönthetetlen problémák

Ebben a fejezetben megmutatjuk, hogy bár a Turing-gép a Church-Turing tézis alapján a lehető legáltalánosabb algoritmus modell, mégis vannak olyan problémák, melyek nem számíthatók ki vele.

Először felidézzük azt (lásd a 2.3. definíciót), hogy egy L nyelvet Turing-felismerhetőnek (vagy rekurzívan felsorolhatónak) nevezünk, ha van olyan M Turing-gép, ami az összes L-beli szóra q_i-ben áll meg, a többi szóra pedig q_n-ben áll meg, vagy esetleg nem áll meg. Továbbá L-et eldönthetőnek (vagy rekurzívnak) nevezzük, ha M minden bemeneten meg is áll. Világos, hogy fennáll az $\mathbf{R} \subseteq \mathbf{RE}$ tartalmazás. A célunk az, hogy megmutassuk, az $\mathbf{R}$ valódi részhalmaza az $\mathbf{RE}$-nek, azaz van olyan nyelv (probléma) ami Turing-felismerhető, de nem eldönthető.

Ebben a fejezetben csak olyan Turing-gépeket fogunk vizsgálni, melyek bemenő ábécéje a $\{0,1\}$ halmaz. Ez nem jelenti az általánosság megszorítását, hiszen ha találunk egy olyan $\{0,1\}$-feletti nyelvet, melyet nem lehet eldönteni ilyen Turing-géppel, akkor ezt a nyelvet egyáltalán nem lehet eldönteni (ne felejtsük el, hogy a Turing-gépnek csak a bemenő ábécéjét szorítottuk meg, a szalagábécéjét nem).

Szorgos hód (busy beaver) Turing-gépek

A következő részekben számos Turing-géppel kapcsolatos problémáról látni fogjuk, hogy eldönthetetlen. Mint például az is, hogy vajon megáll-e egy M Turing-gép egy w bemeneten. Gondolhatnánk, hogy mi sem könnyebb, mint ezt eldönteni: futtassuk M-et egészen addig, amíg megáll vagy amíg már látjuk, hogy biztosan nem fog megállni. Sajnos nem ilyen egyszerű a helyzet.

Az n állapotú *szorgos hód* az az M Turing-gép, ami az üres szalaggal indítva bizonyíthatóan megáll, csak az $\sqcup$ és 1 szimbólumokat írja a szalagjára, és nincs olyan Turing-gép ami ugyanilyen feltételek mellett több 1-est írna a szalagjára mint az M.

A mai napig nem ismert, hogy melyik Turing-gép a 6 állapotú szorgos hód. A legjobb jelölt egy olyan Turing-gép ami kb. $3.5 \cdot 10^{18267}$ darab egyest ír a szalagjára mielőtt megállna (és összesen kb. $7.4 \cdot 10^{36534}$-et lép ezalatt). Ezek alapján sejthető, hogy miért kilátástalan szimulációval a véges és végtelen viselkedést megkülönböztetni egy tetszőleges Turing-gépre még az üres bemenet esetén is.

Az egyik nyelv melyről belátjuk majd, hogy eldönthetetlen a következő: azon (M, w) párok halmaza (egy megfelelő bináris szóval kódolva), ahol M egy $\{0,1\}$ bemenő ábécé feletti Turing-gép, w pedig egy $\{0,1\}$-feletti szó úgy, hogy $w \in L(M)$, azaz M elfogadja w-t. Ezt a nyelvet *univerzális nyelvnek* nevezzük és L_{u}-val jelöljük. Tehát $L_{\mathrm{u}} = \{\langle M, w\rangle \mid w \in L(M)\}$, ahol a $\langle M, w\rangle$ jelölés a később megadásra kerülő bináris kódolást jelenti.

Ahhoz, hogy meg tudjuk mutatni, hogy $L_{\mathrm{u}} \notin \mathbf{R}$, először a következő feladatokat kell elvégeznünk. Először belátjuk, hogy a $\{0,1\}$ ábécé feletti Turing-gépek egyértelműen kódolhatóak bináris szavakkal. Ezután pedig megmutatjuk, hogy az a nyelv, mely azon $\{0,1\}$-feletti Turing-gépek bináris kódjait tartalmazza, melyek nem fogadják el önmaguk kódját mint bemenő szót, egy olyan nyelv, ami nem rekurzív, sőt még csak nem is rekurzívan felsorolható. Ezt a nyelvet *diagonális nyelvnek* nevezzük, és $L_{\text{átló}}$-val jelöljük.

2.4.1. Turing-gépek kódolása

Először megjegyezzük, hogy a $\{0,1\}$-feletti szavak sorba rendezhetőek. Valóban, tekintsük azt a felsorolást, amelyben a szavak a hosszuk szerint követik egymást, és két egyforma hosszú szó közül pedig az van előbb, amelyik a lexikografikus rendezés szerint megelőzi a másikat. Ily módon a $\{0,1\}^*$ halmaz elemeinek egy felsorolása a következőképpen alakul: $w_1 = \varepsilon$, $w_2 = 0$, $w_3 = 1$, $w_4 = 00$, $w_5 = 01$, és így tovább. Ebben a fejezetben a w_i szóval a $\{0,1\}^*$ halmaz i-ik elemét jelöljük. Könnyű belátni, hogy egy adott i számra, a w_i szó véges sok lépésben kiszámítható.

Legyen a továbbiakban $M = (Q, \{0,1\}, \Gamma, \delta, q_0, q_i, q_n)$ Turing-gép. Legyen $|Q| = k$ és $|\Gamma| = m$. Ekkor Q-t felírhatjuk $Q = \{p_1, p_2, \ldots, p_k\}$ alakban, ahol $p_1 = q_0$, $p_{k-1} = q_i$ és $p_k = q_n$ (azaz tulajdonképpen megsorszámozzuk Q elemeit az $1, \ldots, k$ számokkal úgy, hogy a két végállapot, q_i és q_n legyen az utolsó). Továbbá Γ felírható $\Gamma = \{X_1, X_2, \ldots, X_m\}$ alakban, ahol $X_1 = 0$, $X_2 = 1$, $X_3 = \sqcup$ és $X_4, \ldots, X_m$ az M további szalagszimbólumai. Nevezzük végül az L, R és S szimbólumokat, azaz a Turing-gép átmenetfüggvényében szereplő irányokat, rendre a D_1, D_2 és D_3 iránynak. Ezek után M egy $\delta(p_i, X_j) = (p_r, X_s, D_t)$ átmenete $(i, r \in [k]$, $j, s \in [m]$ és $t \in [3])$ kódolható a $0^i 1 0^j 1 0^r 1 0^s 1 0^t$ szóval. Valóban, az első és harmadik 0-s blokkban szereplő 0-k száma megadja az átmenetben szereplő állapotok indexeit. A második, negyedik és ötödik 0-s blokkban szereplő 0-k száma pedig rendre az átmenetben szereplő szalagszimbólumok és az irány indexeit adják. Továbbá, mivel minden 0-s blokk hossza legalább 1, az átmenetet kódoló szóban nem szerepel az 11 részszó. Tehát az M összes átmenetét kódoló szavakat összefűzhetjük egy olyan szóvá, melyben az átmeneteket az 11 részszó választja el egymástól. Az így kapott szó lesz M kódja, amit $\langle M\rangle$-mel fogunk jelölni..

A továbbiakban minden $i \geq 1$-re M_i-vel jelöljük azt a Turing-gépet, amit a w_i bináris szó kódol (emlékeztetőül, w_i az i-ik elem a $\{0,1\}^*$ elemeit felsoroló listá-

ban). Megegyezünk továbbá abban, hogy ha valamely i-re w_i nem egyezik meg egyetlen Turing-gép (fent leírt) kódolásával sem, akkor M_i-t egy olyan Turing-gépnek tekintjük, ami minden inputon azonnal a q_n állapotba megy, vagyis $L(M_i) = \emptyset$. A későbbiekben szükségünk lesz arra, hogy kódoljunk egy (M, w) Turing-gép és bemenet párost egy $\{0,1\}$-feletti szóval. Ehhez felhasználjuk azt, hogy a Turing-gépek fenti kódolása nem tartalmazhat három 1-est egymás mellett. Ezért az (M, w) párt úgy kódoljuk el, hogy M kódja után írjuk az 111 szót, ezután pedig w-t. A továbbiakban egy (M, w) párt kódoló $\langle M \rangle 111 w$ szó helyett gyakran írunk egyszerűen $\langle M, w \rangle$-t.

2.4.2. A diagonális nyelv

Felhasználva a Turing-gépek előbb látott kódolását, most már pontosan is definiálni tudjuk, hogy mit értünk a fent említett $L_{\text{átló}}$ nyelv alatt: $L_{\text{átló}} = \{w_i \mid i \geq 1, w_i \notin L(M_i)\}$. Először azt mutatjuk meg, hogy ez a nyelv nem rekurzívan felsorolható.

Ezt a nyelvet azért nevezzük diagonális nyelvnek, mert a karakterisztikus függvénye megkapható az alábbi módon. Tekintsük azt a T táblázatot, melynek oszlopai rendre a $w_1, w_2, \ldots$ szavakkal, a sorai pedig az $M_1, M_2, \ldots$ Turing-gépekkel vannak címkézve (tehát T egy végtelen kétdimenziós mátrix). A T elemei 0 és 1 lehetnek az alábbiak szerint. Minden $i, j \geq 1$-re, $T(i, j)$ értéke pontosan akkor 1, ha $w_j \in L(M_i)$. Tegyük fel, hogy az így kitöltött táblázatunk a 2.7. ábrán látható módon néz ki.

A diagonális módszer

Az olvasónak ismerős lehet ahogyan az $L_{\text{átló}} \notin \mathbf{RE}$ eredményt bizonyítottuk. Ezt a módszert Georg Cantor dolgozta ki és használta először annak bizonyítására, hogy a $[0, 1]$ intervallumbeli valós számok számossága nagyobb, mint a természetes számok számossága.

Ebben a táblázatban például a második sor második eleme 1, ami azt jelenti, hogy M_2 elfogadja a w_2 szót (ez persze a valóságban nem igaz, hisz w_2 nem kódol legális Turing-gépet, így M_2 nem is ismerhet fel semmilyen szót, de az összefüggések szemléltetéséhez feltesszük, hogy a fenti táblázat helyes).

Minden $i \geq 1$-re, a fenti táblázat i-ik sora tekinthető úgy, mint az $L(M_i)$ nyelv karakterisztikus függvénye. Nevezetesen azért, mert minden $j \geq 1$-re, a w_j szó pontosan akkor eleme $L(M_i)$-nek, ha az i-ik sor j-ik eleme 1. Ebben az értelemben a táblázat átlójában szereplő bitsorozat komplementere pedig nem más, mint az $L_{\text{átló}}$ karakterisztikus függvénye. Ezért nyilvánvaló, hogy minden $i \geq 1$-re az $L_{\text{átló}}$ karakterisztikus függvénye különbözik $L(M_i)$ karakterisztikus függvényétől. Mivel minden M Turing-géphez van olyan $i \geq 1$ szám, hogy $L(M) = L(M_i)$, kapjuk, hogy az $L_{\text{átló}}$ nem lehet egyenlő az $L(M)$-mel semmilyen M Turing-gépre. Adódik tehát a következő tétel:

T	w_1	w_2	w_3	w_4	...
M_1	0	0	0	0	...
M_2	0	1	0	0	...
M_3	1	0	1	0	...
M_4	0	0	1	1	...
⋮	⋮	⋮	⋮	⋮	

2.7. ábra. Az $L_{\text{átló}} \notin \mathbf{RE}$ bizonyításában használt T táblázat

2.12. Tétel

$L_{\text{átló}} \notin \mathbf{RE}$.

2.4.3. Az univerzális nyelv

Az előző alfejezet eredményét felhasználva szeretnénk megmutatni, hogy az univerzális nyelv olyan rekurzívan felsorolható nyelv, ami nem rekurzív.

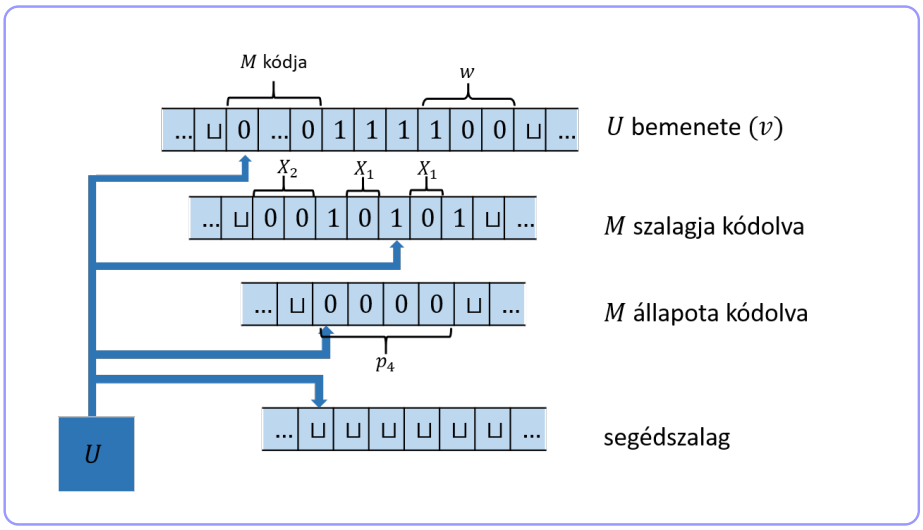
Először azt mutatjuk meg, hogy L_{u} rekurzívan felsorolható.

2.13. Tétel

$L_{\text{u}} \in \mathbf{RE}$.

Bizonyítás. Megadunk egy olyan U Turing-gépet, ami L_{u}-t ismeri fel. Ezt a Turing-gépet *univerzális Turing-gépnek* nevezzük, mert mint látni fogjuk U szimulálja a bemenetén kapott $\langle M, w\rangle$ szó által kódolt M Turing-gép működését a w szón.

U-nak négy szalagja van. Először ellenőrzi, hogy a bemenete $\langle M, w\rangle$ alakú szóe. Ha igen, akkor elkezdi szimulálni M működését. U a második szalagján tárolja az M szalagját, mégpedig ugyanolyan a kódolással, mint amilyen az M kódolásánál is használatos. Vagyis az M X_j szalagszimbóluma a 0^j szóval van reprezentálva és a szalagszimbólumok pedig egy 1-es szimbólummal vannak elválasztva. A harmadik szalagon tárolja U az M aktuális állapotát a szokásos



2.8. ábra. Az L_{u} nyelvet felismerő U Univerzális Turing-gép

módon kódolva, azaz a p_i állapotot a 0^i szóval reprezentálva. Az U felépítése a 2.8. ábrán látható:

Az U működése egy v bemeneten az alábbi módon foglalható össze.

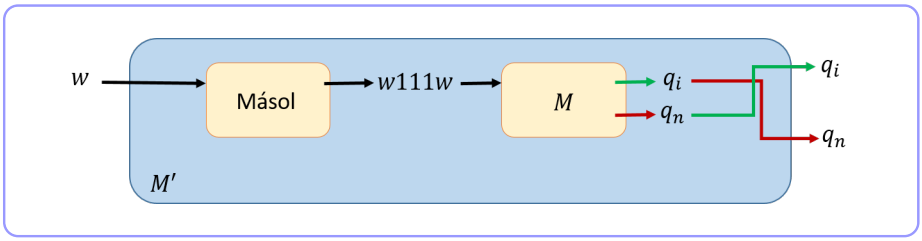
1. U először megvizsgálja, hogy a v szó $\langle M, w\rangle$ alakú-e. Ha nem, akkor elutasítja a bemenetet. Ezt helyesen teszi, hisz ha a szó eleje nem a kódolásnak megfelelő, akkor megállapodásunk szerint v egy olyan Turing-gépet kódol, ami nem fogad el egyetlen bemenő szót sem. Ekkor viszont v nem lehet eleme L_{u}-nak.

2. U átmásolja w-t a második szalagra a kódolt formában.

3. U 0-t ír a harmadik szalagjára, ezzel reprezentálva M kezdőállapotát.

4. U szimulálja M egy lépését az alábbi módon. Keres egy $0^i10^j10^r10^s10^t$ alakú részszót M kódjában az első szalagján úgy, hogy 0^i a harmadik szalagon lévő állapot legyen, 0^j pedig az a 0-s blokk, ami a második szalagon a fej pozíciójában kezdődik. Ezek után U elvégzi a fent kódolt átmenet alapján M egy lépését:

   - Kitörli a harmadik szalagon 0^i-t és a másodikról átmásolja a harmadikra 0^r-t (annak az állapotnak a kódját, amibe M lép).
   - Kicseréli a második szalagon 0^j-t 0^s-re, azaz átírja M szalagszimbólumát az átmenetnek megfelelően. Ehhez, ha kell, felhasználja a



2.9. ábra. $L_{\text{átló}}$ eldöntése egy L_{u}-t eldöntő Turing-gép segítségével

negyedik szalagot, ugyanis ha $j \neq s$, akkor rámásolja a negyedik szalagra a második szalagról a 0^j mögötti részt, kitörli a második szalagon 0^j-t, a helyére írja 0^s-t, végül visszamásolja a negyedik szalagon lévő szót a második szalagra, és a fejet a 0^s blokk elejére állítja.

- U megkeresi a második szalagon a fej pozícióján kezdődő 0-s blokktól balra vagy jobbra lévő 0-s blokk kezdetét, vagy helyben marad attól függően, hogy t értéke 1, 2, vagy 3.

5. Ha U azt találja, hogy M a 4-ik pontban elfogadó vagy elutasító állapotba lépett, akkor U is q_i-be vagy q_n-be lép. Egyébként pedig folytatja M következő lépésének szimulálását a 4-ik ponttal.

Világos, hogy U pontosan akkor fogad el egy $\langle M, w\rangle$ alakú bemenetet, ha M elfogadja w-t, vagyis ha $\langle M, w\rangle \in L_{\text{u}}$. Tehát U az univerzális nyelvet ismeri fel. Ezzel a bizonyítást befejeztük. □

Ezek után megmutatjuk, hogy az univerzális nyelv nem eldönthető.

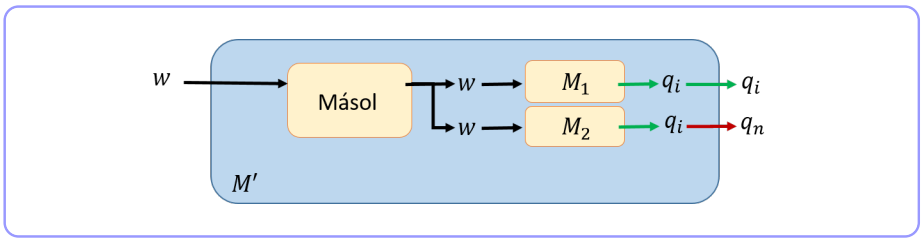
2.14. Tétel

$L_{\text{u}} \notin \mathbf{R}$.

Bizonyítás. Az állítást indirekt módon bizonyítjuk. Tegyük fel, hogy L_{u} eldönthető, és legyen M egy Turing-gép, ami eldönti L_{u}-t. M-ből megkonstruálunk egy olyan M' Turing-gépet, ami $L_{\text{átló}}$-t dönti el. M' felépítése a 2.9. ábrán látható.

Adott w bemenetre M' a következőképpen működik:

1. Előállítja w-ből a $w' = w111w$ szót.
2. w'-n szimulálja M-et
3. Ha M elfogadja w'-t, akkor M' elutasítja w-t.



2.10. ábra. Egy L-et eldöntő Turing-gép konstrukciója az L és $\overline{L}$ nyelveket felismerő gépek felhasználásával

4. Ha M elutasítja w'-t, akkor M' elfogadja w-t.

Mivel M minden bemeneten megáll, M' is megáll minden bemeneten. Továbbá $w \in L(M')$ akkor és csak akkor, ha $w' \notin L_{\text{u}}$, azaz ha a w által kódolt Turing-gép nem fogadja el önmaga kódját bemenetként. Ez azt jelenti, hogy $w \in L(M')$ pontosan akkor teljesül, ha $w \in L_{\text{átló}}$. Azt kaptuk tehát, hogy $L(M') = L_{\text{átló}}$, vagyis $L_{\text{átló}}$ eldönthető. Ez viszont ellentmond a 2.12. tételünknek, mely szerint $L_{\text{átló}} \notin \mathbf{RE}$. Ezzel a tétel bizonyítását befejeztük. □

Most egy olyan tételt bizonyítunk, amely a 2.13. és 2.14. tételekkel együtt azt eredményezi hogy a rekurzívan felsorolható nyelvek nem zártak a komplementer képzésre.

2.15. Tétel

Legyen L egy nyelv. Ha $L, \overline{L} \in \mathbf{RE}$, akkor $L \in \mathbf{R}$.

Bizonyítás. Ha L és $\overline{L}$ is rekurzívan felsorolható, akkor van olyan M_1 és M_2 Turing-gép, mely rendre L-et és $\overline{L}$-et ismeri fel. M_1 és M_2 felhasználásával megkonstruálunk egy olyan M Turing-gépet, ami eldönti L-et. M konstrukciója a 2.10. ábrán látható.

M egy olyan kétszalagos gép, ami egy w bemeneten az alábbiakat teszi. Első lépésként a második szalagra másolja w-t. Ezután egy ciklusban szimulálja az M_1 egy lépését az első szalagon és az M_2 egy lépését a másodikon. A ciklus addig fut, amíg M_1 és M_2 közül valamelyik elfogadó állapotba nem lép. Mivel w vagy az $L(M_1)$-nek vagy az $L(M_2)$-nek az eleme, véges számú szimulációs lépés után vagy M_1 vagy M_2 elfogadja w-t, tehát valamelyik véges sok lépés után biztosan q_i-be lép. Ha M_1 lép q_i-be, akkor M elfogadja a bemenetet, ha M_2, akkor pedig elutasítja azt. Könnyen látható, hogy M is L-et ismeri fel ráadásul minden bemeneten megáll, tehát el is dönti L-et. Következésképpen $L \in \mathbf{R}$. □

Hogyan lehet belátni ezek után azt, hogy a rekurzívan felsorolható nyelvek nem zártak a komplementerképzésre? Tekintsük például az L_{u} nyelv komplemen-

terét, azaz az $\overline{L}_{\mathrm{u}}$ nyelvet. Azt állítjuk, hogy $\overline{L}_{\mathrm{u}} \notin \mathbf{RE}$. Valóban, tegyük fel, hogy $\overline{L}_{\mathrm{u}} \in \mathbf{RE}$. Akkor, mivel a 2.13. tétel alapján $L_{\mathrm{u}} \in \mathbf{RE}$, a 2.15. tételből következik, hogy $L_{\mathrm{u}} \in \mathbf{R}$. Ez viszont ellentmond annak, hogy a 2.14. tétel alapján $L_{\mathrm{u}} \notin \mathbf{R}$. Kapjuk tehát, hogy $\overline{L}_{\mathrm{u}}$ nem lehet rekurzívan felsorolható, amiből következik az alábbi megállapítás.

2.16. Következmény

A rekurzívan felsorolható nyelvek nem zártak a komplementerképzésre.

A rekurzívan felsorolható nyelvekkel ellentétben a rekurzív nyelvek zártak a komplementer képzésre.

2.17. Tétel

Ha $L \in \mathbf{R}$, akkor $\overline{L} \in \mathbf{R}$.

Bizonyítás. Legyen L egy rekurzív nyelv. Ekkor van olyan M Turing-gép, ami eldönti L-et. Tudjuk, hogy ekkor M olyan, hogy minden L-beli szón q_i-ben áll meg, az összes többi szón pedig q_n-ben áll meg, vagyis minden szón megáll. Legyen M' az a Turing-gép, amit úgy kapunk M-ből, hogy M összes olyan átmenetét, ami q_i-be megy q_n-be irányítjuk, a q_n-be menő átmeneteket pedig q_i-be. Nem nehéz belátni, hogy az így definiált M' az $\overline{L}$ nyelvet dönti el. □

2.5. További eldönthetetlen problémák

Ebben a fejezetben néhány további Turing-gépekkel kapcsolatos eldönthetetlen probléma mellett megvizsgálunk több olyat is, melyek nem kapcsolódnak közvetlenül a Turing-gépekhez. Először viszont végiggondoljuk, hogyan lehet felhasználni egy eldönthetetlen problémát további problémák eldönthetetlenségének bizonyítására.

Tegyük fel, hogy van két eldöntési problémánk, legyenek ezek L_1 és L_2. Előfordulhat, hogy csak az L_2 probléma eldöntésére van egy A_2 algoritmusunk, az L_1-ére nincs. Viszont, ha igaz az, hogy az L_1 probléma minden w példányához meg tudjuk konstruálni a L_2 probléma egy w' példányát úgy, hogy a w pontosan akkor igen példánya L_1-nek, ha w' igen példánya L_2-nek, akkor már az L_1 problémát is el tudjuk dönteni az alábbi módon. Az L_1-et eldöntő A_1 algoritmus legyen a következő: egy w bemenetből először készítsük el az L_2 fentebb leírt w' példányát, ezután szimuláljuk A_2 működését a w' bemeneten. Végül A_1 pontosan akkor adjon *igen* választ a w bemenetre, ha A_2 *igen* választ ad a w' bemenetre. Könnyen látható, hogy az így leírt A_1 algoritmus valóban az L_1

problémát dönti el. A most vázolt módszert nevezzük visszavezetésnek, hiszen az L_1 probléma eldöntését visszavezettük az L_2 probléma eldöntésére.

A visszavezetést felhasználhatjuk arra is, hogy megmutassuk egy problémáról, hogy eldönthetetlen. Tudjuk például, hogy az előző fejezetben vázolt L_{u} probléma eldönthetetlen. Ha L_{u}-t sikerülne visszavezetni egy másik problémára, akkor ezáltal bizonyítani tudnánk, hogy ez az újabb probléma is eldönthetetlen (gondoljuk meg, ha az újabb problémánk mégis eldönthető lenne, akkor a visszavezetést felhasználva meg tudnánk adni egy L_{u}-t eldöntő algoritmust, amiről tudjuk, hogy nem létezik, ezért ez ellentmondáshoz vezetne).

A fent vázolt visszavezetést formálisan az alábbi módon definiáljuk.

2.18. Definíció

Legyen Σ és Δ két ábécé és f egy Σ^*-ból Δ^*-ba képező függvény. Azt mondjuk, hogy f *kiszámítható*, ha van olyan M Turing-gép, hogy M-et egy $w \in \Sigma^*$ szóval a bemenetén indítva, M úgy áll meg, hogy a szalagján az $f(w)$ szó van.
Ezután legyen $L_1 \subseteq \Sigma^*$ és $L_2 \subseteq \Delta^*$ két nyelv (azaz eldöntési probléma). Azt mondjuk, hogy L_1 *visszavezethető* L_2-re (jele $L_1 \leq L_2$), ha van olyan $f : \Sigma^* \to \Delta^*$ kiszámítható függvény, hogy minden $w \in \Sigma^*$ szóra, $w \in L_1$ akkor és csak akkor, ha $f(w) \in L_2$.

Az alábbi tétel tükrözi a fejezet elején a visszavezetésekről elmondottakat.

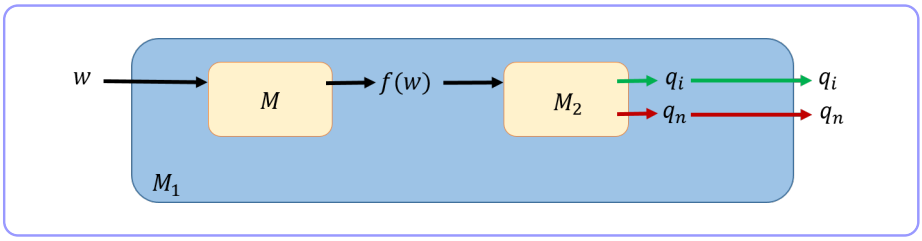
2.19. Tétel

Legyen $L_1 \subseteq \Sigma^*$ és $L_2 \subseteq \Delta^*$ két eldöntési probléma és tegyük fel, hogy $L_1 \leq L_2$. Ekkor igazak az alábbi állítások:

1. Ha $L_1 \notin \mathbf{R}$, akkor $L_2 \notin \mathbf{R}$.
2. Ha $L_1 \notin \mathbf{RE}$, akkor $L_2 \notin \mathbf{RE}$.

Bizonyítás. Csak a második állítást bizonyítjuk, az első bizonyítása hasonló. Indirekt módon tegyük fel, hogy L_2 mégis rekurzívan felsorolható. Ekkor van olyan M_2 Turing-gép, hogy $L_2 = L(M_2)$. Továbbá van olyan M Turing-gép, ami kiszámolja az $L_1 \leq L_2$-ben alkalmazott f függvényt. Ezen gépek segítségével megadunk egy olyan M_1 Turing-gépet, ami L_1-et ismeri fel. M_1 szerkezete a 2.11. ábrán látható.

M_1 egy w bemeneten először szimulálja az M gépet. Amikor végez, akkor a szalagján lévő $f(w)$ szón szimulálja M_2 működését. Ha M_2 elfogadja az $f(w)$ szót, akkor M_1 is elfogadja a w-t. Ha M_2 elutasítja $f(w)$-t, akkor M_1 is elutasítja w-t. Megjegyezzük, hogy ha M_2 nem áll meg az $f(w)$ bemeneten, akkor M_1 sem áll



2.11. ábra. A 2.19. tétel bizonyításában konstruált Turing-gép

meg w-n. Könnyen belátható, hogy az így vázolt M_1 gép pontosan az L_1 nyelvet ismeri fel. Feltevésünk szerint azonban az $L_1 \notin \mathbf{RE}$, tehát ellentmondáshoz jutottunk. Következésképpen L_2 nem lehet rekurzívan felsorolható. □

2.5.1. Turing-gépekkel kapcsolatos eldönthetetlenségi kérdések

2.5.1.1. A megállási probléma

A fentiek demonstrálására tekintsük a következő problémát, mely a Turing-gépek megállási problémájaként is ismert.

Legyen $L_\mathrm{h} = \{\langle M, w\rangle \mid M \text{ megáll a } w \text{ bemeneten}\}$, azaz L_h azon $\langle M, w\rangle$ Turing-gép és bemenet párosokat tartalmazza megfelelően kódolva, melyekre teljesül, hogy az M gép megáll a w bemeneten. Nevezzük ezt a nyelvet *megállási problémának*.

Nyilvánvaló, hogy ha lenne olyan Turing-gép, ami képes lenne eldönteni a megállási problémát, akkor tetszőleges Turing-gépről, (és így a Church-Turing tézis értelmében tetszőleges algoritmusról, C++, Pascal programról, stb.) el tudnánk dönteni, hogy megáll-e a bemenetén vagy sem. Ez a Turing-gép egy igen hasznos eszköz lenne a számunkra, sajnos azonban ilyen gép nem létezik.

2.20. Tétel

$L_\mathrm{h} \in \mathbf{RE} - \mathbf{R}$.

Bizonyítás. Először megmutatjuk, hogy $L_\mathrm{u} \leq L_\mathrm{h}$ ami a 2.19. tétel szerint, mivel $L_\mathrm{u} \notin \mathbf{R}$, bizonyítja azt, hogy $L_\mathrm{h} \notin \mathbf{R}$. A visszavezetés a következő módon történik. Egy tetszőleges M Turing-géphez legyen M' a következő Turing-gép.

M' egy w bemeneten:

1. Futtatja M-et a w szón (tulajdonképpen meghívja az U univerzális Turing-

gépet az $\langle M, w\rangle$ szóra).

2. Ha M elfogadja a w bemenetet, akkor M' is elfogadja w-t.

3. Ha M elutasítja w-t, akkor M' egy olyan állapotba megy, ahol egy végtelen ciklusban lépteti a fejet jobbra, tehát M' ebben az esetben soha nem áll meg.

Könnyű belátni, hogy az $\langle M, w\rangle \mapsto \langle M', w\rangle$ leképezés kiszámítható. Továbbá $\langle M, w\rangle \in L_{\mathrm{u}}$ akkor és csak akkor, ha $\langle M', w\rangle \in L_{\mathrm{h}}$. Azt kaptuk tehát, hogy L_{u} visszavezethető L_{h}-ra, amiből következik, hogy L_{h} eldönthetetlen.

Ezután megmutatjuk, hogy $L_{\mathrm{h}} \leq L_{\mathrm{u}}$ amiből a 2.19. tétel 2-ik pontja alapján, felhasználva, hogy $L_{\mathrm{u}} \in \mathbf{RE}$ adódik, hogy $L_{\mathrm{h}} \in \mathbf{RE}$.

Tetszőleges $\langle M, w\rangle$ Turing-gép–bemenet pároshoz legyen $\langle M', w\rangle$ az a páros, melyben az M' Turing-gépet úgy kapjuk M-ből, hogy annak minden q_n-be vezető átmenetét q_i-be irányítjuk. Nem nehéz belátni, hogy egy M Turing-gépre és annak w bemenő szavára az $\langle M, w\rangle \in L_{\mathrm{h}}$ akkor és csak akkor, ha $\langle M', w\rangle \in L_{\mathrm{u}}$. Azaz a fent vázolt konstrukció valóban az L_{h} visszavezetése L_{u}-ra. □

2.5.1.2. További eldönthetetlen kérdések

Tekintsük az $L_{\neg\emptyset} = \{\langle M\rangle \mid L(M) \neq \emptyset\}$ nyelvet.

2.21. Tétel

$L_{\neg\emptyset} \notin \mathbf{R}$.

Bizonyítás. Visszavezetjük az L_{u}-t $L_{\neg\emptyset}$-re. A visszavezetés egyik módja a következő. Adott (M, w) Turing-gép–bement pároshoz konstruáljuk meg a következő M' Turing-gépet. M' egy adott u bemeneten a következőt teszi:

1. Megnézi, hogy $u = w$ teljesül-e. Ha nem, akkor elutasítja u-t. Egyébként szimulálja az M-et az u-n (azaz w-n).

2. Ha M elfogadja u-t, akkor M' is q_i-be lép, ha M elutasítja u-t, akkor M' is q_n-be lép.

Könnyen látható, hogy $\langle M'\rangle \in L_{\neg\emptyset} \Leftrightarrow L(M') \neq \emptyset \Leftrightarrow w \in L(M) \Leftrightarrow \langle M, w\rangle \in L_{\mathrm{u}}$, vagyis a fenti konstrukció az L_{u} visszavezetése az $L_{\neg\emptyset}$-re. A 2.19. tétel alapján tehát az $L_{\neg\emptyset}$ is eldönthetetlen. □

Most azt mutatjuk meg, hogy $L_{\neg\emptyset}$ felismerhető Turing-géppel.

2.22. Tétel

$L_{\neg\emptyset} \in \mathbf{RE}$.

Bizonyítás. Tekintsük azt az M' kétszalagos Turing-gépet, ami egy adott $\langle M \rangle$ bemeneten a következőképpen működik.

1. M' a második szalagjára írja a 0-t.

2. Jelöljük a második szalagon lévő számot i-vel. M' egy ciklusban szimulálja a bemenetén kapott M Turing-gép első i lépését rendre a $w_1, \ldots, w_i$ szavakon.

3. Ha M' azt találja, hogy M valamelyik szón elfogadó állapotba lép, akkor M' is átlép a saját elfogadó állapotába. Egyébként növeli a második szalagon lévő számot eggyel és folytatja működését a második lépéssel.

Világos, hogy az így definiált M' elfogadja $\langle M \rangle$-et ha, van olyan w_j $(j \geq 1)$ szó, melyre $w_j \in L(M)$, egyébként pedig nem áll meg. Kapjuk tehát, hogy adott M Turing-gépre, $\langle M \rangle \in L_{\neg\emptyset} \Leftrightarrow \langle M \rangle \in L(M')$, vagyis M' az $L_{\neg\emptyset}$ nyelvet ismeri fel. □

Tekintsük az $L_{\neg\emptyset}$ nyelv komplementerét, azaz az $L_\emptyset = \{\langle M \rangle \mid L(M) = \emptyset\}$ nyelvet. A 2.17. tétel alapján $L_\emptyset$ sem eldönthető. Sőt, a 2.15, 2.21. és 2.22. tételek alapján könnyedén adódik az alábbi állítás.

2.23. Következmény

$L_\emptyset \notin \mathbf{RE}$.

Tegyük fel, hogy adottak a P_1 és P_2 problémák úgy, hogy a P_1 a P_2 speciális esete. Ekkor, ha $P_1 \notin \mathbf{R}$ (rendre, $P_1 \notin \mathbf{RE}$), akkor nyilvánvalóan $P_2 \notin \mathbf{R}$ (rendre, $P_2 \notin \mathbf{RE}$) is teljesül. Gondoljuk végig, ha például a P_1-nél általánosabb P_2 eldönthető lenne, akkor nyilvánvalóan a P_1-et is el lehetne dönteni.

Tekintsük például a Turing-gépek *ekvivalencia-problémáját*, azaz az

$$L_{\mathrm{EQ}} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ és } M_2 \text{ ekvivalens Turing-gépek}\}$$

nyelvet. Könnyen látható, hogy az $L_\emptyset$ az L_{EQ} speciális esete. Valóban, ha L_{EQ} bemenetét megszorítjuk úgy, hogy M_2 csak olyan Turing-gép lehet, ami nem fogad el egyetlen szót sem, azaz $L(M_2) = \emptyset$, akkor az $L_\emptyset$ problémát kapjuk. A fentiekből, felhasználva a 2.23. következményt, adódik az alábbi eredmény.

2.24. Következmény

$L_{\text{EQ}} \notin \mathbf{RE}$.

Az továbbiakban azt mutatjuk meg, hogy nem csak a fenti problémák eldönthetetlenek, hanem az összes Turing-gépekkel kapcsolatos nemtriviális kérdés is az.

2.5.1.3. Rice tétele

Először tisztázni kell, hogy mit nevezünk a rekurzívan felsorolható nyelvek egy *nemtriviális tulajdonságának*. Ha $\mathcal{P}$ a rekurzívan felsorolható nyelvek egy osztálya, akkor $\mathcal{P}$-t a *rekurzívan felsorolható nyelvek egy tulajdonságának* nevezzük. Azt mondjuk, hogy egy $L \in \mathbf{RE}$ nyelv rendelkezik a $\mathcal{P}$ tulajdonsággal, ha $L \in \mathcal{P}$. $\mathcal{P}$ *triviális tulajdonság*, ha minden nyelvre, vagy egyetlen nyelvre sem teljesül, így $\mathcal{P}$ egy *nemtriviális tulajdonság*, ha $\mathcal{P} \neq \emptyset$ és $\mathcal{P} \neq \mathbf{RE}$.

Például, ha $\mathcal{P}$ az üres halmaz, akkor egyetlen rekurzívan felsorolható nyelv sem rendelkezik a $\mathcal{P}$ tulajdonsággal. Viszont ha $\mathcal{P} = \{\emptyset\}$, azaz $\mathcal{P}$ az üres nyelvet tartalmazza (ami nyilván rekurzívan felsorolható nyelv), akkor kizárólag a $\emptyset$, vagyis az üres nyelv rendelkezik a $\mathcal{P}$ tulajdonsággal.

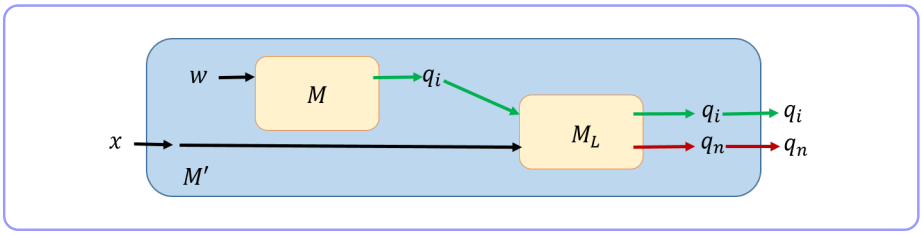
Ebben az alfejezetben megmutatjuk, hogy ha $\mathcal{P}$ nemtriviális tulajdonság, akkor nincs olyan Turing-gép, ami tetszőleges $L \in \mathbf{RE}$ nyelvről eldöntené, hogy rendelkezik-e a $\mathcal{P}$ tulajdonsággal, azaz $L \in \mathcal{P}$ teljesül-e vagy sem (megjegyezzük, hogy ha $\mathcal{P}$ egy triviális tulajdonság lenne, akkor a probléma eldöntése is triviális lenne). Igazából a következőt fogjuk bizonyítani. Adott $\mathcal{P}$ tulajdonságra jelöljük $L_{\mathcal{P}}$-vel azon Turing-gépek kódjainak a halmazát, melyek $\mathcal{P}$-beli nyelvet ismernek fel. Az alábbi tétel azt mondja ki, hogy az $L_{\mathcal{P}}$ nyelv eldönthetetlen, amennyiben $\mathcal{P}$ nemtriviális tulajdonság.

2.25. Tétel

Legyen $\mathcal{P}$ a rekurzívan felsorolható nyelvek egy nemtriviális tulajdonsága. Ekkor az $L_{\mathcal{P}}$ nyelv eldönthetetlen.

Bizonyítás. Először tegyük fel, hogy $\emptyset \notin \mathcal{P}$ (a bizonyítás végén majd visszatérünk ahhoz az esethez, amikor $\emptyset \in \mathcal{P}$). Legyen továbbá L egy tetszőleges nem üres nyelv nyelv $\mathcal{P}$-ből (ilyen L létezik, hiszen $\mathcal{P}$ nem üres, mivel nemtriviális, és $\emptyset \notin \mathcal{P}$). Legyen M_L az L-et felismerő Turing-gép.

A következőkben visszavezetjük az L_{u} nyelvet az $L_{\mathcal{P}}$-re. Legyen $\langle M, w\rangle$ egy tetszőleges Turing-gép és bemenet pár. Megmutatjuk, hogy megkonstruálható egy M' Turing-gép úgy, hogy $L(M') = \emptyset$, ha $w \notin L(M)$ és $L(M') = L$, ha $w \in L(M)$. A megkonstruált M' egy kétszalagos Turing-gép lesz, a gép vázlata



2.12. ábra. L_{u} visszavezetése $L_{\mathcal{P}}$-re

a 2.12. ábrán látható.

M' egy tetszőleges x bemeneten a következőket csinálja:

1. Függetlenül attól, hogy mi az x bemenet, M' szimulálja az egyik szalagján az M működését a w szón (M és w kódja be van építve M' kódjába; M' felmásolja a szalagjára a w-t, és meghívja az univerzális Turing-gépet, melynek segítségével szimulálja M működését w-n).

2. Ha M' azt találja, hogy M nem fogadja el w-t, akkor M' nem csinál semmit, vagyis nem fogadja el az x bemenetet. Ebben az esetben $L(M') = \emptyset$, vagyis $\langle M' \rangle \notin L_{\mathcal{P}}$.

3. Ha M' azt találja, hogy M elfogadja w-t, akkor elkezdi szimulálni M_L működését az x bemeneten. Ebben az esetben pedig $L(M) = L$, vagyis $\langle M' \rangle \in L_{\mathcal{P}}$.

Látható, hogy $\langle M, w \rangle \in L_{\mathrm{u}}$ akkor és csak akkor teljesül, ha $\langle M' \rangle \in L_{\mathcal{P}}$, vagyis a fenti eljárás L_{u} visszavezetése $L_{\mathcal{P}}$-re. Mivel L_{u} eldönthetetlen, kapjuk, hogy $L_{\mathcal{P}}$ is eldönthetetlen.

Meg kell még vizsgálni azt az esetet, amikor $\emptyset \in \mathcal{P}$. Ebben az esetben ismételjük meg a fenti bizonyítást a $\overline{\mathcal{P}} = \mathbf{RE} - \mathcal{P}$ tulajdonságra. Azt kapjuk, hogy $L_{\overline{\mathcal{P}}}$ nem eldönthető. Tegyük fel most, hogy az $L_{\mathcal{P}}$ nyelv viszont eldönthető. Korábbi tételünk alapján tudjuk, hogy ebben az esetben $\overline{L}_{\mathcal{P}}$ is eldönthető. Másrészt nem nehéz belátni, hogy $L_{\overline{\mathcal{P}}} = \overline{L}_{\mathcal{P}}$, ami azt jelenti, hogy $L_{\overline{\mathcal{P}}}$ is eldönthető, ez pedig ellentmondás, amiből következik, hogy $L_{\mathcal{P}}$ eldönthetetlen, amit bizonyítani akartunk. □

A fentiek alapján az alábbi problémák mindegyike eldönthetetlen (néhányról korábban már láttuk is, hogy az).

1. Egy tetszőleges M Turing-gép az üres nyelvet ismeri-e fel. (Ebben az esetben $\mathcal{P} = \{\emptyset\}$.)

2. Egy M Turing-gép véges nyelvet ismer-e fel ($\mathcal{P} = \{L \mid L \text{ véges}\}$).

3. Egy M Turing-gép környezetfüggetlen nyelvet ismer-e fel ($\mathcal{P} = \{L \mid L$ környezetfüggetlen$\}$).

4. Egy M Turing-gép elfogadja-e az üres szót ($\mathcal{P} = \{L \mid \varepsilon \in L\}$).

Eddig olyan eldönthetetlen problémákat láttunk, melyek Turing-gépekkel illetve az általuk felismert nyelvekkel kapcsolatosak. A következőkben megismerünk néhány olyan eldönthetetlen problémát, ami nem a Turing-gépek valamilyen tulajdonságáról szól.

2.5.2. A Post megfelelkezési probléma

Először az úgynevezett *Post megfelelkezési problémát* (röviden PMP-t) ismertetjük. Ezt a problémát szokás *dominóproblémának* is nevezni.

2.26. Definíció

A PMP-t a következőképpen definiáljuk. Legyen Σ egy legalább két betűt tartalmazó ábécé, és legyen $D = \left\{\left[\frac{u_1}{v_1}\right], \ldots, \left[\frac{u_n}{v_n}\right]\right\}$ $(n \geq 1)$ egy dominóhalmaz ahol $u_1, \ldots, u_n, v_1, \ldots, v_n \in \Sigma^+$. A kérdés az, hogy van-e egy olyan $1 \leq i_1, \ldots, i_m \leq m$ $(m \geq 1)$ indexsorozat, melyre teljesül, hogy a $\left[\frac{u_{i_1}}{v_{i_1}}\right], \ldots, \left[\frac{u_{i_m}}{v_{i_m}}\right]$ dominókat egymás mellé írva alul és felül ugyanaz a szó adódik, azaz $u_{i_1} \ldots u_{i_m} = v_{i_1} \ldots v_{i_m}$. Ebben az esetben a fenti index- vagy dominósorozatot a D *egy megoldásának* nevezzük.
Formális nyelvként a következőképpen definiálhatjuk a PMP-t:

$$\text{PMP} = \{\langle D \rangle \mid D\text{-nek van megoldása}\}.$$

2.27. Példa

- A $D = \left[\frac{a}{baa}\right]\left[\frac{ab}{aa}\right]\left[\frac{bba}{bb}\right]$ egy megoldása a $3, 2, 3, 1$ sorozat, hiszen $bba\,ab\,bba\,a = bb\,aa\,bb\,baa$.

- A $D = \left[\frac{bb}{b}\right]\left[\frac{ab}{ba}\right]\left[\frac{a}{b}\right]$ dominókészletnek nincs megoldása, mert a második és harmadik dominóval nem kezdődhet egy megoldás sem, és ha az első dominót felhasználjuk, akkor kelleni fog egy olyan dominó is, ahol fent kevesebb betű van, mint lent, de ilyen nincs D-ben.

Az alábbiakban megmutatjuk, hogy a PMP algoritmikusan eldönthetetlen. Ehhez először definiáljuk az úgynevezett *módosított Post megfelelkezési problémát* (MPMP), és megmutatjuk, hogy ez visszavezethető a PMP-re. Majd megmutatjuk, hogy az L_{u} probléma visszavezethető az MPMP-re.

Először vizsgáljuk meg tehát, hogy hogyan néz ki az MPMP probléma. Adott egy $D = \left\{ \left[\frac{u_1}{v_1}\right], \ldots, \left[\frac{u_n}{v_n}\right] \right\}$ dominókészlet valamely $n \geq 1$-re (úgy, ahogy a PMP esetében is). A kérdés az, hogy van-e egy olyan megoldása a D-nek (a PMP szerint), melyben az első dominó az $\left[\frac{u_1}{v_1}\right]$.

Először tehát megmutatjuk, hogy az MPMP visszavezethető a PMP-re.

2.28. Tétel

MPMP $\leq$ PMP.

Bizonyítás. Tekintsük az MPMP probléma egy

$$D = \left\{ \left[\frac{u_1}{v_1}\right], \ldots, \left[\frac{u_n}{v_n}\right] \right\}$$

példányát. Ehhez konstruáljuk meg a

$$D' = \left\{ \left[\frac{*u_1}{*v_1*}\right], \left[\frac{*u_1}{v_1*}\right], \left[\frac{*u_2}{v_2*}\right], \ldots, \left[\frac{*u_n}{v_n*}\right], \left[\frac{*\#}{\#}\right] \right\}$$

dominókészletet. Itt a $*$ és a $\#$ jelek új, a D dominóiban nem szereplő szimbólumok. Világos, hogy ha D-nek van megoldása (az MPMP probléma szerint), akkor D'-nek is van megoldása (a PMP szerint). Valóban, legyen az $i_1, i_2, \ldots, i_m$ ($m \geq 1$) indexsorozat a D egy megoldása. Ekkor $i_1 = 1$, hiszen a D az MPMP egy pozitív példánya. Ekkor válasszuk D'-ből először az $\left[\frac{*u_1}{*v_1*}\right]$ dominót, majd rendre minden $2 \leq j \leq m$-re válasszuk a $\left[\frac{*u_{i_j}}{v_{i_j}*}\right]$ dominókat. Végül válasszuk a $\left[\frac{*\#}{\#}\right]$ dominót. Az így kapott dominósorozat a D' egy megoldása lesz.

Tegyük fel most azt, hogy a D'-nek van egy megoldása. Nyilvánvaló, hogy ez a megoldás az $\left[\frac{*u_1}{*v_1*}\right]$ dominóval kezdődik, az $\left[\frac{*u_{i_1}}{v_{i_1}*}\right], \ldots, \left[\frac{*u_{i_m}}{v_{i_m}*}\right]$ dominókkal folytatódik (valamely $m \geq 0$-ra és $i_1, \ldots, i_m$ indexekre) és a $\left[\frac{*\#}{\#}\right]$ dominóval végződik. Ekkor viszont az $\left[\frac{u_1}{v_1}\right]\left[\frac{u_{i_1}}{v_{i_1}}\right]\ldots\left[\frac{u_{i_m}}{v_{i_m}}\right]$ dominósorozat a D egy megoldása. Ezzel a bizonyítást befejeztük. □

2.29. Tétel

PMP $\notin \mathbf{R}$.

Bizonyítás. Elég megmutatni, hogy az L_{u} visszavezethető a MPMP-re. Valóban, ekkor a 2.19. tétel alapján az MPMP is eldönthetetlen, és mivel MPMP $\leq$ PMP (2.28. tétel) a PMP is eldönthetetlen.

Legyen tehát $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ egy Turing-gép és $w \in \Sigma^*$. Megadjuk az MPMP egy D bemenetét úgy, hogy D pontosan akkor *igen* példánya az MPMP-nek, ha $\langle M, w\rangle \in L_{\mathrm{u}}$.

A 2.8. tétel alapján feltehetjük, hogy M sosem próbálja a fejet a kiindulási pozíciótól balra mozgatni. Tegyük fel továbbá, hogy $w = a_1 \dots a_n$ ($n \in [n], a_1, \dots, a_n \in \Sigma$). Ezután a D-t a következőképpen definiáljuk:

Első rész. Az első dominó amit felveszünk D-be legyen a $\left[\frac{\#}{\#q_0 a_1 \dots a_n \#}\right]$.

Második rész. Ezután, minden $a, b, c \in \Gamma$-ra, $p, q \in Q$-ra ahol $p \neq q_i$,

- ha $\delta(p, a) = (q, b, R)$, akkor a $\left[\frac{pa}{bq}\right]$ dominót,
- ha $\delta(p, a) = (q, b, L)$, akkor a $\left[\frac{cpa}{qcb}\right]$ dominót,
- ha $\delta(p, a) = (q, b, S)$, akkor a pedig a $\left[\frac{pa}{qb}\right]$ dominót vesszük fel D-be.

Harmadik rész. Minden $a \in \Gamma$-ra vegyük fel az $\left[\frac{a}{a}\right]$ dominót D-be.

Tekintsünk egy egyszerű példát a fentiek demonstrálására. Legyen M_{ex} egy Turing-gép, melynek a szalagábécéje a $\Gamma = \{a, b, \sqcup\}$, az állapothalmaza pedig a $Q = \{q_0, q_1, q_2, q_i, q_n\}$ halmaz. Tekintsük az M_{ex} egy $w = bab$ bemenetét. Ekkor, az első részben leírtak alapján a D-be bekerül a $\left[\frac{\#}{\#q_0 bab\#}\right]$ dominó.

Továbbá, ha például az M-nek van egy $\delta(q_0, b) = (q_2, a, R)$ átmenete, akkor a második rész alapján a D-be bekerül a $\left[\frac{q_0 b}{a q_2}\right]$ dominó. Végül, a harmadik rész miatt bekerülnek a D-be az $\left[\frac{a}{a}\right], \left[\frac{b}{b}\right]$ és $\left[\frac{\sqcup}{\sqcup}\right]$ dominók.

Ha M_{ex} a bemenetén a $w = bab$ szót kapja, akkor a $q_0 bab$ kezdőkonfigurációból át tud lépni az $aq_2 ab$ konfigurációba. Nézzük meg, hogy milyen D-beli dominók sorozatával szimulálható az M_{ex} ezen konfiguráció-átmenete! A cél az, hogy egy olyan dominósorozatot adjunk meg, melyben az alsó szó az M_{ex} $q_0 bab$ kezdőkonfigurációjával kezdődik és a $aq_2 ab$ konfigurációval folytatódik (és persze a konfigurációk a # jellel vannak elválasztva egymástól):

$$\left[\frac{\#}{\#q_0 bab\#}\right]\left[\frac{q_0 b}{a q_2}\right]\left[\frac{a}{a}\right]\left[\frac{b}{b}\right].$$

Látható, hogy ekkor az alsó szó pontosan az $aq_2 ab$ konfigurációval lesz hosszabb a felsőnél. Persze ahhoz, hogy tovább folytathassuk a dominósorozatunkat szükség van egy olyan dominóra, ami a felső sor végére ír egy # szimbólumot. Ezt definiáljuk a D megadásának következő részében.

Negyedik rész. Vegyük fel D-be a $\left[\frac{\#}{\#}\right]$ és a $\left[\frac{\#}{\sqcup\#}\right]$ dominókat. Ezek közül a másodikra akkor van szükség, ha M-ben előáll egy olyan konfiguráció, amikor

a fej az utolsó nem $\sqcup$ szimbólum utáni $\sqcup$-ra mutat. Ilyenkor a $\left[\frac{\#}{\sqcup\#}\right]$ dominó az alsó szó végére a # elé beír egy $\sqcup$-ot.

A fenti példánknál maradva, a következő feladatunk biztosítani azt, hogy ha az M_{ex} q_i-be lép, akkor a D-ben legyen olyan dominósorozat, melyek alkalmazásával az alsó és felső szavak egyforma hosszúak lesznek, hisz ekkor lesz csak igaz az, hogy megoldást találhatunk D-ben. Ehhez olyan dominók kellenek, melyek az alsó szóban olyan ál-konfigurációkat hoznak létre, melyekben a q_i állapot körül „eltűnnek" a nem $\sqcup$ szimbólumok.

Ötödik rész. Vegyük fel a D-be minden $a \in \Gamma$-ra a $\left[\frac{aq_i}{q_i}\right]$ és a $\left[\frac{q_i a}{a}\right]$ dominókat.

Ezen rész szemléltetéséhez tegyük fel, hogy M_{ex}-nek van egy $\delta(q_2, a) = (q_i, b, S)$ átmenete. Ekkor a harmadik részben látott dominósorozat folytatható a következőképpen:

$$\left[\frac{\#}{\#q_0bab\#}\right]\left[\frac{q_0b}{aq_2}\right]\left[\frac{a}{a}\right]\left[\frac{b}{b}\right]\left[\frac{\#}{\#}\right]\left[\frac{a}{a}\right]\left[\frac{q_2a}{q_ib}\right]\left[\frac{b}{b}\right]\left[\frac{\#}{\#}\right].$$

A könnyebb olvashatóság kedvéért nézzük meg, hogyan néz ki most a fenti sorozat felső illetve alsó szava:

$$\frac{\#q_0bab\#aq_2ab\#}{\#q_0bab\#aq_2ab\#aq_ibb\#}.$$

Az ötödik részben definiált dominók segítségével „eltüntethetjük" az alsó szó utolsó két #-ja közti q_i-től különböző szimbólumokat. Valóban, folytassuk a fenti dominósorozatot az alábbi dominókkal:

$$\left[\frac{aq_i}{q_i}\right]\left[\frac{b}{b}\right]\left[\frac{b}{b}\right]\left[\frac{\#}{\#}\right].$$

Ekkor a dominósorozat szavai a következőképpen néznek ki:

$$\frac{\#q_0bab\#aq_2ab\#aq_ibb\#}{\#q_0bab\#aq_2ab\#aq_ibb\#q_ibb\#}.$$

Folytatva a fenti dominósorozatot a

$$\left[\frac{q_ib}{q_i}\right]\left[\frac{b}{b}\right]\left[\frac{\#}{\#}\right]\left[\frac{q_ib}{q_i}\right]\left[\frac{\#}{\#}\right]$$

dominókkal, azt kapjuk, hogy a teljes dominósorozat felső és alsó szavai a következőképpen néznek ki:

$$\frac{\#q_0bab\#aq_2ab\#aq_ibb\#q_ibb\#q_ib\#}{\#q_0bab\#aq_2ab\#aq_ibb\#q_ibb\#q_ib\#q_i\#}.$$

Hatodik rész. Ahogy a fenti példából is látszik, ahhoz, hogy a D-nek legyen egy megoldása, elegendő a $\left[\frac{q_i\#\#}{\#}\right]$ dominót felvenni a D-be.

A fentiek alapján elmondható, hogy ha $w \in L(M)$, azaz $\langle M, w\rangle \in L_{\mathrm{u}}$, akkor a D-nek van megoldása. Másrészt, ha D-nek van megoldása, akkor mivel a D az MPMP egy pozitív példánya, ez a megoldás biztos, hogy a $\left[\frac{\#}{\#q_0a_1\ldots a_n\#}\right]$ dominóval kezdődik. Ez viszont azt jelenti, hogy a D ezen megoldása szimulálja az M egy elfogadó számítási sorozatát a w-n. Kapjuk tehát, hogy $w \in L(M)$, vagyis $\langle M, w\rangle \in L$.

Így a D konstrukciója L_{u} visszavezetése az MPMP-re, és ezzel a bizonyítást befejeztük. □

2.5.3. Környezetfüggetlen grammatikákkal kapcsolatos eldönthetetlen problémák

2.5.3.1. Környezetfüggetlen grammatikák egyértelműsége

A PMP segítségével megmutatjuk, hogy eldönthetetlen, vajon egy környezetfüggetlen grammatika egyértelmű-e. Emlékeztetőül, egy G CF grammatika egyértelmű, ha minden $L(G)$-beli szónak pontosan egy baloldali levezetése van G-ben.

2.30. Tétel

A környezetfüggetlen grammatikák egyértelműsége eldönthetetlen.

Bizonyítás. Megmutatjuk, hogy PMP visszavezethető a tételben szereplő probléma komplementerére. Ez a 2.29, 2.19 és 2.17. tételek segítségével bizonyítja az állításunkat.

Legyen $D = \left\{\left[\frac{u_1}{v_1}\right], \ldots, \left[\frac{u_n}{v_n}\right]\right\}$ $(n \geq 1)$ a PMP egy bemenete, melyben a dominók szavai egy Σ ábécé felettiek. Konstruáljuk meg a G CF grammatikát a következőképpen. Először definiálunk két további CF grammatikát. Legyen $\Delta = \{a_1, \ldots, a_n\}$ úgy, hogy $\Delta \cap \Sigma = \emptyset$. Legyen továbbá $G_A = (\{A\}, \Sigma \cup \Delta, P_A, A)$ és $G_B = (\{B\}, \Sigma \cup \Delta, P_B, B)$, ahol

$$P_A = \{A \to u_1Aa_1, \ldots, A \to u_nAa_n, A \to u_1a_1, \ldots, A \to u_na_n\} \text{ és}$$

$$P_B = \{B \to v_1Ba_1, \ldots, B \to v_nBa_n, B \to v_1a_1, \ldots, B \to v_na_n\}.$$

Végül, legyen $G = (\{S, A, B\}, \Sigma \cup \Delta, P_A \cup P_B \cup \{S \to A, S \to B\}, S)$ ahol S egy új nemterminális szimbólum.

Könnyen látható, hogy G-ben minden levezetés baloldali. Ezért G akkor és csak akkor nem egyértelmű, ha van egy olyan w szó a $\Sigma \cup \Delta$ felett, hogy w levezethető G-ben két különböző módon. De ez, mivel G-ben a G_A és G_B szabályi nem „keverednek", csak úgy lehetséges, hogy w levezethető G_A-ban és G_B-ben is. Megjegyezzük, hogy G_A-ban csak $u_{i_1} \ldots u_{i_m} a_{i_m} \ldots a_{i_1}$, G_B-ben

pedig $v_{i_1} \dots v_{i_m} a_{i_m} \dots a_{i_1}$ alakú szavak vezethetők le ($m \geq 1, 1 \leq i_1, \dots, i_m \leq n$).

Elég tehát belátnunk azt, hogy D-nek akkor és csak akkor megoldása az $\left[\frac{u_{i_1}}{v_{i_1}}\right]$, $\dots$, $\left[\frac{u_{i_m}}{v_{i_m}}\right]$ ($m \geq 1$) dominó sorozat, ha a $w = u_{i_1} \dots u_{i_m} a_{i_m} \dots a_{i_1}$ szó levezethető G_A-ban és G_B-ben is.

Először tegyük fel, hogy D-nek van egy $\left[\frac{u_{i_1}}{v_{i_1}}\right], \dots, \left[\frac{u_{i_m}}{v_{i_m}}\right]$ ($m \geq 1$) megoldása. Ekkor G_A definíciója alapján, az $u_{i_1} \dots u_{i_m} a_{i_m} \dots a_{i_1}$ szó levezethető G_A-ban rendre az $A \to u_{i_1} A a_{i_1}, \dots, A \to u_{i_{m-1}} A a_{i_{m-1}}, A \to u_{i_m} a_{i_m}$ szabályok alkalmazásával. Hasonlóan, a G_B-ben is levezethető a $v_{i_1} \dots v_{i_m} a_{i_m} \dots a_{i_1}$ szó. Mivel a fenti dominósorozat megoldása a D-nek, kapjuk, hogy $u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$, és így a bizonyítás egyik részével készen vagyunk, az $u_{i_1} \dots u_{i_m} a_{i_m} \dots a_{i_1}$ szó levezethető G_A-ban és G_B-ben is.

Most tegyük fel, hogy egy $w = u a_{i_m} \dots a_{i_1}$ ($u \in \Sigma^*$) alakú szó levezethető G-ben az A-ból és a B-ből is. Ekkor a w-beli $a_{i_1} \dots a_{i_k}$ részszó miatt a w levezetésében mindkét grammatika ugyanolyan sorrendben használja a szabályait. Nevezetesen, G_A-ban pontosan akkor kapható meg a w az $A \Rightarrow u_{i_1} A a_i \Rightarrow \dots \Rightarrow u_{i_1} \dots u_{i_{m-1}} A a_{i_{m-1}} \dots a_{i_1} \Rightarrow u_{i_1} \dots u_{i_m} a_{i_m} \dots a_{i_1} = w$ levezetéssel, amikor a G_B-ben megkapható a w a $B \Rightarrow v_{i_1} B a_{i_1} \Rightarrow \dots \Rightarrow v_{i_1} \dots v_{i_{m-1}} B a_{i_{m-1}} \dots a_{i_1} \Rightarrow v_{i_1} \dots v_{i_m} a_{i_m} \dots a_{i_1} = w$ levezetéssel. Tehát $u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$, azaz az $\left[\frac{u_{i_1}}{v_{i_1}}\right], \dots, \left[\frac{u_{i_m}}{v_{i_m}}\right]$ dominósorozat D egy megoldása. Ezzel a bizonyítás végére értünk. □

2.5.3.2. További eldönthetetlen kérdések

Legyen D a PMP egy példánya. A 2.30. tétel bizonyításban szereplő G_A és G_B grammatikákat, illetve az általuk felismert $L_A = L(G_A)$ és $L_B = L(G_B)$ nyelveket fogjuk felhasználni arra, hogy az alábbi tételben megadott problémák eldönthetetlenségét bizonyítsuk.

Szükségünk lesz továbbá arra hogy $\overline{L}_A$ és $\overline{L}_B$ is CF nyelvek. Mivel a CF nyelvek nem zártak a komplementer képzésre, a fenti állítás bizonyításra szorul, de ezt nem közöljük, mivel túlmutat a jegyzet határain.

2.31. Tétel

Legyen G_1 és G_2 két környezetfüggetlen grammatika. Ekkor az alábbi problémák mindegyike eldönthetetlen:

1. $L(G_1) \cap L(G_2) = \emptyset$?
2. $L(G_1) = L(G_2)$?
3. $L(G_1) = \Gamma^*$, valamely Γ ábécére?

4. $L(G_1) \subseteq L(G_2)$?

Bizonyítás. Legyen D egy dominókészlet valamely Σ ábécé felett, és legyen Δ a 2.30. tétel bizonyításában látott ábécé, L_A és L_B pedig a fent definiált nyelvek. Világos, hogy D-nek akkor és csak akkor van megoldása, ha $L_A \cap L_B$ nem üres. Ebből a 2.17. tételt felhasználva azt kapjuk, hogy se az $L_A \cap L_B = \emptyset$, se az $L_A \cap L_B \neq \emptyset$ kérdés nem dönthető el.

1. A fentiek alapján a probléma eldönthetetlensége azonnal következik, ha G_1-et G_A-nak, G_2-t pedig G_B-nek választjuk.

2. Korábbi megjegyzésünk alapján $\overline{L}_A$ és $\overline{L}_B$ is CF nyelvek. Ismert, hogy a CF nyelvek zártak az unió műveletére, tehát $\overline{L}_A \cup \overline{L}_B$ is CF nyelv. Azt is tudjuk, hogy $\overline{L}_A \cup \overline{L}_B = \overline{L_A \cap L_B}$. Legyen G_1 és G_2 két olyan CF grammatika, hogy $L(G_1) = \overline{L_A \cap L_B}$ és $L(G_2) = (\Sigma \cup \Delta)^*$. Ekkor $L(G_1) = L(G_2)$ azt jelenti, hogy $\overline{L_A \cap L_B} = (\Sigma \cup \Delta)^*$, azaz $L_A \cap L_B = \emptyset$. Mivel az utóbbi egyenlőség eldönthetetlen, kapjuk, hogy $L(G_1) = L(G_2)$ is eldönthetetlen.

3. Ha ez a probléma eldönthető lenne, akkor G_1-et úgy választva, mint 2-ben, Γ-nak pedig $\Sigma \cup \Delta$-t választva, el lehetne dönteni, hogy D-nek van-e megoldása, ami ellentmondás.

4. Ennek a problémának az eldönthetetlensége szintén a 2. probléma eldönthetetlenségének következménye, hiszen tetszőleges G_1 és G_2 CF grammatikákra, $L(G_1) = L(G_2)$ akkor és csak akkor teljesül, ha $L(G_1) \subseteq L(G_2)$ és $L(G_2) \subseteq L(G_1)$. Tehát, ha a tartalmazás eldönthető lenne, akkor az ekvivalenciát is el lehetne dönteni, ami ellentmondáshoz vezetne.

Ezzel a tétel bizonyítását befejeztük. □

2.5.4. Eldönthetetlenség az elsőrendű logikában

Világos, hogy az ítéletkalkulusban eldönthető az, hogy egy φ formula kielégíthető, kielégíthetetlen vagy érvényes-e. Ehhez nem kell mást tenni, csak véges sok interpretációban ki kell számítani a φ formula értékét. Az elsőrendű logikában viszont, mivel megszámlálhatatlanul sok interpretációt lehet megadni egy adott formulához, már nem járhatunk el így. Sőt, amint azt látni fogjuk, az elsőrendű logikában a fenti kérdések mind eldönthetetlenek. Először az érvényesség eldönthetetlenségét mutatjuk meg.

Legyen ValidityPred a következő probléma:

$$\textsc{ValidityPred} = \{\langle\varphi\rangle \mid \varphi \text{ érvényes elsőrendű logikai formula}\}.$$

2.32. Tétel

ValidityPred $\notin \mathbf{R}$.

Bizonyítás. Megmutatjuk, hogy PMP $\leq$ ValidityPred. Legyen $D = \{\left[\frac{u_1}{v_1}\right], \ldots, \left[\frac{u_k}{v_k}\right]\}$ $(k \geq 1)$ egy $\Sigma = \{a_1, \ldots, a_n\}$ $(n \geq 2)$ ábécé feletti dominókészlet. Tekintsük azt az elsőrendű nyelvet, melyben $Pred = \{p\}$ $(ar(p) = 2)$, $Func = \{f_{a_1}, \ldots, f_{a_n}\}$ $(ar(f_{a_i}) = 1$ minden $i \in [n]$-re) és $Const = \{c\}$. Megadunk egy olyan φ_D formulát, hogy D-nek pontosan akkor van megoldása, ha φ_D érvényes formula. Az könnyebb olvashatóság kedvéért, egy $f_{a_{i_1}}(f_{a_{i_2}}(\ldots(f_{a_{i_m}}(t))))$ $(m \geq 1)$ alakú term helyett gyakran $f_{a_{i_1}\ldots a_{i_m}}(t)$-t írunk. Legyen $\varphi_D = (\varphi_1 \wedge \varphi_2) \to \varphi_3$, ahol

- $\varphi_1 = p(f_{u_1}(c), f_{v_1}(c)) \wedge \ldots \wedge p(f_{u_k}(c), f_{v_k}(c))$,
- $\varphi_2 = \forall x \forall y (p(x, y) \to (p(f_{u_1}(x), f_{v_1}(y)) \wedge \ldots \wedge p(f_{u_k}(x), f_{v_k}(y))))$,
- $\varphi_3 = \exists z p(z, z)$.

Először tegyük fel, hogy φ_D tautológia. Legyen I a következő interpretáció. I alaphalmaza Σ^*, $f^I_{a_i}(u) = a_i u$ $(i \in [k], u \in \Sigma^*)$, $c^I = \varepsilon$, a p interpretációja pedig az alábbi. Tetszőleges $u, v \in \Sigma^*$ esetén $p^I(u, v) = igaz$ akkor és csak akkor, ha a következő (i)-vel jelölt feltétel teljesül: van olyan $m \geq 1$ és $i_1, \ldots, i_m \in [k]$, hogy $u_{i_1} \ldots u_{i_m} = u$ és $v_{i_1} \ldots v_{i_m} = v$.

Először belátjuk, hogy $I \models \varphi_1 \wedge \varphi_2$. Mivel minden $i \in [k]$-ra $|f_{u_i}(c)|^I = u_i$, $|f_{v_i}(c)|^I = v_i$ és $p^I(u_i, v_i) = igaz$, kapjuk, hogy $I \models \varphi_1$. Ahhoz, hogy $I \models \varphi_2$-t belássuk elég bizonyítani, hogy tetszőleges $u, v \in \Sigma^*$ esetén teljesül a következő állítás: (ii) ha $p^I(u, v) = igaz$, akkor minden $i \in [k]$-ra $p^I(f_{u_i}(u), f_{v_i}(v)) = igaz$. Tegyük fel, hogy $p^I(u, v) = igaz$ valamely $u, v \in \Sigma^*$-ra. Akkor (i) miatt van olyan $m \geq 1$ és $i_1, \ldots, i_m \in [k]$, hogy $u = u_{i_1} \ldots u_{i_m}$ és $v = v_{i_1} \ldots v_{i_m}$. Világos, hogy ekkor minden $i \in [k]$ esetén az $u_i u$ és $v_i v$ szavakra is teljesül az (i) feltétel. Kapjuk tehát, hogy minden $i \in [k]$-ra, $p^I(f_{u_i}(u), f_{v_i}(v)) = igaz$, azaz (ii) teljesül.

Beláttuk tehát, hogy $I \models \varphi_1 \wedge \varphi_2$. Ekkor viszont, mivel $I \models \varphi_D$ kapjuk, hogy $I \models \varphi_3$. Ez pedig pontosan azt jelenti, hogy D-nek van megoldása.

Tegyük fel most, hogy D-nek van megoldása, és legyen $I = \langle U, I_{Pred}, I_{Func}, I_{Const}\rangle$ egy tetszőleges interpretáció. Megmutatjuk, hogy $I \models \varphi_D$. Ha $I \not\models \varphi_1 \wedge \varphi_2$, akkor $I \models \varphi_D$ azonnal adódik. Tegyük fel, hogy $I \models \varphi_1 \wedge \varphi_2$. Legyen $\left[\frac{u_{i_1}}{v_{i_1}}\right] \ldots \left[\frac{u_{i_m}}{v_{i_m}}\right]$ a D egy megoldása. Mivel $I \models \varphi_1$, $I \models p(f_{u_{i_m}}(c), f_{v_{i_m}}(c))$. De ekkor, mivel $I \models \varphi_2$, azt kapjuk, hogy $I \models p(f_{u_{i_{m-1}} u_{i_m}}(c), f_{v_{i_{m-1}} v_{i_m}}(c))$. Ezt a gondolatmenetet folytatva kapjuk, hogy $I \models p(f_{u_{i_1}\ldots u_{i_m}}(c), f_{v_{i_1}\ldots v_{i_m}}(c))$.

Ekkor, mivel $u_{i_1} \dots u_{i_m} = v_{i_1} \dots v_{i_m}$, $|f_{u_{i_1} \dots u_{i_m}}(c)|^I = |f_{v_{i_1} \dots v_{i_m}}(c)|^I$. Tehát van olyan U-beli elem, például az $u = |f_{u_{i_1} \dots u_{i_m}}(c)|^I$, hogy $I \models p(u, u)$. Következik, hogy $I \models \varphi_3$ és ezért $I \models \varphi_D$. Ezzel a tételt bebizonyítottuk. □

A fenti eredményből az 1.1. tétel alkalmazásával azonnal adódik az alábbi eredmény.

2.33. Következmény

Legyen F elsőrendű logikai formulák tetszőleges halmaza és legyen φ egy további ilyen formula. Az alábbi kérdések eldönthetetlenek.

1. Vajon kielégíthetetlen-e φ?
2. Vajon kielégíthető-e φ?
3. Vajon teljesül-e, hogy $F \models \varphi$?

Ismert, hogy van olyan algoritmus, ami egy tetszőleges φ elsőrendű logikai formulára pontosan akkor áll meg *igen* válasszal, ha φ kielégíthetetlen (ilyen például a elsőrendű logika rezolúciós algoritmusa). Ezért a kielégíthetetlenség eldöntése **RE**-beli probléma. Mivel a kielégíthetőség illetve kielégíthetetlenség eldöntése egymás komplementerei, a fentiekből és a 2.15. tételből következik, hogy olyan algoritmus viszont nem létezhet, ami tetszőleges φ esetén akkor áll meg *igen* válasszal, ha φ kielégíthető. Ezért a kielégíthetőség eldöntése még csak nem is **RE**-beli probléma.

3. fejezet

Bevezetés a bonyolultságelméletbe

Amint azt a jegyzet bevezetőjében említettük, a bonyolultságelmélet célja az eldönthető problémák osztályozása a megoldáshoz szükséges erőforrások (jellemzően az idő és a tár) mennyisége szerint. Először az időbonyolultsággal foglalkozunk. Szükségünk lesz az alábbi definíciókra.

3.1. Definíció

Legyen $f : \mathbb{N} \to \mathbb{N}$ egy függvény. Ekkor

$$\mathbf{TIME}(f(n)) = \{L \mid L \text{ eldönthető } \mathcal{O}(f(n)) \text{ időigényű Turing-géppel}\}.$$

Továbbá $\mathbf{P} = \bigcup_{k \geq 1} \mathbf{TIME}(n^k)$.

Tehát **P** azon nyelveket tartalmazza, melyek eldönthetők polinom időkorlátos determinisztikus Turing-géppel. Ilyen például a jól ismert Elérhetőség probléma, melynek bemenete egy G irányított gráf és annak két kitüntetett csúcsa, s és t. A kérdés az, hogy van-e a G-ben s-ből t-be vezető út. Az Elérhetőség problémát formális nyelvként a következőképpen definiáljuk.

$$\text{Elérhetőség} = \{\langle G, s, t\rangle \mid G \text{ irányított gráf és } G\text{-ben elérhető } s\text{-ből } t\},$$

ahol a $\langle G, s, t\rangle$ jelölés, amint azt már megszokhattuk, a G, s, t egy megfelelő kódolása valamilyen ábécé feletti szóval. Könnyen megadható az Elérhetőség problémát eldöntő M Turing-gép: M a szélességi keresés algoritmusát implementálva felépíti az s-ből elérhető csúcsok H halmazát, és pontosan akkor áll meg q_i-ben, ha $t \in H$. Mivel a szélességi keresés algoritmusa a gráf méretében

polinom időigényű, és M megadható úgy, hogy hatékonyan implementálja ezt az algoritmus kapjuk, hogy ELÉRHETŐSÉG $\in \mathbf{P}$.

Ezután a fent definiált problémaosztályok nemdeterminisztikus megfelelőit definiáljuk.

3.2. Definíció

Ha $f : \mathbb{N} \to \mathbb{N}$ egy függvény, akkor

$$\mathbf{NTIME}(f(n)) = \{L \mid L \text{ eldönthető } \mathcal{O}(f(n)) \text{ időigényű } \mathbf{nemdeterminisztikus} \text{ Turing-géppel}\}.$$

Továbbá $\mathbf{NP} = \bigcup_{k \geq 1} \mathbf{NTIME}(n^k)$.

Az **NP**-beli problémák rendelkeznek egy közös tulajdonsággal: ha tekintjük egy **NP**-beli probléma egy bemenetét és egy lehetséges tömör „bizonyítékot" arra nézve, hogy ez a bemenet pozitív bemenete az adott problémának, akkor ezen bizonyíték helyességének leellenőrzése már polinom időben elvégezhető. Az, hogy a bizonyíték „tömör" azt jelenti, hogy megadható a problémához egy $p(n)$ polinom úgy, hogy ha a bemenet mérete n, akkor a bizonyíték mérete $\mathcal{O}(p(n))$.

Ennek megfelelően egy **NP**-beli problémát eldöntő polinom időigényű nemdeterminisztikus Turing-gép működhet úgy, hogy nemdeterminisztikusan „megsejti" a probléma bemenetének egy tömör megoldását (pozitív voltának bizonyítékát), és polinom időben leellenőrzi, hogy a megoldás helyes-e.

Tekintsük például a korábban már említett SAT problémát. Tudjuk, hogy SAT pozitív bemenetei azon ítéletkalkulusbeli konjunktív normálformák, melyek kielégíthetőek. Legyen φ egy tetszőleges ítéletkalkulusbeli KNF. Annak, hogy φ kielégíthető tömör bizonyítéka egy olyan interpretáció, ami mellett kiértékelve a φ-t igaz értéket kapunk. Egy tetszőleges interpretáció tehát a φ kielégíthetőségének egy lehetséges bizonyítéka. Annak ellenőrzése pedig, hogy ez az interpretáció tényleg igazzá teszi-e φ-t, polinom időben elvégezhető. Ennek megfelelően, a SAT eldönthető egy olyan nemdeterminisztikus Turing-géppel, mely nemdeterminisztikusan megsejt egy interpretációt, azaz végigolvassa a formulát balról jobbra és minden új ítéletváltozó esetén felírja egy szalagjára ezt a változót és annak egy lehetséges igazságértékét (itt jelenik meg a nemdeterminisztikusság a Turing-gép működésében). A gép ezután polinom időben ellenőrzi, hogy a kapott igazságértékelés kielégíti-e a bemenetet. Következésképpen a SAT **NP**-beli probléma.

A SAT fenti tulajdonságára szokás úgy is hivatkozni, hogy a SAT polinom időben ellenőrizhető. Általánosan, egy L nyelv *polinom időben ellenőrizhető*, ha van olyan $L_V \in \mathbf{P}$ nyelv és $k \geq 1$ szám, hogy

$$L = \{u \mid \exists v : (u, v) \in L_V \text{ és } l(v) = \mathcal{O}(l(u)^k)\}.$$

A polinom idejű ellenőrizhetőség definícióját felhasználva adódik az **NP** osztály alternatív definíciója:

$$\mathbf{NP} = \{L \mid L \text{ polinom időben ellenőrizhető}\}.$$

Ahogy arról fentebb már volt szó, a nemdeterminisztikus polinom időigényű Turing-gépek rendelkeznek azzal a nem realisztikus tulajdonsággal, hogy képesek „megsejteni" a probléma bemenetének egy tömör megoldását. Egy ilyen gépet azonban elképzelhetünk úgy is, mint egy olyat, ami ha egy adott konfigurációban az átmenetfüggvénye alapján $n \geq 2$ következő konfigurációba tud átmenni, akkor a soron következő lépése során a következőt teszi. Először készít a szóban forgó konfigurációból n identikus másolatot, majd ezeken rendre alkalmazza a nemdeterminisztikus átmeneteinek egyikét. Ezen felfogás szerint egy ilyen gép tekinthető egy olyan eszköznek, ami képes párhuzamos számítások végrehajtására: adott bemeneten az aktuális lépése során kiszámítja az összes lehetséges számítási sorozatának a következő konfigurációját. Ez a felfogás összhangban van a nemdeterminisztikus Turing-gép 37. oldalon látott időigény definíciójával is, miszerint egy ilyen gép időigénye egy u bemeneten az u-n felépített számítási fa magasságával egyezik meg.

Könnyen látható viszont, hogy egy számítási fának az n-ik szintjén n-ben már exponenciálisan sok konfiguráció szerepelhet. Tehát, ha a nemdeterminisztikus Turing-gépre úgy tekintünk, mint egy olyan eszközre ami egyetlen lépése során képes az exponenciálisan sok lehetséges számításának a következő konfigurációját kiszámítani, akkor egy nem realisztikus számítási modellt kapunk. Valóban, jelenlegi ismereteink szerint nincs olyan számítástechnikai eszköz, ami az időigény nagyságrendi romlása nélkül képes lenne szimulálni egy ilyen Turing-gépet.

Ezen felfogás szerint egy a SAT-ot polinomidőben eldöntő nemdeterminisztikus Turing-gép a következőképpen működik. A gép minden olyan esetben amikor egy újabb x ítéletváltozót olvas a bemeneten, készít két identikus példányt az aktuális konfigurációjából és az egyik konfiguráció-példányon úgy folytatja a számítást, hogy x értékét *igaz*nak veszi, a másikon pedig *hamis*nak.

A SAT probléma speciális az **NP**-beli problémák között abban az értelemben, hogy csak nemdeterminisztikus Turing-géppel tudjuk hatékonyan megoldani, determinisztikussal, jelenlegi ismereteink szerint, nem. Másrészt viszont nincs bizonyítékunk arra, hogy nem is lehet determinisztikus Turing-géppel hatékonyan megoldani. Tapasztalataink alapján *erős a sejtésünk*, hogy a SAT olyan **NP**-beli probléma, ami nem **P**-beli. Az viszont egy ismert eredmény, hogy ha sikerülne a SAT eldöntésére polinomiális időigényű algoritmust találni (ez a fenti sejtésünk alapján igen valószínűtlen), akkor ez azt jelentené, hogy az **NP** osztály megegyezik a **P** osztállyal. A SAT-ot és a vele megegyező nehézségű problémákat nevezzük majd később **NP**-teljes problémáknak.

3.1. NP-teljes problémák

Ahhoz, hogy az **NP** osztály szerkezetét vizsgálni tudjuk, szükségünk lesz a visszavezetések egy speciális osztályára. Ezek a polinom idejű visszavezetések. Később más típusú visszavezetéseket is fogunk használni, ezért a kiszámítható függvények egy tetszőleges osztályát alkalmazó visszavezetéseket az alábbi módon definiáljuk.

3.3. Definíció

Legyenek $L_1 \in \Sigma^*$ és $L_2 \in \Delta^*$ nyelvek és, legyen v kiszámítható függvények egy osztálya. Azt mondjuk, hogy L_1 visszavezethető v szerint az L_2-re (jele: $L_1 \leq_v L_2$), ha $L_1 \leq L_2$ és az $L_1 \leq L_2$ visszavezetésben alkalmazott f függvény v-ben van. Ha v a polinom időben kiszámítható függvények osztálya, akkor azt mondjuk, hogy L_1 *polinom időben visszavezethető* L_2-re, és ekkor az $L_1 \leq_v L_2$ helyett az $L_1 \leq_p L_2$ jelölést fogjuk használni.
Legyen **C** problémák, v pedig kiszámítható függvények egy osztálya. Azt mondjuk, hogy **C** *zárt a* v*-beli visszavezetésekre nézve*, ha a következő teljesül. Tetszőleges L_1, L_2 problémák esetén, ha $L_1 \leq_v L_2$ és $L_2 \in \mathbf{C}$, akkor $L_1 \in \mathbf{C}$.

3.4. Tétel

P és **NP** zártak a polinom idejű visszavezetésekre nézve.

Bizonyítás. Legyen L_1 és L_2 két nyelv úgy, hogy $L_1 \leq_p L_2$. Tegyük fel először azt, hogy $L_2 \in \mathbf{NP}$. Megmutatjuk, hogy $L_1 \in \mathbf{NP}$ is teljesül. Legyen M egy polinom idejű Turing-gép, ami az $L_1 \leq_p L_2$ visszavezetést számolja ki, M_2 pedig egy polinom idejű nemdeterminisztikus gép, ami eldönti L_2-t. Tegyük fel, hogy M és M_2 időigénye rendre $p_1(n)$ és $p_2(n)$. Konstruáljunk meg ezekből a Turing-gépekből egy M_1 Turing-gépet a 2.11. ábrán látható módon: M_1 egy w bemenetre számolja ki az $f(w)$ szót (azaz szimulálja M-et), majd a kapott $f(w)$ szóra hívja meg M_2-t. Ha M_2 elfogadja $f(w)$-t, akkor M_1 is fogadja el w-t, ha M_2 elutasítja $f(w)$-t, akkor M_1 is utasítsa el w-t. Könnyű belátni, hogy az így vázolt M_1 egy olyan nemdeterminisztikus Turing-gép, ami L_1-et dönti el. Be kell még látnunk, hogy M_1 polinom időigényű. Világos, hogy ha w hossza n, akkor $f(w)$ hossza legfeljebb $p_1(n)$. Azaz az M_2 legfeljebb $p_2(p_1(n))$ lépésben megáll. Mivel a p_1 és p_2 kompozíciója maga is polinom függvény azt kapjuk, hogy M_1 polinom időigényű. Ebből következik, hogy $L_1 \in \mathbf{NP}$ is teljesül.

A **P** zártságának belátásához tegyük fel, hogy $L_2 \in \mathbf{P}$. Ekkor M_2 választható determinisztikus Turing-gépnek. Ez esetben viszont a fenti konstrukció egy

L_1-et eldöntő determinisztikus M_1 Turing-gépet eredményez, azaz ekkor $L_1 \in \mathbf{P}$. □

Most definiáljuk **NP** egy fontos részosztályát.

3.5. Definíció

Legyen L egy probléma. Azt mondjuk, hogy L **NP**-teljes, ha

1. **NP**-beli és
2. minden további **NP**-beli probléma polinom időben visszavezethető L-re.

Ha csak a második pont teljesül, akkor azt mondjuk, hogy L **NP**-nehéz.

Most megmutatjuk, hogy ha egy **NP**-teljes probléma eleme **P**-nek, akkor $\mathbf{P} = \mathbf{NP}$.

3.6. Tétel

Legyen L egy **NP**-teljes probléma. Ha $L \in \mathbf{P}$, akkor $\mathbf{P} = \mathbf{NP}$.

Bizonyítás. Tegyük fel, hogy L egy **P**-beli **NP**-teljes probléma. Legyen $L' \in \mathbf{NP}$ egy tetszőleges nyelv. Mivel L **NP**-teljes, $L' \leq_p L$. Ekkor viszont, mivel **P** zárt a polinom idejű visszavezetésekre nézve (3.4. tétel), $L' \in \mathbf{P}$ is teljesül.

Azt kaptuk tehát, hogy $\mathbf{NP} \subseteq \mathbf{P}$. Mivel $\mathbf{P} \subseteq \mathbf{NP}$ triviálisan teljesül, $\mathbf{P} = \mathbf{NP}$. □

Ahogy azt az **NP**-teljes problémák definíciójában láttuk, ha meg akarjuk mutatni, hogy egy **NP**-beli probléma **NP**-teljes, akkor meg kell mutatni, hogy minden **NP**-beli probléma visszavezethető rá. Ez általában, mint ahogy azt a SAT esetében hamarosan látni is fogjuk, nehéz feladat. Az alábbi tételt alkalmazva viszont, ha már egy probléma **NP**-teljességét bizonyítottuk, segítségével további **NP**-beli problémákról láthatjuk be, hogy **NP**-teljesek.

3.7. Tétel

Legyen L_1 egy **NP**-teljes, L_2 pedig egy **NP**-beli probléma. Ha $L_1 \leq_p L_2$, akkor L_2 is **NP**-teljes.

Bizonyítás. Legyen L egy tetszőleges **NP**-beli probléma. Mivel L_1 **NP**-teljes, $L \leq_p L_1$. Legyen M_1 az $L \leq_p L_1$ visszavezetést, M_2 pedig az $L_1 \leq_p L_2$ visszavezetést kiszámító polinom időigényű Turing-gép. M_1-ből és M_2-ből könnyen megadható egy olyan M Turing-gép, ami az $L \leq_p L_2$ visszavezetést számolja

ki. Továbbá, alkalmazva a 3.4. tétel bizonyításában használt gondolatmenetet, feltehetjük, hogy M polinom időigényű. Mivel L tetszőleges **NP**-beli nyelv volt kapjuk, hogy minden **NP**-beli nyelv visszavezethető polinom időben L_2-re. Így, mivel L_2 **NP**-beli következik, hogy L_2 is **NP**-teljes. □

3.1.1. A SAT probléma

Ebben a részben megmutatjuk a bonyolultságelmélet egyik fontos eredményét, nevezetesen, hogy SAT **NP**-teljes. Emlékeztetőül, SAT azon konjunktív normálformákat tartalmazza, egy megfelelő ábécé felett kódolva, melyek kielégíthetőek. Például $\langle (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \rangle \in$ SAT.

3.8. Tétel (Cook–Levin tétele)

SAT **NP**-teljes.

Bizonyítás. A fejezet elején már vázoltuk, hogy hogyan lehet megoldani a SAT-ot egy polinom idejű nemdeterminisztikus Turing-géppel. Tehát SAT **NP**-beli. Amit be kell még bizonyítani az az, hogy minden **NP**-beli nyelv polinom időben visszavezethető SAT-ra. Legyen L egy tetszőleges **NP**-beli nyelv, és legyen $M = (Q, \Sigma, \Gamma, \delta, q_0, q_i, q_n)$ egy $p(n)$ időigényű nemdeterminisztikus Turing-gép valamely $p(n)$ polinomra úgy, hogy $L = L(M)$. Feltehető, hogy minden $n \in \mathbb{N}$-re, $p(n) \geq n$.

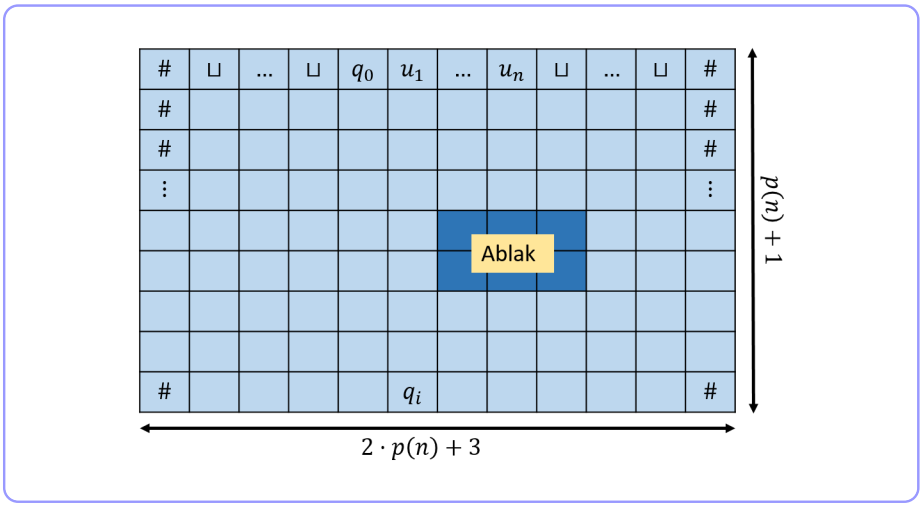
A feladatunk az, hogy M minden w bemenő szavához elkészítsünk egy φ formulát úgy, hogy:

- $w \in L$ akkor és csak akkor, ha φ kielégíthető,
- a $w \mapsto \langle \varphi \rangle$ hozzárendelés megvalósítható polinom időkorlátos deterministikus Turing-géppel.

M működése w szón leírható egy olyan táblázattal, melynek első sora M kezdőkonfigurációjának, két egymást követő sora pedig M két egymást követő konfigurációjának felel meg a w-n. Egy ilyen táblázat a 3.1. ábrán látható.

Tegyük fel, hogy $w = u_1 u_2 \dots u_n$ valamely $n \in \mathbb{N}$ számra és $u_1, \dots, u_n \in \Sigma$ betűkre. Mivel M $p(n)$ időkorlátos gép, azaz minden számítási sorozata maximum $p(n)$ lépésben véget ér a w-n, a fenti táblázat $2(p(n)+3) \cdot (p(n)+1)$ darab cellából áll. Ha $w \in L(M)$, akkor a táblázatnak van egy olyan i-ik sora $(1 \leq i \leq p(n)+1)$ mely M egy megállási konfigurációját reprezentálja. Ekkor megegyezünk abban, hogy a táblázat összes i-nél nagyobb sorszámú sora egyezzen meg az i-ik sorral.

A táblázat minden cellája a $C = Q \cup \Gamma \cup \{\#\}$ halmaz egy elemét tartalmazhatja. Minden cellához és C-beli szimbólumhoz bevezetünk egy ítéletváltozót, azaz



3.1. ábra. SAT **NP**-teljes

minden $i \in [p(n)+1]$-re, $j \in [2p(n)+3]$-ra és $s \in C$-re, az $x_{i,j,s}$ egy ítéletváltozó lesz φ-ben. Az $x_{i,j,s}$ változó azt az állítást reprezentálja, hogy „A táblázat i-ik sorának j-ik oszlopában az s szimbólum van”. A táblázat elfogadó, vagyis $w \in L$, akkor és csak akkor ha az alábbiak teljesülnek:

- A táblázat első sora M kezdőkonfigurációját írja le a w-n.
- A táblázat egymást követő sorai megfelelnek M legális konfigurációátmeneteinek.
- A táblázat utolsó sora M egy elfogadó konfigurációja, azaz az utolsó sorban szereplő állapot a q_i.

A megkonstruált φ-nek azt kell leírnia, hogy a táblázat elfogadó táblázat. Ennek megfelelően φ négy részformulából áll:

$$\varphi = \varphi_0 \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}.$$

φ_0 azt fejezik ki, hogy a táblázat minden egyes mezőjében pontosan egy karakter van:

$$\varphi_0 = \bigwedge_{\substack{i \in [p(n)+1] \\ j \in [2p(n)+3]}} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \bigwedge_{s,t,\in C, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right].$$

φ_{start} azt fejezi ki, hogy a táblázat első sorában a w-hez tartozó kezdőkonfiguráció van:

$$\varphi_{start} = x_{1,1,\#} \wedge x_{1,2,\sqcup} \wedge \ldots \wedge x_{1,p(n)+1,\sqcup} \wedge x_{1,p(n)+2,q_0} \wedge x_{1,p(n)+3,u_1} \wedge \\ \ldots \wedge x_{1,p(n)+n+2,u_n} \wedge x_{1,p(n)+n+3,\sqcup} \wedge \ldots \wedge x_{1,2p(n)+2,\sqcup} \wedge x_{1,2p(n)+3,\#}.$$

φ_{accept} azt fejezi ki, hogy a táblázat utolsó sorában elfogadó konfiguráció van:

$$\varphi_{accept} = \bigvee_{2 \le j \le 2p(n)+2} x_{p(n)+1,j,q_i}.$$

Végezetül, a φ_{move} formulának azt kell kifejeznie, hogy a táblázat két egymást követő sora M két egymást követő konfigurációjának felel meg a w-n. Ahhoz, hogy ezt formalizálni tudjuk, szükségünk lesz némi előkészületre.

Nevezzük a táblázat egy $\begin{array}{|c|c|c|}\hline a_1 & a_2 & a_3 \\ \hline a_4 & a_5 & a_6 \\ \hline \end{array}$ részét a táblázat egy ablakának. Azt mondjuk, hogy a táblázat egy ablaka *legális*, ha a felső és alsó sora megegyezik, vagy ha az ablak „kompatibilis" M átmeneti függvényével.

Példaként tegyük fel, hogy $\delta(q_1, a) = \{(q_1, b, R)\}$ és

$$\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\} \ (q_1, q_2 \in Q \text{ és } a, b, c \in \Gamma).$$

Ekkor például a következő két ablak legális:

$$\begin{array}{|c|c|c|}\hline a & a & q_1 \\ \hline a & a & b \\ \hline \end{array}, \quad \begin{array}{|c|c|c|}\hline a & q_1 & b \\ \hline q_2 & a & c \\ \hline \end{array}.$$

Másrészt, bármilyen további átmeneteket engedjen is meg δ, az $\begin{array}{|c|c|c|}\hline a & a & a \\ \hline a & b & a \\ \hline \end{array}$ ablak nem legális.

Ezek után az általunk megadott φ_{move} azt fogja kifejezni, hogy a táblázat minden ablaka legális. Az, hogy φ_{move} így azt fejezi ki, amit elvárunk tőle következik az alábbi állításból, melyet nem bizonyítunk:

Ha a táblázat első sora az M kezdőkonfigurációja w-n és a táblázat minden ablaka legális, akkor a táblázat bármely két sora által reprezentált C_1 és C_2 konfigurációkra vagy $C_1 \vdash_M C_2$ vagy C_1 elfogadó konfiguráció és $C_1 = C_2$.

Legyen tehát

$$\varphi_{move} = \bigwedge_{\substack{i \in [p(n)] \\ 2 \le j \le 2p(n)+2}} \psi_{i,j},$$

ahol adott i-re és j-re $\psi_{i,j}$ az a formula, aminek azt fejezi ki, hogy a táblázat i-sorában és annak $j-1$-ik pozíciójában kezdődő ablak legális. Ezt az állítást a

$$\bigvee_{\substack{(a_1,\ldots,a_6) \\ \text{legális}}} x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}$$

formulával lehet leírni, de ez a formula sajnos nem KNF. Ezért $\psi_{i,j}$-vel egy ekvivalens állítást fogunk formalizálni: nem igaz az, hogy a táblázat i-sorában és annak $j-1$-ik pozíciójában kezdődő ablak nem legális. Ezt az állítást a

$$\neg\Big(\bigvee_{\substack{(a_1,\ldots,a_6)\\ \text{nem legális}}} x_{i,j-1,a_1}\wedge x_{i,j,a_2}\wedge x_{i,j+1,a_3}\wedge x_{i+1,j-1,a_4}\wedge x_{i+1,j,a_5}\wedge x_{i+1,j+1,a_6}\Big)$$

formula írja le. Ebből a formulából a De Morgan azonosságokat felhasználva kapjuk a keresett $\psi_{i,j}$ formulát:

$$\psi_{i,j} = \bigwedge_{\substack{(a_1,\ldots,a_6)\\ \text{nem legális}}} \neg x_{i,j-1,a_1} \vee \neg x_{i,j,a_2} \vee \neg x_{i,j+1,a_3} \vee \neg x_{i+1,j-1,a_4} \vee \neg x_{i+1,j,a_5} \vee \neg x_{i+1,j+1,a_6}.$$

Mivel így $\psi_{i,j}$ egy KNF, kapjuk, hogy φ_{move} is az. Másrészt a $\varphi_0, \varphi_{start}$, valamint a φ_{accept} részformulák mindegyike KNF. Ezért az egész φ formula maga is KNF. Felhasználva továbbá azt, hogy ezen részformulák mérete n függvényében polinomiális nagyságrendű, megmutatható, hogy φ konstrukciója n függvényében polinom idő alatt elvégezhető.

Másrészt az is könnyen látható, hogy $w \in L$ akkor és csak akkor igaz, ha φ kielégíthető. Tehát φ konstrukciója az L nyelv polinom idejű visszavezetése SAT-ra. Mivel L tetszőleges **NP**-beli nyelv volt, kapjuk, hogy SAT **NP**-nehéz. Ebből, felhasználva, hogy SAT $\in$ **NP**, következik, hogy SAT **NP**-teljes. □

3.1.2. További NP-teljes problémák

Érdekes módon SAT alábbi speciális esete is **NP**-teljes: a probléma példányainak azokat a KNF-eket tekintjük, melyek minden tagja pontosan három literált tartalmaz. A továbbiakban ezt az így kapott problémát vizsgáljuk.

3.9. Definíció

Legyen $k \geq 1$. Ekkor kSAT a következő probléma:

$$k\text{SAT} = \{\langle\varphi\rangle \mid \langle\varphi\rangle \in \text{SAT}, \varphi \text{ minden tagjában pontosan } k \text{ literál van}\}.$$

Most megmutatjuk, hogy a 3SAT probléma is **NP**-teljes.

φ egy c klóza	ψ c-nek megfelelő c' részformulája
l	$(l \vee x \vee y) \wedge (l \vee \neg x \vee y) \wedge (l \vee x \vee \neg y) \wedge (l \vee \neg x \vee \neg y)$
$l_1 \vee l_2$	$(l_1 \vee l_2 \vee x) \wedge (l_1 \vee l_2 \vee \neg x)$
$l_1 \vee l_2 \vee l_3$	$l_1 \vee l_2 \vee l_3$
$l_1 \vee l_2 \vee l_3 \vee l_4$	$(l_1 \vee l_2 \vee x) \wedge (\neg x \vee l_3 \vee l_4)$
$l_1 \vee \cdots \vee l_n$ $(n \geq 5)$	$(l_1 \vee l_2 \vee x_1) \wedge (\neg x_1 \vee l_3 \vee x_2) \wedge \cdots \wedge (\neg x_{n-2} \vee l_{n-1} \vee l_n)$

3.1. táblázat. 3SAT **NP**-teljes

3.10. Tétel

3SAT **NP**-teljes.

Bizonyítás. Mivel SAT $\in$ **NP**, nyilvánvaló, hogy 3SAT $\in$ **NP** is teljesül. Megmutatjuk, hogy SAT $\leq_p$ 3SAT, amiből a 3.7. tétel alapján következik, hogy 3SAT **NP**-teljes.

Legyen φ a SAT egy bemenete. φ-hez megkonstruálunk egy olyan ψ formulát, melyben minden tag pontosan három literált tartalmaz, és $\langle\varphi\rangle \in$ SAT akkor és csak akkor áll fenn, ha $\langle\psi\rangle \in$ 3SAT teljesül.

A 3.1. táblázat szerint φ minden c tagját átalakítjuk a ψ egy c' olyan részformulájává ami KNF. A táblázatban $l, l_1, \ldots, l_n$ literálokat, $x, y, x_1, \ldots, x_{n-2}$ pedig mindig új, korábban még nem használt ítéletváltozókat jelöl.

Mivel ψ klózainak mérete a megfelelő φ-beli klózok konstansszorosa, ψ konstrukciója elvégezhető polinom időben. Az, hogy $\langle\varphi\rangle \in$ SAT akkor és csak akkor, ha $\langle\psi\rangle \in$ 3SAT a következőképpen látható be. Egyrészt, ha φ kielégíthető, akkor az őt kielégítő I interpretáció mindig kiterjeszthető az $x, y, x_1, \ldots, x_n$ változókra úgy, hogy a kapott interpretáció kielégítse a ψ összes klózát. Ez az egy, két, és három literált tartalmazó φ-beli klózokból kapott ψ-beli klózok esetében könnyen látható. Ha a ψ egy c klóza $l_1 \vee l_2 \vee l_3 \vee l_4$ alakú, akkor $I \models l_1 \vee l_2$ vagy $I \models l_3 \vee l_4$. De akkor c-nek megfelelő ψ-beli c' formula *igazzá* tehető az x értékének megfelelő megválasztásával. Ez az észrevétel általánosítható a négynél több literált tartalmazó φ-beli klózokra is.

Fordítva, ha ψ igaz egy I interpretációban, akkor mivel az $x, y, x_1, \ldots, x_n$ változókat tartalmazó literálok nem lehetnek egyszerre igazak egy φ-beli c klóznak megfelelő ψ-beli c' összes klózában, azt kapjuk, hogy olyan literálnak is igaznak kell lennie valamelyik c'-beli klózban, ami szerepel a c-ben is. De ekkor c-t ugyancsak kielégíti az I. Azaz φ is igaz lesz I-ben.

A fenti konstrukció tehát a SAT egy polinom idejű visszavezetése a 3SAT-ra. A 3.7. tétel alapján következik, hogy 3SAT **NP**-teljes □

A 3SAT segítségével további problémákról mutatjuk meg, hogy **NP**-teljesek. Először a következő, egymással szoros kapcsolatban álló gráfelméleti problémákkal foglalkozunk: TELJES RÉSZGRÁF, FÜGGETLEN CSÚCSHALMAZ és CSÚCSLEFEDÉS.

3.11. Definíció

Az alábbiakban G mindig egy irányítatlan gráfot jelöl.

$$\text{TELJES RÉSZGRÁF} = \{\langle G, k\rangle \mid k \geq 1, G\text{-nek létezik } k \text{ csúcsú teljes részgráfja }\}.$$

Tehát a TELJES RÉSZGRÁF azon G és k párokat tartalmazza egy megfelelő ábécé feletti szavakkal kódolva, melyekre igaz, hogy G-ben van k csúcsú teljes részgráf, azaz olyan részgráf, melyben bármely két csúcs között van él.

$$\text{FÜGGETLEN CSÚCSHALMAZ} = \{\langle G, k\rangle \mid k \geq 1, G\text{-nek van } k \text{ elemű független csúcshalmaza }\}.$$

Vagyis a FÜGGETLEN CSÚCSHALMAZ azon G és k párokat tartalmazza, melyekre igaz, hogy G-ben van k olyan csúcs, melyek közül egyik sincs összekötve a másikkal.

CSÚCSLEFEDÉS $= \{\langle G, k\rangle \mid k \geq 1, G$-nek van olyan k elemű csúcshalmaza, mely tartalmazza G minden élének legalább egyik végpontját $\}$.

Megmutatjuk, hogy a fenti problémák **NP**-teljesek.

3.12. Tétel

TELJES RÉSZGRÁF **NP**-teljes.

Bizonyítás. Először is, TELJES RÉSZGRÁF $\in$ **NP**, hisz megadható egy nemdeterminisztikus Turing-gép, ami az egyik szalagjára írja a bemenetként kapott G gráf k darab csúcsát (azaz megsejt k darab csúcsot G-ből), majd polinom időben ellenőrzi, hogy ezen csúcsok mindegyike össze van-e kötve a $k-1$ másik csúccsal.

Másrészt megmutatjuk, hogy 3SAT $\leq_p$ TELJES RÉSZGRÁF a következő módon. Legyen φ a 3SAT egy példánya. Ekkor $\varphi = c_1 \wedge \ldots \wedge c_k$, valamely $k \geq 1$-re,

továbbá minden $i \in [k]$-ra, $c_i = l_{i_1} \vee l_{i_2} \vee l_{i_3}$, ahol l_{i_j} ($j \in [3]$) literálok. φ-hez megadunk egy G_φ gráfot az alábbi módon. φ minden c_i tagja meghatároz három csúcsot G_φ-ben, nevezzük ezt a c_i-hez tartozó háromszögnek (annak ellenére, hogy e csúcsok között nem lesznek élek). A c_i-hez tartozó háromszögek csúcsait l_{i_1}-el l_{i2}-vel és l_{i3}-mal jelöljük. Ezután G_φ csúcsai között az összes élt behúzzuk, kivéve az alábbiakat:

- az egy klózhoz tartozó háromszög csúcsai közti éleket és
- az ellentétes literálokkal címkézett csúcsok közti éleket.

Nyilvánvaló, hogy G_φ polinom időben megkonstruálható, továbbá az is könnyen belátható, hogy $\langle\varphi\rangle \in$ 3SAT $\Leftrightarrow \langle G, k\rangle \in$ TELJES RÉSZGRÁF. Tehát a fenti konstrukció 3SAT polinom idejű visszavezetése TELJES RÉSZGRÁF-ra. Mivel 3SAT **NP**-teljes, kapjuk, hogy TELJES RÉSZGRÁF is **NP**-teljes. □

Most azt mutatjuk meg, hogy FÜGGETLEN CSÚCSHALMAZ is **NP**-teljes.

3.13. Tétel

FÜGGETLEN CSÚCSHALMAZ **NP**-teljes.

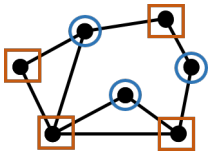
Bizonyítás. FÜGGETLEN CSÚCSHALMAZ $\in$ **NP**, hisz megadható egy nemdeterminisztikus Turing-gép, ami megsejt k darab csúcsot G-ben és polinom időben ellenőrzi, hogy ezen csúcsok egyike sincs semelyik másikkal összekötve.

Ezek után ahhoz, hogy a FÜGGETLEN CSÚCSHALMAZ **NP**-teljességét belássuk elegendő megmutatni, hogy TELJES RÉSZGRÁF $\leq_p$ FÜGGETLEN CSÚCSHALMAZ. Legyen G egy tetszőleges gráf és $\overline{G}$ az a gráf, melynek ugyanazok a csúcsai, mint G-nek és két csúcs között akkor és csak akkor van él $\overline{G}$-ben, ha ugyanezen csúcsok között nincs él G-ben. Világos, hogy $\overline{G}$ polinom időben megkonstruálható G-ből. Továbbá a G gráfban akkor és csak akkor van k elemű teljes részgráf, ha $\overline{G}$-ben van k elemű független csúcshalmaz. Kapjuk tehát, hogy TELJES RÉSZGRÁF polinom időben visszavezethető FÜGGETLEN CSÚCSHALMAZ-ra, amit bizonyítani akartunk. □

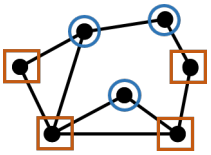
3.14. Tétel

CSÚCSLEFEDÉS **NP**-teljes.

Bizonyítás. Először megint azt kell belátni, hogy CSÚCSLEFEDÉS $\in$ **NP**. Ez viszont igaz, hisz CSÚCSLEFEDÉS eldönthető egy olyan nemdeterminisztikus Turing-gépel, ami megsejt k darab csúcsot G-ben, és polinom időben ellenőrzi, hogy G minden élének legalább az egyik végpontja rajta van-e a k csúcs valamelyikén.



(a) A bekarikázott csúcsok független csúcshalmazt alkotnak, ezért a gráf minden élének legalább egy csúcsa a maradék (bekeretezett) csúcsok egyikén van.



(b) A bekarikázott csúcsok nem alkotnak független csúcshalmazt, ezért van olyan él a gráfban, melynek egyik csúcsa sincs rajta a maradék (bekeretezett) csúcsok valamelyikén.

3.2. ábra. FÜGGETLEN CSÚCSHALMAZ $\leq_p$ CSÚCSLEFEDÉS

A CSÚCSLEFEDÉS probléma **NP**-teljességét úgy bizonyítjuk, hogy visszavezetjük rá az **NP**-teljes FÜGGETLEN CSÚCSHALMAZ problémát. Tekintsünk egy n csúcsú ($n \geq 1$) véges G gráfot és annak egy k elemű ($k \leq n$) G' részgráfját. Nem nehéz belátni, hogy a G'-beli csúcsok pontosan akkor alkotnak egy k elemű független csúcshalmazt G-ben, ha G-nek a G'-n kívüli csúcsai egy $n-k$ elemű csúcslefedést alkotnak G-ben (lásd például a 3.2. ábrát). Tehát $\langle G, k\rangle \in$ FÜGGETLEN CSÚCSHALMAZ $\Leftrightarrow \langle G, n-k\rangle \in$ CSÚCSLEFEDÉS. Világos, hogy az $n-k$ szám polinom időben kiszámítható, vagyis FÜGGETLEN CSÚCSHALMAZ $\leq_p$ CSÚCSLEFEDÉS. Következésképpen CSÚCSLEFEDÉS is **NP**-teljes. □

3.1.3. Hamilton-úttal kapcsolatos NP-teljes problémák

A következőkben megvizsgálunk néhány olyan problémát, ahol különböző gráfokban kell Hamilton-utat vagy kört találni. Az eredményeket felhasználva meg fogjuk mutatni, hogy a jól ismert UTAZÓ ÜGYNÖK probléma **NP**-teljes. A vizsgált problémákat, mint formális nyelveket, a következőképpen definiáljuk.

3.15. Definíció

HAMILTON-ÚT $= \{\langle G, s, t\rangle \mid G = (V, E)$ irányított gráf, $s, t \in V$, és $\exists s$-ből t-be Hamilton-út$\}$.

IRÁNYÍTATLAN HAMILTON-ÚT $= \{\langle G, s, t\rangle \mid G = (V, E)$ irányítatlan gráf, $s, t \in V$, és $\exists s$-ből t-be Hamilton-út$\}$.

TETSZŐLEGES HAMILTON-ÚT $= \{\langle G \rangle \mid G = (V, E)$ irányítatlan gráf, és $\exists s, t \in V$ úgy, hogy van s-ből t-be Hamilton-út$\}$.

IRÁNYÍTATLAN HAMILTON-KÖR $= \{\langle G \rangle \mid G$ olyan irányítatlan gráf, amiben van Hamilton-kör$\}$.

UTAZÓÜGYNÖK $= \{\langle G, k \rangle \mid G$ irányítatlan gráf az éleken egy-egy pozitív egész súllyal, és G-ben van legfeljebb k összsúlyú Hamilton-kör$\}$.

Először azt mutatjuk meg, hogy irányított gráfokban a Hamilton-út keresése **NP**-teljes probléma.

3.16. Tétel

A HAMILTON-ÚT probléma **NP**-teljes.

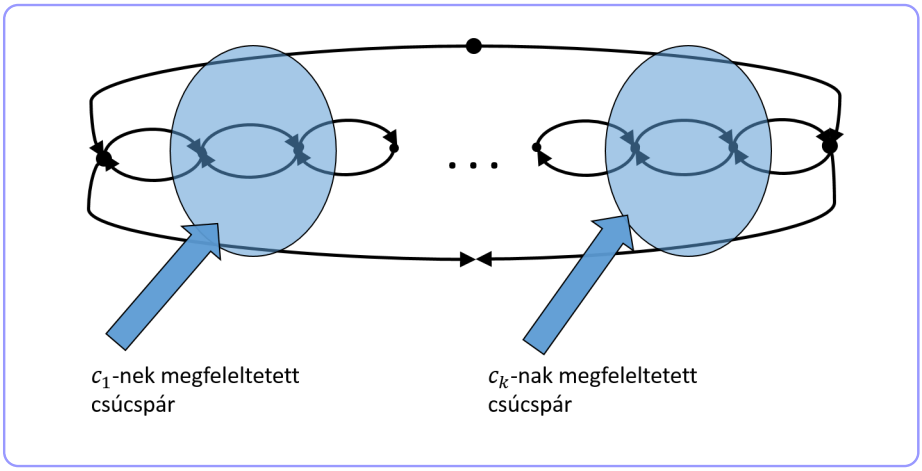
Bizonyítás. Az, hogy a probléma **NP**-beli könnyen belátható: megadható egy nemdeterminisztikus Turing-gép, ami megsejti a bemenő gráf csúcsainak egy $s, \ldots, t$ felsorolását és polinom időben ellenőrzi, hogy a felsorolásban a csúcsok páronként különböznek-e, valamint azt, hogy az egymást követő csúcsok között van-e él.

HAMILTON-ÚT **NP**-teljességének belátásához elegendő tehát megmutatni, hogy egy **NP**-teljes probléma visszavezethető rá. Ez a probléma a SAT lesz. Legyen egy φ egy ítéletkalkulusbeli KNF. Polinom időben megkonstruálunk egy G_φ gráfot úgy, hogy G_φ két kitüntetett csúcsa között pontosan akkor lesz Hamilton-út, ha $\langle \varphi \rangle \in$ SAT.

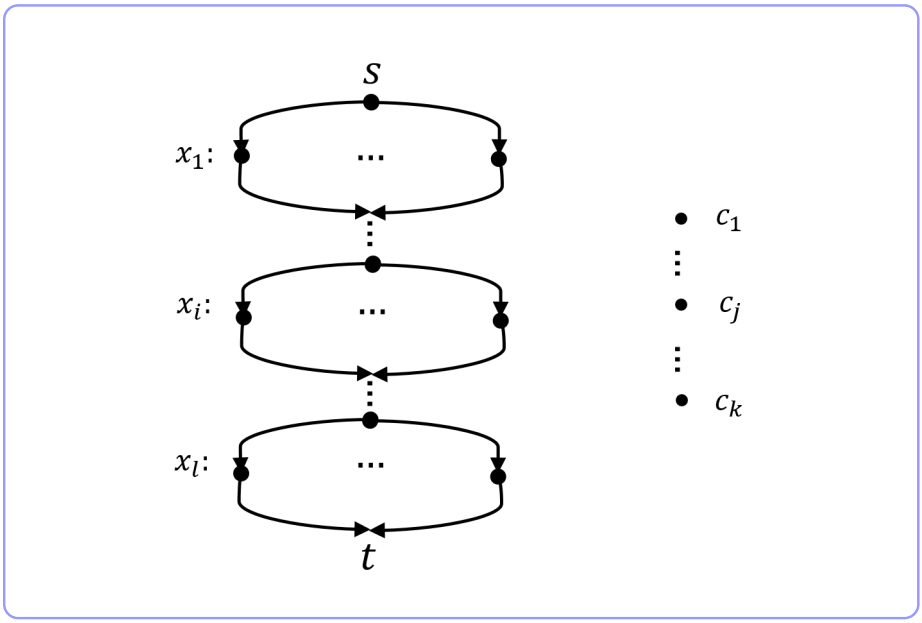
Tegyük fel, hogy $\varphi = c_1 \wedge \ldots \wedge c_k$ $(k \geq 1)$ és, hogy φ-ben csak az $x_1, x_2, \ldots, x_l$ $(l \geq 1)$ változók szerepelnek. Minden x_i változóhoz $(i \in [l])$ megadunk egy a 3.3. ábrán látható részgráfot.

Ebben a részgráfban, a bal oldali csúcs melletti csúcstól kezdve, két-két egymás melletti csúcs rendre a φ egy-egy tagjának van megfeleltetve (az ábrán ezek a csúcspárok be vannak karikázva).

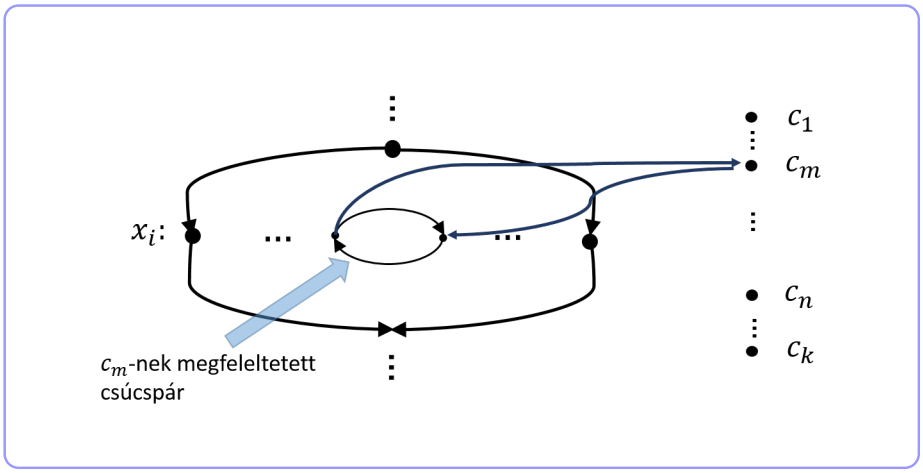
Továbbá minden c_j $(j \in [k])$ taghoz lesz egy-egy külön csúcs G_φ-ben. Ezekből a részgráfokból felépítjük a 3.4. ábrán látható gráfot. Ebben a gráfban a változók által reprezentált eszközöket az alábbi stratégia szerint kötjük össze a tagokat reprezentáló csúcsokkal. Ha egy x_i változó szerepel egy c_m tagban, akkor az x_i-nek megfelelő részgráf c_m-nek megfelelő két csúcsa közül vesszük a *baloldalit* és behúzunk egy élt ebből a csúcsból a c_m csúcsba. Ezután c_m-ből behúzunk



3.3. ábra. Az x_i változóhoz konstruált részgráf



3.4. ábra. A φ-beli változókhoz illetve tagokhoz konstruált részgráfok



3.5. ábra. Amikor az x_i literál szerepel egy c_m tagban

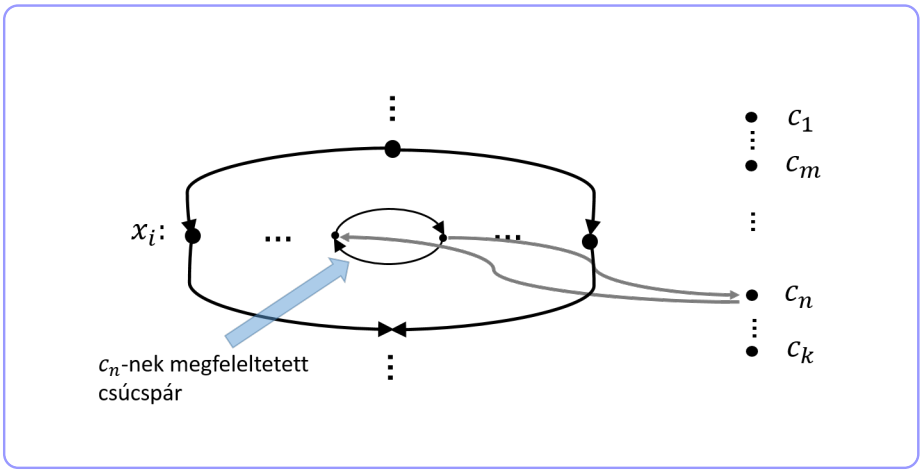
egy élt a második (jobboldali) csúcsba (3.5. ábra).

Ha viszont egy $\neg x_i$ literál szerepel egy c_n tagban, akkor az x_i-nek megfelelő részgráf c_n-nek megfelelő két csúcsa közül vesszük a *jobboldalit* és behúzunk egy élt ebből a csúcsból a c_n csúcsba. Ezután c_n-ből behúzunk egy élt az első (baloldali) csúcsba (3.6. ábra).

Legyen az így kapott gráf G_φ, és legyen s, t rendre a 3.4. ábrán látható két csúcs. Belátható, hogy $\langle G_\varphi, s, t\rangle$ polinomidejű Turing-géppel elkészíthető $\langle\varphi\rangle$-ből. Meg kell még mutatni, hogy $\langle\varphi\rangle \in$ SAT akkor és csak akkor teljesül, ha $\langle G_\varphi, s, t\rangle \in$ HAMILTON-ÚT.

Tegyük fel, hogy $\langle\varphi\rangle \in$ SAT. Legyen I egy olyan interpretáció, ami kielégíti φ-t. I meghatároz egy utat s-ből t-be a következő módon. Ha I egy x_i-hez ($i \in [l]$) *igaz*at rendel, akkor az út az x_i-nek megfelelő részgráf felső csúcsából először balra megy, majd a részgráf bal oldalán lévő csúcsból jobbra lépked egészen addig, amíg eléri a jobb oldalon lévő csúcsot, végül pedig a részgráf alján lévő csúcsot érinti. Ha I az x_i-hez *hamis*at rendel, akkor az út az ellenkező irányban járja be az x_i-nek megfelelő részgráfot: a felső csúcsból először jobbra megy, majd a részgráf jobb oldalán lévő csúcsból balra lépked addig, amíg eléri a részgráf bal oldalán lévő csúcsot, ahonnan pedig továbblép a részgráf alján lévő csúcsra.

Ezt a fent vázolt utat most kiegészítjük úgy, hogy a $c_1, \ldots, c_k$ tagoknak megfelelő csúcsokat is érintse. Tekintsük egy c_j tagot ($j \in [k]$), és vegyünk c_j-ből egy *igaz* literált (ilyen biztos, hogy létezik, hiszen I kielégíti a φ-t). Ez a literál x_i vagy $\neg x_i$ alakú valamely $i \in [l]$-re. Ha a literál x_i alakú, akkor az út balról jobbra halad az x_i-nek megfelelő részgráfban (hisz x_i értéke *igaz*). Ha a literál $\neg x_i$ alakú, akkor az út jobbról balra halad (mert az x_i értéke *hamis*). A c_j csúcs



3.6. ábra. Amikor a $\neg x_i$ literál szerepel egy c_n tagban

viszont úgy van hozzákötve az x_i-nek megfelelő részgráfban a c_j-nek megfelelő csúcspárhoz, hogy az utunk mindkét esetben képes megtenni egy kitérőt a c_j csúcshoz. Nem nehéz belátni, hogy az így vázolt út egy Hamilton-út a G_φ-ben s-ből t-be.

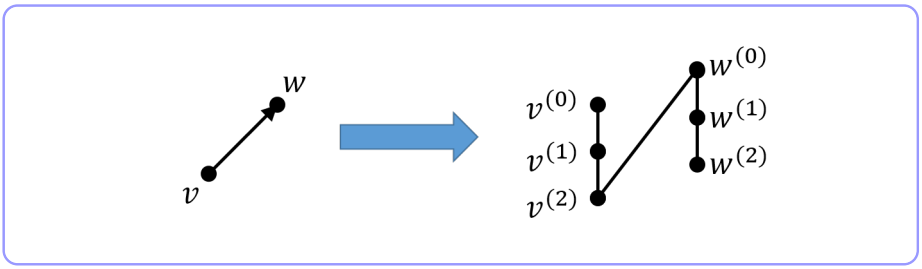
Tegyük fel most, hogy van egy Hamilton-út s-ből t-be a G_φ-ben. Nem nehéz belátni, hogy ez az út olyan, mint amilyet az előbb vázoltunk, azaz az út nem teheti meg azt, hogy egy x_i-nek megfelelő részgráfból kitérőt tesz egy c_j csúcshoz, de a c_j-ből nem ugyanabba a részgráfba megy vissza. Ellenkező esetben ugyanis lennének olyan csúcsai G_φ-nek, amit az út nem lenne képes meglátogatni, ez viszont ellentmondana annak, hogy az út Hamilton-út. Így tehát a Hamilton-út meghatároz egy olyan I interpretációt, ami kielégíti φ-t. □

Most azt mutatjuk meg, hogy a HAMILTON-ÚT probléma akkor is **NP**-teljes marad, ha megengedjük azt, hogy a bemenetben szereplő gráf irányítatlan legyen.

3.17. Tétel

AZ IRÁNYÍTATLAN HAMILTON-ÚT probléma **NP**-teljes.

Bizonyítás. Az, hogy a probléma **NP**-beli hasonlóan adódik, mint a HAMILTON-ÚT probléma esetében. Azt, hogy a probléma **NP**-nehéz úgy bizonyítjuk, hogy visszavezetjük rá a HAMILTON-ÚT problémát. Adott $G = (V, E)$ irányított gráfhoz és $s, t \in V$ csúcsokhoz legyen $G_u = (V_u, E_u)$ a következő irányítatlan gráf. Legyen $s^{(2)}, t^{(0)} \in V_u$ és minden $v \in V - \{s, t\}$ csúcsra legyen $v^{(0)}, v^{(1)}, v^{(2)}$ egy-egy csúcs V_u-ban, valamint $\{v^{(0)}, v^{(1)}\}$ és $\{v^{(1)}, v^{(2)}\}$ egy-egy él E_u-ban. Legyen továbbá minden $(v, w) \in E$ élre $\{v^{(2)}, w^{(0)}\} \in E_u$. A 3.7. ábrán az látható,



3.7. ábra. HAMILTON-ÚT visszavezetése IRÁNYÍTATLAN HAMILTON-ÚT-ra

hogy a (v, w) G-beli élnek milyen élek felelnek meg G_u-ban.

Világos, hogy G_u polinom időben megkonstruálható G-ből. Annak belátásához, hogy $\langle G, s, t\rangle \in$ HAMILTON-ÚT akkor és csak akkor teljesül, ha $\langle G_u, s^{(2)}, t^{(0)}\rangle \in$ IRÁNYÍTATLAN HAMILTON-ÚT elég meggondolni a következőket. Az világos, hogy ha G-ben van egy $s, v_1, \ldots, v_{n-2}, t$ (irányított) Hamilton-út, ahol n a G csúcsainak a száma, akkor az a H út mely rendre az

$$s^{(2)}, v_1^{(0)}, v_1^{(1)}, v_1^{(2)}, \ldots, v_{n-2}^{(0)}, v_{n-2}^{(1)}, v_{n-2}^{(2)}, t^{(0)}$$

csúcsokat érinti, egy Hamilton-út G_u-ban $s^{(2)}$-ből $t^{(0)}$-ba. Másrészt, tegyük fel, hogy van egy Hamilton-út G_u-ban $s^{(2)}$-ből $t^{(0)}$-ba. Ez az út az $s^{(2)}$ után és a $t^{(0)}$ előtt, nyilvánvalóan, $v^{(0)}, v^{(1)}, v^{(2)}$ sorrendben látogatja meg a csúcsokat minden $v \in V - \{s^{(2)}, t^{(0)}\}$-ra. Tehát ez a G_u-beli Hamilton-út, szükségszerűen, olyan alakú, mint a fentebb látott H-val jelölt Hamilton-út. Ekkor viszont az $s, v_1, \ldots, v_{n-2}, t$ csúcssorozat egy s-ből t-be tartó (irányított) Hamilton-utat alkot G-ben. Ezzel a tételt bebizonyítottuk. □

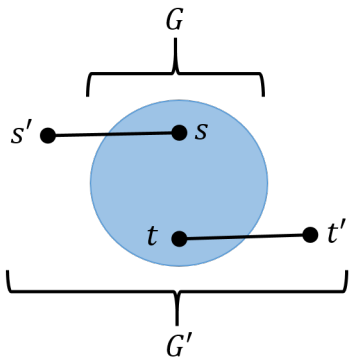
Érezhető, hogy ha IRÁNYÍTATLAN HAMILTON-ÚT **NP**-teljes, akkor a TETSZŐLEGES HAMILTON-ÚT és az IRÁNYÍTATLAN HAMILTON-KÖR problémák sem lehetnek könnyebbek.

3.18. Tétel

A TETSZŐLEGES HAMILTON-ÚT probléma **NP**-teljes.

Bizonyítás. Az, hogy a probléma **NP**-beli hasonlóan adódik, mint az IRÁNYÍTATLAN HAMILTON-ÚT probléma esetében. Azt, hogy a probléma **NP**-nehéz úgy bizonyítjuk, hogy visszavezetjük rá az IRÁNYÍTATLAN HAMILTON-ÚT problémát. Adott $G = (V, E)$ irányítatlan gráfhoz és $s, t \in V$ csúcsokhoz legyen $G' = (V', E')$ az az irányítatlan gráf, amit a 3.8. ábrán látható módon konstruálunk meg.

Világos, hogy G' polinom időben megkonstruálható. Figyeljük meg, hogy ha G'-ben van valamely két csúcs között Hamilton-út, akkor ez a két csúcs csakis az s'



3.8. ábra. IRÁNYÍTATLAN HAMILTON-ÚT visszavezetése TETSZŐLEGES HAMILTON-ÚT-ra

és t' lehet. Ebből már következik, hogy G-ben akkor és csak akkor van Hamilton-út s-ből t-be, ha V'-ben van két olyan csúcs, melyek között van Hamilton-út. □

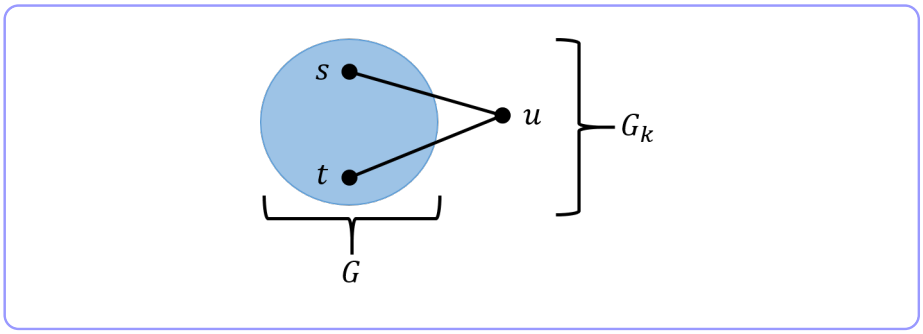
3.19. Tétel

AZ IRÁNYÍTATLAN HAMILTON-KÖR probléma **NP**-teljes.

Bizonyítás. Az, hogy a probléma **NP**-beli megint csak hasonlóan adódik, mint az IRÁNYÍTATLAN HAMILTON-ÚT probléma esetében. Azt, hogy a probléma **NP**-nehéz úgy bizonyítjuk, hogy visszavezetjük rá az IRÁNYÍTATLAN HAMILTON-ÚT problémát. Adott $G = (V, E)$ irányítatlan gráfhoz és $s, t \in V$ csúcsokhoz legyen $G_k = (V_k, E_k)$ az az irányítatlan gráf, amit a 3.9. ábrán látható módon konstruálunk meg.

Látható, hogy G_k polinom időben megkonstruálható, és akkor és csak akkor tartalmaz Hamilton-kört, ha G-ben van Hamilton-út s-ből t-be. □

A következőkben azt mutatjuk meg, hogy az UTAZÓ ÜGYNÖK probléma is **NP**-teljes. A probléma **NP**-nehézségének bizonyításához a következő, általánosan használható gondolatmenetet alkalmazzuk. Legyen P_1 és P_2 két eldöntési probléma úgy, hogy a P_1 a P_2 speciális esete. Ha van egy A algoritmusunk a P_2 eldöntésére, akkor az A-t nyilvánvalóan használhatjuk a P_1 eldöntésére is. Tehát a P_1 nem lehet bonyolultabb probléma, mint a P_2. Ebből következik az alábbi állítás.



3.9. ábra. IRÁNYÍTATLAN HAMILTON-KÖR **NP**-teljes

3.20. Tétel

Legyen P_1 és P_2 két probléma úgy, hogy a P_1 a P_2 speciális esete és P_1 **NP**-nehéz. Akkor P_2 is **NP**-nehéz.

Ezt felhasználva könnyen adódik az alábbi eredmény.

3.21. Tétel

AZ UTAZÓ ÜGYNÖK probléma **NP**-teljes.

Bizonyítás. Azt a korábbiak alapján ismét nem nehéz belátni, hogy a probléma **NP**-beli. Az **NP**-nehézség a 3.20. tétel alapján abból adódik, hogy a IRÁNYÍTATLAN HAMILTON-KÖR probléma az UTAZÓ ÜGYNÖK probléma speciális esete. Valóban, ha az UTAZÓ ÜGYNÖK probléma bemenetei csak olyan $\langle G, k\rangle$ párok lehetnek, ahol $G = (V, E)$ egy olyan élsúlyozott irányítatlan gráf, melyben az élek 1-gyel vannak súlyozva, és a $k = |V|$, akkor megkapjuk a IRÁNYÍTATLAN HAMILTON-KÖR problémát. □

Végezetül azt mutatjuk meg, hogy az alábbi két probléma is **NP**-teljes.

3.22. Definíció

LEGHOSSZABB ÚT $= \{\langle G, k\rangle \mid G = (V, E)$ irányítatlan gráf,
$k \leq |V|, G$-ben van legalább k csúcsot érintő út$\}$,

KORLÁTOZOTT FESZÍTŐFA =
$\{\langle G, k\rangle \mid G = (V, E)$ irányítatlan gráf, $k \leq |V|$ és G-ben van olyan G' feszítőfa, hogy G'-ben minden csúcs foka legfeljebb $k\}$.

3.23. Tétel

LEGHOSSZABB ÚT és KORLÁTOZOTT FESZÍTŐ FA **NP**-teljesek.

Bizonyítás. Könnyen ellenőrizhető, hogy mindkét probléma polinom időben ellenőrizhető, azaz **NP**-ben van. Megmutatjuk, hogy TETSZŐLEGES HAMILTON-ÚT mindkét problémának speciális esete. Ebből a 3.20. tétel alapján következnek a tétel állításai. Ha a LEGHOSSZABB ÚT bemenetét úgy korlátozzuk, hogy k-t $|V|$-nek választjuk, akkor nyilvánvalóan a TETSZŐLEGES HAMILTON-ÚT problémát kapjuk. Ha pedig a KORLÁTOZOTT FESZÍTŐ FA bemeneteit korlátozzuk úgy, hogy $k = 2$, akkor a probléma olyan feszítő fa keresésére redukálódik, ahol minden csúcs foka legfeljebb kettő. De egy gráfban egy u út akkor és csak akkor Hamilton-út, ha u egy olyan feszítőfa, melyben a csúcsok foka legfeljebb kettő. Kapjuk tehát, hogy a fenti korlátozással ismét a TETSZŐLEGES HAMILTON-ÚT problémát kaptuk. □

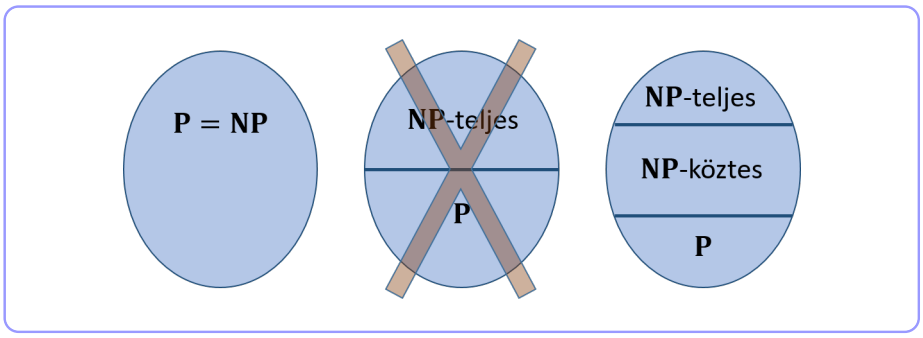
Az **NP**-teljes problémák nagyon fontosak a bonyolultságelméletben. Általában, ha egy új **NP**-beli problémával találkozunk, akkor vagy azt próbáljuk megmutatni róla, hogy **P**-beli vagy azt, hogy **NP**-teljes. Ha **P**-beli, akkor rendszerint hatékonyan megoldható a gyakorlatban is, ha **NP**-teljes, akkor viszont (valószínűleg) nincs a megoldására hatékony algoritmus, és így fel is hagyhatunk annak keresésével. Azt már tudjuk, hogy $\mathbf{P} \subseteq \mathbf{NP}$. Elvileg lehetséges, hogy $\mathbf{P} = \mathbf{NP}$, de mint említettük az a sejtés, hogy ez az egyenlőség nem áll fenn. Azt is tudjuk, hogy az **NP**-teljes problémák a legnehezebbek az **NP**-beli nyelvek között. Vajon van-e olyan nyelv, ami nem eleme **P**-nek, de nem is **NP**-teljes? Nyilvánvaló, hogy ha $\mathbf{P} = \mathbf{NP}$, akkor nincs ilyen nyelv, ellenkező esetben viszont igaz az alábbi tétel, amit bizonyítás nélkül közlünk:

3.24. Tétel

Ha $\mathbf{P} \neq \mathbf{NP}$, akkor van olyan $L \in \mathbf{NP}$ nyelv, hogy $L \notin \mathbf{P}$, de L nem is **NP**-teljes.

A fenti tételben szereplő nyelveket nevezzük **NP**-*köztes* nyelveknek. Egy lehetséges **NP**-köztes probléma a következő:

GRÁF IZOMORFIZMUS $= \{\langle G_1, G_2\rangle \mid G_1$ és G_2 izomorf irányítatlan gráfok$\}$.



3.10. ábra. Az **NP** szerkezete

Ismert, hogy a Gráf izomorfizmus probléma **NP**-ben van. Az a tény, hogy eddig se azt nem sikerül bizonyítani, hogy **P**-beli, se azt, hogy **NP**-teljes, szintén a $\mathbf{P} \subset \mathbf{NP}$ sejtést erősíti. Megjegyezzük azonban, hogy a jegyzet megjelenésének idejében egy nagyon fontos eredményt jelentett be Babai László magyar matematikus. Ezek szerint a Gráf izomorfizmus egy olyan probléma ami úgynevezett kvázipolinomiális időben megoldható, azaz Gráf izomorfizmus $\in \bigcup_{k\in\mathbb{N}} \mathbf{TIME}(2^{(\log n)^k})$. Ez azt jelenti, hogy ez a probléma mégis inkább a **P** osztályhoz van közelebb, és nem lehetetlen, hogy egyszer sikerül azt is megmutatni, hogy **P**-ben van. Ez esetben természetesen nem lenne igaz, hogy a Gráf izomorfizmus **NP**-köztes probléma, így a segítségével nem lehetne bizonyítani a $\mathbf{P} \subset \mathbf{NP}$ sejtést sem.

A Gráf izomorfizmus speciális esete az alábbi problémának:

$$\text{Részgráf izomorfizmus} = \{\langle G_1, G_2\rangle \mid \ G_1 \text{ és } G_2 \text{ irányítatlan gráfok, és } G_1\text{-nek van } G_2\text{-vel izomorf részgráfja}\}.$$

Nyilvánvaló, hogy ha a Részgráf izomorfizmus probléma bemeneteit megszorítjuk úgy, hogy az egyik gráf csak teljes gráf lehet, akkor megkapjuk a Teljes részgráf problémát. Ebből, mivel a probléma nyilvánvalóan **NP**-beli, a 3.20. tétel alapján kapjuk a következő eredményt.

3.25. Tétel

Részgráf izomorfizmus **NP**-teljes.

Az **NP** osztály lehetséges szerkezetét a 3.10. ábrán szemléltetjük.

3.1.4. A coNP osztály

Érdemes megvizsgálni egyes **C** bonyolultsági osztályokra a **C**-beli problémák komplementereinek osztályát, azaz a

$$\mathbf{coC} = \{\overline{L} \mid L \in C\}$$

osztályt. Determinisztikus bonyolultsági osztályokra a **C** és a **coC** megegyezik, hisz ha egy $L \in \mathbf{C}$ probléma eldönthető egy M Turing-géppel, akkor az M elfogadó és elutasító állapotát megcserélve egy $\overline{L}$-et eldöntő Turing-gépet kapunk. Azaz például $\mathbf{P} = \mathbf{coP}$.

Nemdeterminisztikus bonyolultsági osztályoknál azonban nem ilyen egyszerű a helyzet. Tekintsük például az ÁltSat problémát, ami a Sat azon általánosítása, melyben tetszőleges formula (azaz nem csak KNF) lehet a bementet. Ismert, hogy ez a probléma is **NP**-teljes. Tekintsük az Unsat $= \overline{\text{ÁltSat}}$ problémát, azaz annak eldöntését, hogy egy φ ítéletkalkulusbeli formula kielégíthetetlen-e. Érdekes kérdés, hogy Unsat benne van-e **NP**-ben.

Az olvasó gondolhatná, hogy miért lenne más a bonyolultsága egy kérdés eldöntésének attól, hogy nem azt kérdezzük, hogy teljesül-e valami, hanem azt hogy nem teljesül-e. Az igazság az, hogy már láttunk példát arra, hogy egy probléma bonyolultsága nem egyezik meg a komplementerének a bonyolultságával. Az összes probléma ami nem eldönthető, de rekurzívan felsorolható ilyen probléma, hisz ezek komplementere nem rekurzívan felsorolható. Az a sejtés, hogy az **NP** és **coNP** osztályok összehasonlíthatatlanok, és azok a problémák melyek teljesek az egyik osztályban nincsenek benne a másikban. Mint látni fogjuk, az **NP**-teljes problémák komplementerei **coNP**-teljesek, tehát az ÁltSat és az Unsat vélhetően nem azonos bonyolultságúak.

Polinom időben cáfolhatóság

Ahogy arról már volt szó, az **NP** tartalmazza a polinom időben ellenőrizhető problémákat. Értelemszerűen akkor **coNP** tartalmazza a polinom időben cáfolható problémákat. Az ÁltSat esetében ez azt jelenti, hogy adott φ formulára az, hogy $\langle\varphi\rangle \notin$ Unsat igazolható úgy, hogy megadunk egy I interpretációt, amit kielégíti φ-t.

Legyen **C** eldöntési problémák, v pedig kiszámítható függvények egy osztálya. Azt mondjuk, hogy egy $L \in \mathbf{C}$ nyelv **C**-*teljes a* v-*beli visszavezetésekre nézve*, ha minden $L' \in \mathbf{C}$ esetén, $L' \leq_v L$. Az alábbi tétel alapján az Unsat probléma például **coNP**-teljes.

3.26. Tétel

Legyen **C** eldöntési problémák, v pedig kiszámítható függvények egy osztálya. Egy L nyelv akkor és csak akkor **C**-teljes a v-beli visszavezetésekre nézve, ha $\overline{L}$ **coC**-teljes a v-beli visszavezetésekre nézve.

Bizonyítás. Mivel $\mathbf{co}(\mathbf{coC}) = \mathbf{C}$ elég az egyik irányt bizonyítani. Legyen tehát L egy $\mathbf{C}$-teljes nyelv. Ekkor minden $L' \in \mathbf{C}$-re $L' \leq_v L$. De ez pontosan azt jelenti, hogy minden $L' \in \mathbf{C}$-re, $\overline{L'} \leq_v \overline{L}$. Azaz $\overline{L}$ $\mathbf{coC}$-teljes a v-beli visszavezetésekre nézve. □

A $\mathbf{C}$ és $\mathbf{coC}$ osztályok között a következő kapcsolatot figyelhetjük meg.

3.27. Tétel

Legyen $\mathbf{C}$ eldöntési problémák, v pedig kiszámítható függvények egy osztálya. Ha $\mathbf{C}$ zárt a v-beli visszavezetésekre nézve, akkor $\mathbf{coC}$ is az.

Bizonyítás. Legyen L_1 és L_2 két nyelv úgy, hogy $L_1 \leq_v L_2$ és $L_2 \in \mathbf{coC}$. Ekkor nyilvánvalóan $\overline{L_1} \leq_v \overline{L_2}$ valamint $\overline{L_2} \in \mathbf{C}$ is teljesül. De akkor $\mathbf{C}$ zártsága miatt $\overline{L_1} \in \mathbf{C}$, és ebből következik, hogy $L_1 \in \mathbf{coC}$. Tehát $\mathbf{coC}$ is zárt a v-beli visszavezetésekre nézve. □

A fenti tétel alapján adódik az alábbi eredmény.

3.28. Tétel

Legyen $\mathbf{C}$ eldöntési problémák, v pedig kiszámítható függvények egy osztálya úgy, hogy $\mathbf{C}$ zárt a v-beli visszavezetésekre nézve. Ha van olyan $L \in \mathbf{C}$ nyelv ami $\mathbf{coC}$-teljes a v-beli visszavezetésekre nézve, akkor $\mathbf{C} = \mathbf{coC}$.

Bizonyítás. Legyen L egy $\mathbf{C}$-beli nyelv ami $\mathbf{coC}$-teljes a v-beli visszavezetésekre nézve. Legyen $L' \in \mathbf{coC}$. Mivel $L' \leq_v L$, $L \in \mathbf{C}$ és $\mathbf{C}$ zárt a v-beli visszavezetésekre nézve, azt kapjuk $L' \in \mathbf{C}$. Következik tehát, hogy $\mathbf{coC} \subseteq \mathbf{C}$. A másik irányú tartalmazás hasonló gondolatmenet alapján következik felhasználva, hogy $\mathbf{co}(\mathbf{coC}) = \mathbf{C}$, $\overline{L}$ egy olyan $\mathbf{coC}$-beli nyelv, ami $\mathbf{C}$-teljes (3.26. tétel) és $\mathbf{coC}$ zárt a v-beli visszavezetésekre nézve (3.27. tétel). □

Mivel $\mathbf{NP}$ zárt a polinom idejű visszavezetésekre nézve (3.4. tétel), az előző tétel alapján adódik, hogy ha sikerülne megmutatni, hogy $\textsc{Unsat} \in \mathbf{NP}$, akkor teljesülne, hogy $\mathbf{NP} = \mathbf{coNP}$.

3.2. Tárbonyolultság

A jegyzet következő fejezetében tárbonyolultsággal kapcsolatos néhány fontosabb eredményt ismertetünk. Az alapvető különbség az idő- és a tárbonyolultság között az, hogy a tár újra felhasználható. Könnyen lehet adni egy olyan Turing-gépet, ami nagyon sokáig dolgozik, de a bemenet hosszának függvényében csak lineáris számú cellát használ fel a szalagjain. További érdekesség még, hogy a szublineáris tárbonyolultságnak is van értelme, persze csak akkor, ha a

bemenet hosszát nem számoljuk bele a felhasznált tár nagyságába. Összehasonlításképpen, a szublineáris időbonyolultságnak nincs gyakorlati jelentősége, hiszen ebben az esetben a Turing-gép nem is olvassa végig a bemenetet, tehát a bemenet redundáns információkat tartalmazhat.

A fentiek értelmében a tárbonyolultságot egy speciális, úgynevezett off-line Turing-gépen vizsgáljuk.

3.29. Definíció

Off-line Turing-gépnek nevezünk egy olyan többszalagos Turing-gépet, mely a bemenetet tartalmazó szalagot csak olvashatja, a többi, úgynevezett munkaszalagra pedig írhat is. Az off-line Turing-gép tárigényébe csak a munkaszalagokon felhasznált terület számít be.

A fejezet további részében Turing-gép alatt mindig off-line Turing-gépet értünk. A következőkben definiálunk néhány fontos tárbonyolultsággal kapcsolatos problémaosztályt.

3.30. Definíció

$$\mathbf{SPACE}(f(n)) = \{L \mid L \text{ eldönthető } \mathcal{O}(f(n)) \text{ tárigényű determinisztikus Turing-géppel }\},$$

$$\mathbf{NSPACE}(f(n)) = \{L \mid L \text{ eldönthető } \mathcal{O}(f(n)) \text{ tárigényű nemdeterminisztikus Turing-géppel }\}.$$

Ezek után a **P** és az **NP** osztályok tárbonyolultsági megfelelőit az alábbi módon definiálhatjuk.

3.31. Definíció

$\mathbf{PSPACE} = \bigcup_{k>0} \mathbf{SPACE}(n^k)$ és $\mathbf{NPSPACE} = \bigcup_{k>0} \mathbf{NSPACE}(n^k)$.

Korábban említettük, hogy szublineáris tárbonyolultságnak is van értelme. Valójában az alábbi tárbonyolultsági osztályokra lesz szükségünk.

3.32. Definíció

$\mathbf{L} = \mathbf{SPACE}(\log_2 n)$ és $\mathbf{NL} = \mathbf{NSPACE}(\log_2 n)$.

3.2.1. Savitch tétele

Mint arról korábban már volt szó, egy nemdeterminisztikus Turing-gép determinisztikussal való szimulációja (jelenlegi tudásunk szerint) exponenciális időigény romlást eredményez. A tárbonyolultság esetében viszont bizonyított, hogy a determinisztikus Turing-gépek a tárbonyolultság négyzetes romlása árán képesek szimulálni a legalább logaritmikus tárbonyolultságú nemdeterminisztikus Turing-gépeket. Ebben a részben ezt az eredményt ismertetjük.

3.33. Tétel (Savitch tétele)

Ha $f(n) \geq \log n$, akkor $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}(f^2(n))$.

Bizonyítás. Legyen $f(n)$ a tétel feltételeit kielégítő függvény, és tekintsünk egy $L \in \mathbf{NSPACE}(f(n))$ nyelvet. Tegyük fel, hogy L-et eldönti az M $\mathcal{O}(f(n))$ tárigényű nemdeterminisztikus Turing-gép. Megadunk egy olyan M' $\mathcal{O}(f^2(n))$ tárigényű determinisztikus Turing-gépet, ami szintén L-et dönti el. Először is feltesszük, hogy amikor M elfogad egy szót, akkor azt úgy teszi, hogy mielőtt megáll a q_i állapotban, letörli a szalagjáról a nem $\sqcup$ szimbólumokat. Így M-nek csak egy elfogadó konfigurációja van, jelöljük ezt $c_{elfogadó}$-val.

Legyen w egy n hosszú szó. Jelöljük M kezdőkonfigurációját a w-n $c_{kezdő}$-vel. Az M *konfigurációs gráfjának* a w szón azt a C_w gráfot nevezzük, melynek csúcsai M konfigurációi a w-n, és a c_1 és c_2 csúcsok között pontosan akkor van él, ha $c_1 \vdash_M c_2$. Mivel M egy $\mathcal{O}(f(n))$ tárkorlátos gép, a C_w gráf minden konfigurációja legfeljebb $\mathcal{O}(f(n))$ méretű. Valóban, M legfeljebb $\mathcal{O}(f(n))$ méretű szalagot használhat fel a munkaszalagjain, és mivel a bemenetet tartalmazó szalag nem változik, itt elég csak a fej pozícióját eltárolni a konfigurációkban, ami megtehető $\mathcal{O}(\log n)$ tárral. Ezért C_w mérete $2^{\mathcal{O}(f(n))}$. Megadható továbbá egy olyan d konstans, hogy tetszőleges $i \in \mathbb{N}$ számra, C_w-ben legfeljebb 2^{di} darab i méretű konfiguráció van. Nyilvánvaló, hogy tetszőleges i-re, M' fel tudja sorolni az egyik munkaszalagján az M összes legfeljebb i méretű konfigurációját.

M'-nek tehát a C_w gráfban kellene keresnie két kitüntetett csúcs, nevezetesen a $c_{kezdő}$ és a $c_{elfogadó}$ között egy utat úgy, hogy a keresés során legfeljebb $\mathcal{O}(f^2(n))$ méretű tár kerüljön felhasználásra. Ehhez először definiálunk egy négy változós Boole-függvényt az alábbi módon. Minden $c_1, c_2 \in C_w$ konfigurációra és $i, t \in \mathbb{N}$ számra, legyen $\mathbf{ELÉR}(c_1, c_2, i, t) = igaz$ akkor és csak akkor, ha C_w-ben van egy legfeljebb t hosszú út c_1-ből c_2-be úgy, hogy az érintett konfigurációk mérete legfeljebb i. Nem nehéz meggondolni, hogy $\mathbf{ELÉR}(c_1, c_2, i, t)$ ponto-

san akkor *igaz*, ha van olyan legfeljebb i méretű $c \in C_w$ konfiguráció, melyre **ELÉR**$(c_1, c, i, \lceil \frac{t}{2} \rceil)$=*igaz* **és ELÉR**$(c, c_2, i, \lceil \frac{t}{2} \rceil)$=*igaz*. Ezek alapján az **ELÉR** függvény kiszámolható az 1. algoritmussal.

Algoritmus 1 Az **ELÉR** függvény kiszámítása

Bemenet: $c_1, c_2 \in C_w, i, t \in \mathbb{N}$;
Kimenet: **ELÉR**(c_1, c_2, i, t);
if $t = 1$ **then**
  if $c_1 = c_2 \vee (c_1, c_2)$ él C_w-ben **then**
    ELÉR(c_1, c_2, i, t):=*igaz*; **stop**
  else
    ELÉR(c_1, c_2, i, t):=*hamis*; **stop**
  end if
else
  for all $c \in C_w$ legfeljebb i méretű konfigurációra **do**
    if ELÉR$(c_1, c, i, \lceil \frac{t}{2} \rceil) \wedge$ **ELÉR**$(c, c_2, i, \lceil \frac{t}{2} \rceil)$ **then**
      ELÉR(c_1, c_2, i, t):=*igaz*;
    end if;
  end for;
  ELÉR(c_1, c_2, i, t):=*hamis*;
end if.

Tehát a $w \in L(M)$ kérdés eldöntéséhez M'-nek elég lenne kiszámítania az **ELÉR**$(c_{kezdő}, c_{elfogadó}, i, t)$-t megfelelő i, t $f(n)$-től függő értékekre. Sajnos azonban M' nem ismeri explicite az $f(n)$ értéket. Ezért M' a következőket teszi. Egy végtelen ciklusban, minden $i = 1, 2, 3, \ldots$ számra ellenőrzi, hogy **ELÉR**$(c_{kezdő}, c_{elfogadó}, i, 2^{di})$ értéke *igaz*-e. Ha nem, akkor megnézi, hogy egyáltalán elérhető-e a kezdőkonfigurációból i méretű konfiguráció C_w-ben. Ha nem érhető el ilyen konfiguráció, akkor elutasítja a bemenetet. Ily módon M'-t a 2. algoritmus szerint konstruáljuk meg. Világos, hogy az így megvalósított M' q_i-ben áll meg, ha $w \in L(M)$, és q_n-ben áll meg egyébként.

A bizonyítás befejezéséhez meg kell még vizsgálnunk a 1. és 2. algoritmusok tárigényét (belátható, hogy ezen algoritmusokat megvalósító M' tárigénye nem lesz nagyságrendekkel nagyobb, mint a fenti algoritmusoké).

Először is, a legnagyobb 2^{di}-érték, melyre **ELÉR** meghívásra kerül a 2. algoritmusban $2^{\mathcal{O}(f(n))}$-nél nem nagyobb, így a 1. algoritmusban szereplő rekurzióban a legnagyobb mélység legfeljebb $\log_2 2^{\mathcal{O}(f(n))} = \mathcal{O}(f(n))$. Továbbá egy rekurzív hívás tárigénye szintén $\mathcal{O}(f(n))$, így a két algoritmus együttes tárigénye $\mathcal{O}(f^2(n))$, amit bizonyítani akartunk. □

Ha meggondoljuk, hogy minden polinom függvény négyzete is polinom függvény,

Algoritmus 2 Egy $f(n)$ tárkorlátos nemdeterminisztikus Turing-géppel ekvivalens $\mathcal{O}(f^2(n))$ tárkorlátos determinisztikus Turing-gép működésének algoritmusa

Bemenet: M $f(n)$ tárkorlátos nemdeterminisztikus Turing-gép;
for $i = 1, 2, 3, \ldots$ **do**
  if ELÉR$(c_{kezdő}, c_{elfogadó}, i, 2^{di})$ **then**
    Megállás q_i-ben; **STOP**
  else
    if $\neg\exists c \in C_w$ i méretű konfiguráció, hogy **ELÉR**$(c_{kezdő}, c, i, 2^{di})$ **then**
      Megállás q_n-ben; **STOP**.
    end if
  end if
end for.

azonnal következik az alábbi állítás.

3.34. Következmény

PSPACE = **NPSPACE**.

3.2.2. PSPACE-teljes problémák

Most olyan problémákat vizsgálunk, melyek teljesek a **PSPACE** osztályra nézve (ebben a fejezetben a **PSPACE**-teljesség illetve nehézség alatt a polinom idejű visszavezetésekre való teljességet illetve nehézséget értjük). Az első ilyen probléma a QSAT probléma, ami a SAT probléma egyfajta általánosítása. Teljesen kvantifikált Boole formulának (TKBF-nek) nevezünk egy olyan formulát, melyben minden ítéletváltozó értéke univerzális ($\forall$) vagy egzisztenciális ($\exists$) kvantorral kötött, ráadásul úgy, hogy a formula két részre osztható: elöl vannak a kvantorok és utána egy kvantormentes rész. Ily módon minden kvantor hatásköre a saját pozíciójától a formula végéig tart.

3.35. Példa

Tekintsük a következő formulákat:

- $\varphi_1 = \exists x_1 \forall x_2 (x_1 \to x_2)$
- $\varphi_2 = \exists x_1 \forall x_2 ((\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2))$

Ekkor φ_1 értéke *igaz*, hiszen igaz a következő állítás: van olyan igazságértéke az x_1 változónak, hogy akárhogyan is választjuk az x_2 igazságértékét, a φ_1 értéke *igaz* lesz. Valóban, legyen az x_1 értéke *hamis*. Ekkor

az implikáció tulajdonságai miatt akárhogy választjuk meg az x_2 értékét, φ_1 értéke *igaz* lesz.
Ugyanakkor φ_2 értéke *hamis*, mert akár *igaz* akár *hamis* az x_1 értéke, nem teljesül az, hogy az x_2 tetszőleges igazságértékére *igaz* a φ_2.

Ezek után a QSAT probléma a következő:

3.36. Definíció

QSAT=$\{\langle\varphi\rangle \mid \varphi$ *igaz* teljesen kvantifikált Boole-formula$\}$.

Az alábbi tétel szerint a QSAT a legnehezebb problémák közé tartozik a **PSPACE** osztályon belül.

3.37. Tétel

QSAT **PSPACE**-teljes.

Bizonyítás. először azt mutatjuk meg, hogy QSAT $\in$ **PSPACE**. Ehhez tekintsünk egy φ TKBF-et. Legyen **ÉRTÉK** az az egyváltozós függvény, melyre **ÉRTÉK**$(\varphi) = igaz$ pontosan akkor, ha $\langle\varphi\rangle \in$ QSAT. Megadunk egy algoritmust, ami polinom tárat felhasználva számolja ki az **ÉRTÉK**(φ) függvényt. Legyen $\varphi = Qx\psi$, ahol $Q \in \{\exists, \forall\}$. Jelöljük ψ^i-vel (rendre ψ^h-val) azt a formulát, amit úgy kapunk, hogy ψ-ben minden x változót *igaz*-ra (rendre *hamis*-ra) cserélünk. Belátható, hogy a 3. algoritmus valóban az **ÉRTÉK**(φ)-t számolja ki. Az algoritmus lényege, hogy a $Qx\psi$ értékét úgy számolja ki, hogy rekurzív módon kiszámolja a ψ^i és a ψ^h értékét, majd ezen értékek alapján, attól függően, hogy $Q = \exists$ vagy $Q = \forall$, meghatározza a $Qx\psi$ értékét is.

Az 3. algoritmus tárigénye a formula méretébe lineáris. Valóban, a rekurzió mélysége legfeljebb akkora mint a formulában szereplő változók száma, míg a rekurzió minden szintjén egy változó igazságértékét kell csak tárolni.

Most azt mutatjuk meg, hogy QSAT **PSPACE**-nehéz. Ehhez legyen $L \subseteq \Sigma^*$ egy tetszőleges **PSPACE**-beli nyelv. Megmutatjuk, hogy $L \leq_p$ QSAT. Legyen M egy n^k $(k \geq 1)$ tárigényű L-et eldöntő Turing-gép és w az M egy tetszőleges n $(n \in \mathbb{N})$ hosszú bemenete. Megadunk egy φ TKBF-et úgy, hogy $w \in L(M)$ akkor és csak akkor, ha $\langle\varphi\rangle \in$ QSAT. Tudjuk, hogy M-nek legfeljebb $h = 2^{dn^k}$ különböző konfigurációja lehet a w-n egy alkalmasan választott d konstansra.

Algoritmus 3 Az **ÉRTÉK** függvény kiszámítása polinom tárral

Bemenet: φ TKBF;
Kimenet: **ÉRTÉK**(φ);
if φ nem tartalmaz változót **then**
  Értékeljük ki φ-t;
  ÉRTÉK(φ) := φ értéke
end if
if $\varphi = \exists x\psi$ **then**
  if **ÉRTÉK**(ψ^i) = *igaz* **vagy** **ÉRTÉK**(ψ^h) = *igaz* **then**
    ÉRTÉK(φ) := *igaz*
  else
    ÉRTÉK(φ) := *hamis*
  end if
end if
if $\varphi = \forall x\psi$ **then**
  if **ÉRTÉK**(ψ^i) = *igaz* **és** **ÉRTÉK**(ψ^h) = *igaz* **then**
    ÉRTÉK(φ) := *igaz*
  else
    ÉRTÉK(φ) := *hamis*
  end if
end if

Ezért feltehetjük, hogy M, mivel determinisztikus, legfeljebb h lépésben elfogadja vagy elutasítja w-t. Tegyük fel, hogy M letörli a szalagját, mielőtt q_i-be lép (ez feltehető, hisz minden M-hez megadható egy ekvivalens M', ami így viselkedik, mielőtt megáll). Ekkor M-nek pontosan egy elfogadó konfigurációja van. Jelölje $c_{\text{kezdő}}$ és $c_{\text{elfogadó}}$ rendre az M kezdő- és elfogadó konfigurációját w-n.

Az M c konfigurációinak reprezentálásához egymástól diszjunkt $C = \{c_{j,a} \mid j \in [n^k], a \in Q \cup \Gamma\}$ változóhalmazokat használunk. Ezeket a halmazokat *konfiguráció azonosító halmazoknak* (röviden KH-knak) fogjuk nevezni. A 3.8. tétel bizonyításában már látott módon a $c_{j,a}$ ítéletváltozó az „A c konfiguráció j-ik betűje az a szimbólum" állítást fogja jelöli. Világos, hogy a C-beli változók egy I interpretációja akkor és csak akkor írja le egy legális konfigurációját az M-nek,

- ha minden $j \in [n^k]$-ra pontosan egy olyan $a \in Q \cup \Gamma$ van, hogy $I(c_{j,a}) =$ *igaz* és
- ha pontosan egy $j \in [n^k]$-ra igaz, hogy $I(c_{j,a}) =$ *igaz* valamely $a \in Q$-ra.

Ha például a C KH-val az aqb konfigurációt akarjuk reprezentálni, akkor a C-beli változók azon interpretációja lesz a helyes reprezentáció, mely a $c_{1,a}$, $c_{2,q}$ és $c_{3,b}$ változóhoz igazat, a többihez pedig hamisat rendel. A továbbiakban

jelölje $\exists C$ a C-beli változók egzisztenciális kvantifikálását. Akkor például a $\varphi = \exists C(c_{1,a} \wedge c_{2,q} \wedge c_{3,b} \bigwedge_{c \in C, c \notin \{c_{1,a}, c_{2,q}, c_{3,b}\}} \neg c)$ formula pontosan azt írja le, hogy létezik a C-beli változók olyan interpretációja, ami az aqb konfigurációt reprezentálja.

A bizonyítás folytatásaként a következőket tesszük. Tetszőleges $c, d \in \mathcal{C}_M$ konfigurációkat reprezentáló C, D KH-kra és $t \geq 1$-re megadunk egy $\varphi_{C,D,t}$ TQBF-et, hogy $\varphi_{C,D,t}$ akkor és csak akkor *igaz*, ha M el tud jutni c-ből d-be legfeljebb t lépésben. Ha ezzel megvagyunk, akkor már könnyen meg fogjuk tudni adni a keresett φ-t a $\varphi_{C_{\text{kezdő}}, C_{elfogad}, h}$ formula segítségével, ahol $C_{\text{kezdő}}, C_{elfogad}$ rendre a $c_{\text{kezdő}}, c_{elfogad}$ konfigurációkat reprezentáló KH-k.

Használni fogjuk a következő jelöléseket. Tetszőleges C, D KH-kra $\forall C$ jelölje az összes C-beli változó univerzális kvantifikálását. Továbbá $C = D$ egy olyan formulát jelöljön, ami akkor és csak akkor *igaz* egy interpretációban, ha tetszőleges $j \in [n^k]$-ra és $a \in Q \cup \Gamma$-ra, $c_{j,a}$ igazságértéke ugyanaz, mint $d_{j,a}$ igazságértéke.

Ezután legyen

$$\varphi_{C,D,t} = \exists M \forall C_1 \forall C_2((C_1 = C \wedge C_2 = M) \vee (C_1 = M \wedge C_2 = D) \rightarrow \varphi_{C_1, C_2, \lceil \frac{t}{2} \rceil}).$$

Ezzel a formulával tulajdonképpen azt írjuk le amit Savitch tételénél is láttunk: M a c-ből akkor és csak akkor tud eljutni legfeljebb t lépésben d-be, ha van olyan m konfiguráció, hogy M el tud jutni c-ből m-be legfeljebb $\lceil \frac{t}{2} \rceil$ lépésben és m-ből d-be legfeljebb $\lceil \frac{t}{2} \rceil$ lépésben.

Legyen $\varphi' = \exists C_{\text{kezdő}} \exists C_{elfogad}(\varphi_K \wedge \varphi_{C_{\text{kezdő}}, C_{elfogad}, h} \wedge \varphi_V)$ ahol φ_K és φ_V rendre azok a formulák, melyekben a literálok úgy vannak megadva, hogy azok a $\exists C_{\text{kezdő}}$ és $\exists C_{\text{elfogadó}}$ kvantorokkal együtt a $C_{\text{kezdő}}$ és $C_{\text{elfogadó}}$ változóinak értékét úgy határozzák meg, hogy azok az M kezdő és elfogadó konfigurációját írják le. Legyen végül a keresett φ a φ' prenex alakra hozva.

Belátható, hogy $w \in L$ akkor és csak akkor, ha $\langle \varphi \rangle \in$ QSAT. Be kell még látnunk, hogy φ polinom időben megkonstruálható a w és M ismeretében. Ha megfigyeljük a $\varphi_{C,D,t}$ formulát, akkor láthatjuk, hogy a megkonstruálásának az időigénye a következőktől függ. Egyrészt attól, hogy mennyi a $\varphi_{C_1, C_2, \lceil \frac{t}{2} \rceil}$-n kívüli rész konstrukciójának időigénye. Másrészt attól, hogy hányszor kell elvégezni ezt a konstrukciót. Világos, hogy a $\varphi_{C,C',\lceil \frac{t}{2} \rceil}$-n kívüli részt $\mathcal{O}(n^k)$ időben meg lehet konstruálni. A konstrukciót pedig $\mathcal{O}(\log_2 t)$-szer

QSAT mint kétszemélyes játék

A QSAT azon megszorítása amikor csak olyan bemeneteket tekintünk ahol a kvantorok alternálnak, az első és az utolsó kvantor a $\exists$, és a kvantormentes rész egy KNF, felfogható úgy, mint az alábbi kétszemélyes játék. Tegyük fel, hogy adott egy ilyen alakú φ TKBF. Az első játékos választja meg a páratlan sorszámú változók értékét, és a célja a formula igazzá tétele. A második választja meg a páros sorszámú változók értékét, és a célja a formula hamissá tétele. Belátható, hogy az első játékosnak akkor és csak akkor van nyerő stratégiája ebben a játékban, ha $\langle \varphi \rangle \in$ QSAT.

kell elvégezni. Mivel $h = 2^{dn^k}$, azt kapjuk, hogy a $\varphi_{C_{\text{kezdő}},C_{\text{elfogadó}},h}$ formulát $\mathcal{O}(n^{2k})$ időben meg lehet konstruálni. Mivel φ' prenex alakra hozása, valamint a φ_K és φ_V konstrukciója szintén elvégezhető $\mathcal{O}(n^{2k})$ időben, a tétel állítását bebizonyítottuk. □

Itt jegyezzük meg, hogy a fenti tétel bizonyításában a $\varphi_{C,D,t}$ konstrukciója helyett a $\varphi_{C,D,t} = \exists M(\varphi_{C,M,\lceil \frac{t}{2} \rceil} \wedge \varphi_{M,D,\lceil \frac{t}{2} \rceil})$ rekurzív definíció nem felelne meg a céljainknak, mert ekkor a $\varphi_{C_{\text{kezdő}},C_{\text{elfogadó}},h}$ formula mérete exponenciális lenne, hiszen a formula mérete minden rekurziós lépéssel megduplázódna. Így ezt a formulát nyilvánvalóan nem lehetne polinom idő alatt megkonstruálni.

Belátható, hogy A QSAT probléma akkor is **PSPACE**-teljes marad ha a bemenetet az alábbi módon szorítjuk meg: a kvantorok alternálnak, az első és az utolsó kvantor a $\exists$, és a kvantormentes rész pedig KNF. Ez a probléma, ahogy az a keretes megjegyzésben olvasható, felfogható egy kétszemélyes játékként. A következőkben megmutatjuk, hogyan lehet a QSAT ezen változatát polinom időben visszavezetni egy másik **PSPACE**-beli kétszemélyes játékra, a FÖLDRAJZI JÁTÉK problémára.

3.38. Definíció

A FÖLDRAJZI JÁTÉK a következő kétszemélyes játék. Adott egy G irányított gráf és annak egy p csúcsa. A két játékos p-ből kiindulva felváltva jelöli meg a G még meg nem jelölt csúcsait, mindig az utoljára megjelölt csúcsból elérhető csúcsok közül választva. Az veszít, aki nem tud további csúcsot megjelölni.

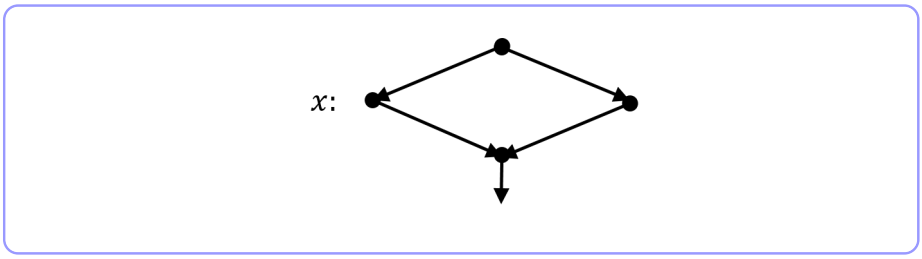
3.39. Tétel

FÖLDRAJZI JÁTÉK **PSPACE**-teljes.

Bizonyítás. Azt, hogy FÖLDRAJZI JÁTÉK $\in$ **PSPACE** hasonló módon lehet látni, mint azt, hogy QSAT $\in$ **PSPACE**.

Ahhoz, hogy a **PSPACE**-nehézséget belássuk elég megmutatni, hogy QSAT $\leq_p$ FÖLDRAJZI JÁTÉK. Ehhez legyen φ egy TQBF az alábbi alakban megadva. $\varphi = \exists x_1 \forall x_2 \exists x_3 \ldots \exists x_k \psi$ $(k \geq 1)$, ahol $\psi = c_1 \wedge \ldots \wedge c_m$ $(m \geq 1)$ egy KNF (emlékezzünk vissza, hogy a QSAT ilyen alakú bemenetek esetében is **PSPACE**-teljes). Konstruáljuk meg a G_φ gráfot a következőképpen. Először minden φ-beli x változóhoz elkészítünk egy a 3.11. ábrán látható részgráfot.

Ezután legyen G_φ a 3.12. ábrán látható gráf. Itt baloldalt a φ-beli változókhoz elkészített részgráfok vannak összekötve. Jobboldalt van egy csúcs ami ψ-vel van címkézve. Az ebből kiinduló élek olyan csúcsokba vezetnek, melyek rendre a ψ klózaival vannak címkézve. Minden klóznak megfelelő csúcsból annyi él vezet



3.11. ábra. Az x változónak megfelelő részgráf

ki, ahány literál van az adott klózban. Minden literállal címkézett csúcsból egy él indul ki a következőképpen. Ha a csúcs x-el van címkézve, akkor a belőle kiinduló él az x-nek megfelelő baloldali részgráf baloldali csúcsába vezet. Ha a csúcs $\neg x$-el van címkézve, akkor a belőle kiinduló él az x-nek megfelelő baloldali részgráf jobboldali csúcsába vezet.

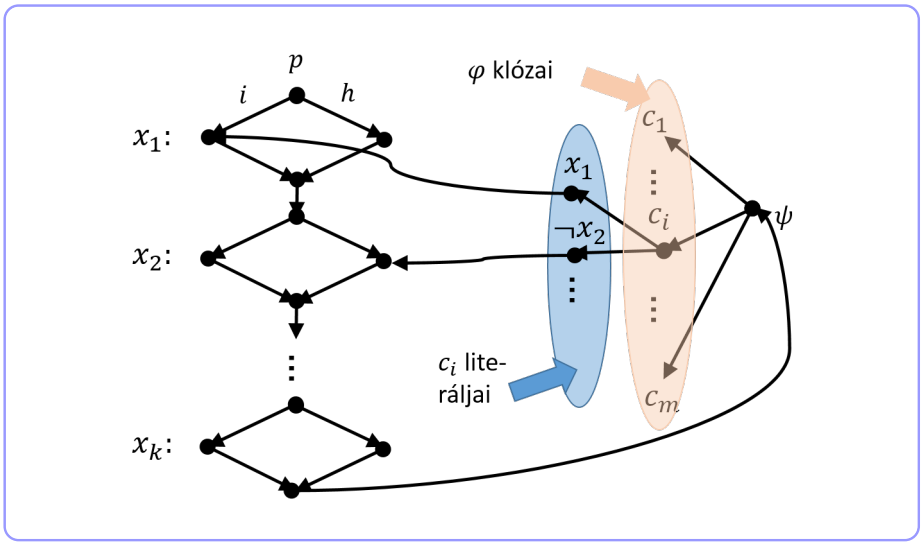
Nézzük meg, hogyan zajlik egy játék a G_φ-ben! A páratlan sorszámú váltóknak megfelelő részgráfok legfelső csúcsaiban az első játékos választja az irányt (azaz kezdetben ő választ egy csúcsot a p-ből kiindulva), a páros sorszámúaknak megfelelőkben pedig a második játékos. Mivel k páratlan, a gráf alján lévő csúcsot csak a második játékos választhatja. Ezután az első játékos csak a ψ-vel címkézett csúcsot választhatja. Itt a második játékos választ egy c klóznak megfelelő csúcsot, majd az első játékos kiválaszt egy c-ben lévő literálnak megfelelő csúcsot. Itt két lehetőség van: vagy tud a második játékos még korábban nem választott csúcsot választani vagy nem. A G_φ konstrukciója miatt a második játékos pontosan akkor tud csúcsot választani, ha a következők egyike teljesül:

A földrajzi játék

Ez a játék eredetileg a jegyzetben definiálthoz képest a következőképpen néz ki. Két játékos felváltva mond városneveket úgy, hogy a következő játékosnak mindig olyan, még el nem hangzott nevet kell mondania, ami a legutoljára mondott név utolsó betűjével kezdődik. Ha vesszük az összes városok gráfját, melyben egy városnévből azokba a városokba vezet él, melyek kezdőbetűje megegyezik a kiindulási név utolsó betűjével, akkor megkapjuk a jegyzetben definiált FÖLDRAJZI JÁTÉK problémát.

- Az aktuális csúcs címkéje x és az x-nek megfelelő részgráfban a baloldali csúcsot még nem jelöltük meg.
- Az aktuális csúcs címkéje $\neg x$ és az x-nek megfelelő részgráfban a jobboldali csúcsot még nem jelöltük meg.

Ennek megfelelően az első játékos úgy próbálja megválasztani a páratlan indexű ítéletváltozóknak megfelelő gráfokban az irányt, hogy később tudjon bármely



3.12. ábra. A φ-hez megkonstruált G_φ gráf

klóznak megfelelő csúcsból egy olyat választani, ami olyan csúccsal van összekötve, ami már korábban választásra került. Ezzel ellentétben a második játékos úgy próbálja megválasztani a páros indexű változóknak megfelelő részgráfokban az irányt, hogy később tudjon olyan c klóznak megfelelő csúcsot választani, hogy a c-ből elérhető csúcsok mindegyike még korábban ki nem választott csúcsokkal legyen összekötve.

Belátható, hogy az első játékosnak pontosan akkor van nyerő stratégiája G_φ-ben, ha a φ-beli kétszemélyes játékban is van, azaz a következők teljesülnek. Az első játékos meg tudja adni a páratlan indexű ítéletváltozók értékét úgy, hogy később akárhogyan is választ a második játékos egy ψ-beli klózt, az első játékos ki tud választani abban a klózban egy olyan literált, ami *igaz* lesz a megadott interpretációban.

Ebből következik, hogy a $\varphi \mapsto G_\varphi$ hozzárendelés a QSAT egy visszavezetése a FÖLDRAJZI JÁTÉKRA. Az is könnyen látható, hogy G_φ a φ méretében polinom időben megkonstruálható, és ezzel a tételt bebizonyítottuk. □

3.2.3. A szóproblémáról

Szóproblémának nevezzük azt a problémát, aminek a bemenete egy $G = (V, \Sigma, R, S)$ grammatika és egy $u \in \Sigma^*$ szó, továbbá a kérdés az, hogy $u \in L(G)$ teljesül-e. A szóproblémára tulajdonképpen a programozási nyelvek elemzésének absztrakciójaként is tekinthetünk, mivel számos programozási nyelv szintaxisa környezetfüggetlen nyelvtanra épül. Nem mellékes tehát, hogy a szópróbléma

eldöntése milyen bonyolultságú.

Reguláris grammatikák esetében lineáris, környezetfüggetlen grammatikák esetében pedig köbös időigényű algoritmus is ismert a szóprobléma eldöntésére. A környezetfüggő grammatikák esetében viszont a probléma már **PSPACE**-teljes, azaz kifejezetten nehéz problémának számít.

Mivel az általános nyelvtanok egyszerűen képesek szimulálni egy egyszalagos Turing-gépeket, kapjuk, hogy a szóprobléma az ő esetükben eldönthetetlen.

3.2.4. Logaritmikus tárbonyolultság

Ebben a fejezetben olyan problémákat vizsgálunk, melyek logaritmikus, azaz nagyon hatékony tárfelhasználás mellett dönthetők el. A fejezet központi problémája az ELÉRHETŐSÉG probléma, azaz annak eldöntése, hogy egy irányított gráfban van-e adott két csúcs között út (a pontos definíciót lásd a 63. oldalon).

Legyen $G = (V, E)$ egy irányított gráf és $s, t \in V$. Világos, hogy a 3.33. tétel bizonyításában alkalmazott 1. algoritmust felhasználva $\mathcal{O}(\log^2 |V|)$ tárigénnyel eldönthető, hogy van-e út G-ben s-ből t-be. Adódik tehát a következő eredmény.

3.40. Tétel

ELÉRHETŐSÉG $\in$ **SPACE**$(\log^2 n)$.

A továbbiakban azt szeretnénk megmutatni, hogy az ELÉRHETŐSÉG probléma **NL**-teljes. Ehhez azonban az eddig használt polinom idejű visszavezetések túl erősek, hiszen, ahogy az látni fogjuk, **NL** $\subseteq$ **P** (3.45. következmény), azaz polinom időben bármely **NL**-beli probléma triviálisan visszavezethető az ELÉRHETŐSÉG problémára. Mivel az a sejtés, hogy **L** $\subset$ **NL**, azaz van olyan logaritmikus tárral nemdeterminisztikus Turing-géppel megoldható probléma, amit determinisztikus Turing-géppel nem lehet logaritmikus tárral megoldani, a logaritmikus tárral kiszámítható visszavezetések jó választásnak tűnnek a további eredmények bizonyításához. Egy $f : \Sigma^* \to \Delta^*$ függvényt *logaritmikus tárral kiszámítható függvénynek* nevezünk, ha kiszámítható egy olyan, legalább három szalagos, logaritmikus táras off-line Turing-géppel, ami az utolsó szalagjára csak írhat, azaz azt nem olvashatja. Ekkor a tárfelhasználásba nem számít bele az első szalagon lévő bemenet, és az utolsó szalagra írt kimenet mérete. Az ilyen gépeket *lyukszalagos gépeknek* is nevezik.

3.41. Definíció

Legyen v a determinisztikus Turing-géppel logaritmikus tárral kiszámítható függvények osztálya. A v-szerinti visszavezetéseket *logaritmikus tárral való visszavezetéseknek* fogjuk hívni, és ha az L_1 logaritmikus tárral visszavezethető L_2-re, akkor ezt így jelöljük: $L_1 \leq_l L_2$.

Ebben a fejezetben tehát egy adott **C** bonyolultsági osztályra a **C**-teljesség illetve a **C**-nehézség alatt a logaritmikus tárral való visszavezetések szerinti **C**-teljességet illetve **C**-nehézséget fogjuk érteni.

Ahogy a **P** és **NP** zártak a polinom idejű visszavezetésekre, úgy az **L** és **NL** is zártak a logaritmikus tárral való visszavezetésekre nézve.

3.42. Tétel

L és **NL** zártak a logaritmikus tárral való visszavezetésre nézve.

Bizonyítás. Legyen L_1, L_2 két nyelv úgy, hogy $L_1 \leq_l L_2$. Először tegyük fel, hogy $L_2 \in \mathbf{NL}$. Legyen M_2 az L_2-t eldöntő nemdeterminisztikus logaritmikus táras, M pedig a visszavezetésben használt f függvényt kiszámoló determinisztikus logaritmikus táras Turing-gép. Most sajnos nem alkalmazhatjuk a 3.4. tétel bizonyításában látott módszert, miszerint tetszőleges bemenetre először futtatjuk M-et, majd a kapott szóra futtatjuk M_2-t, mert az így konstruált Turing-gép nem biztos, hogy logaritmikus táras lesz. Ehelyett legyen M_1 az a három szalagos Turing-gép, ami az alábbi módon működik. M_1 tetszőleges u szóra a következőket teszi.

1. A második szalagján egy bináris számlálóval nyomon követi, hogy az M_2 feje hányadik betűjét olvassa az $f(u)$ szónak. Legyen ez a szám i (értelemszerűen kezdetben $i = 1$).
2. Amikor M_2 lépne egyet, akkor M_1 az M működését a kezdőállapotból szimulálva előállítja a harmadik szalagon az $f(u)$ i-ik betűjét (de csak ezt a betűt, az $f(u)$ többi betűje nem kerül a harmadik szalagra).
3. Ezután M_1 szimulálja M_2 aktuális lépését a harmadik szalagon lévő betű felhasználásával és aktualizálja az első szalagon M_2 fejének újabb pozícióját.
4. Ha eközben M_1 azt látja, hogy M_2 elfogadó vagy elutasító állapotba lép, akkor M_1 is belép a saját elfogadó vagy elutasító állapotába, egyébként folytatja M_2 következő lépésének szimulációját.

Belátható, hogy az így vázolt M_1 az L_1-et dönti el, és a működése során csak logaritmikus méretű tárat használ, azaz $L_1 \in \mathbf{NL}$.

Ha ráadásul L_2 **L**-beli, akkor M_2 választható determinisztikus Turing-gépnek. De akkor M_1 is determinisztikus lesz amiből következik, hogy $L_1 \in \mathbf{L}$. □

Az előző tételt a 3.6. tétel bizonyításában látott módon felhasználva adódik a következő eredmény.

3.43. Következmény

Ha L **NL**-teljes és $L \in \mathbf{L}$, akkor $\mathbf{L} = \mathbf{NL}$.

Mivel az a sejtés, hogy $\mathbf{L} \subset \mathbf{NL}$, a fenti eredményből következik, hogy egy **NL**-teljes problémára sem létezik determinisztikus logaritmikus táras algoritmus. Így az alábbi eredmény miatt valószínűleg az ELÉRHETŐSÉG problémára sem.

3.44. Tétel

ELÉRHETŐSÉG **NL**-teljes.

Bizonyítás. Először azt mutatjuk meg, hogy ELÉRHETŐSÉG $\in$ **NL**. Legyen M egy nemdeterminisztikus Turing-gép, ami adott $G = (V, E)$ irányított gráfra és $s, t \in V$ csúcsokra a következőt teszi:

1. Ráírja s-et a második szalagra.
2. Ráírja a 0-t a harmadik szalagra.
3. Amíg a harmadik szalagon $|V|$-nél kisebb bináris szám van a következőket ismétli:
   (a) Legyen u a második szalagon lévő csúcs.
   (b) M nemdeterminisztikusan felír a második szalagon az u helyére egy u-ból elérhető v csúcsot.
   (c) Ha $v = t$, akkor M elfogadja a bemenetet, egyébként növeli a harmadik szalagon lévő számot binárisan eggyel.
4. Ha a harmadik szalagon lévő bináris szám egyenlő $|V|$-vel és nem volt korábban elfogadás, akkor M elutasítja a bemenetet.

Könnyen látható, hogy M $\mathcal{O}(\log |V|)$ tárat felhasználva azt dönti el, hogy van-e út s-ből t-be.

Azt, hogy ELÉRHETŐSÉG **NL**-nehéz a következőképpen bizonyítjuk. Legyen $L \in \mathbf{NL}$, megmutatjuk, hogy $L \leq_l$ ELÉRHETŐSÉG. Legyen M egy L-et eldöntő $\mathcal{O}(\log n)$ táras nemdeterminisztikus Turing-gép, és legyen u az M egy n hosszú bemenete. Belátható, hogy M minden konfigurációjának a mérete legfeljebb $c \cdot \log(n)$, valamely megfelelően választott c konstansra. Legyen G az M

konfigurációs gráfja az u-n. G megkonstruálható egy logaritmikus táras N determinisztikus Turing-géppel a következő módon. N szisztematikusan felsorolja az összes $c \cdot \log n$ hosszú szót az egyik szalagján és közben teszteli, hogy az M egy legális konfigurációját leíró szó van-e éppen a szalagon. Ha igen, akkor a szót kiírja a kimenetre. G élei (melyek most konfiguráció párok) hasonlóképpen felsorolhatók, tesztelhetők és a kimenetre írhatók.

Jelöljük rendre s-sel és t-vel az M kezdő- és elfogadó konfigurációját (ahogy korábban, most is feltehetjük, hogy pontosan egy elfogadó konfigurációja van M-nek). Világos, hogy $u \in M(L)$ akkor és csak akkor teljesül, ha G-ben van út s-ből t-be. Ezzel az L-et logaritmikus tárral visszavezettük a ELÉRHETŐSÉG-re. □

3.45. Következmény

NL $\subseteq$ **P**.

Bizonyítás. Legyen $L \in$ **NL** és M egy $\mathcal{O}(\log n)$ tárkorlátos nemdeterminisztikus Turing-gép ami L-et dönti el. Ekkor van olyan c konstans, hogy M-nek legfeljebb $p(n) = n2^{c \cdot \log n}$ méretű a konfigurációs gráfja egy n hosszú bemeneten. Ebből következik, hogy M-nek nem lehetnek $p(n)$-nél hosszabb számításai egy n hosszú bemeneten. Mivel $p(n)$ polinom függvény, az állításunkat bebizonyítottuk. □

Korábban már beszéltünk róla, hogy erős sejtés, miszerint **NP** $\neq$ **coNP**. A Savitch tétel következménye (3.34. következmény) és a determinisztikus bonyolultsági osztályok 3.1.4. fejezet elején tárgyalt tulajdonsága miatt kapjuk, hogy **NPSPACE** = **coNPSPACE**. Most megmutatjuk, hogy **NL** = **coNL** is teljesül.

3.46. Tétel (Immermann-Szelepcsényi tétel)

NL = **coNL**.

Bizonyítás. Először megmutatjuk, hogy $\overline{\text{ELÉRHETŐSÉG}} \in$ **NL**, vagyis azt, hogy nemdeterminisztikus logaritmikus tárkorlátos Turing-géppel eldönthető, hogy adott $G = (V, E)$ irányított gráf $s, t \in V$ csúcsaira igaz-e, hogy nincs út s-ből t-be. Később látni fogjuk, hogyan következik ebből a tétel állítása.

Tegyük fel, hogy ismerjük azon csúcsok c számát, melyek elérhetők s-ből. Először vázolunk egy logaritmikus tárkorlátos M nemdeterminisztikus Turing-gépet, ami a c ismeretében eldönti, hogy vajon igaz-e, hogy t nem érhető el s-ből. Utána megmutatjuk, hogyan tudjuk kiszámolni a c-t. Az M egy adott G, s, t bemenetre és c számra a következőt teszi. A G összes t-től különböző csúcsára nemdeterminisztikusan megsejti, hogy a csúcs elérhető-e s-ből. Ha egy v csúcsra azt sejti, hogy igen, akkor növeli egy d számláló értékét eggyel (a d értéke

kezdetben nulla), és egy nemdeterminisztikus logaritmikus táras számítási sorozattal ellenőrzi, hogy v tényleg elérhető-e s-ből. Ha nem érhető el v, akkor elutasítja a bemenetet. Amikor M minden csúcsot megvizsgált, akkor megnézi, hogy $d = c$ teljesül-e. Ha igen, akkor elfogadja a bemenetet, hiszen ekkor megtalálta az összes csúcsot, ami elérhető s-ből, és ezek között nem szerepelt a t. Egyébként pedig, vagyis amikor $d < c$ (ezért nem találta meg az összes elérhető csúcsot), elutasítja a bementet.

Most vázoljuk, hogy M hogyan tudja kiszámítani a fent említett c számot. Jelöljük d_i-vel ($0 \leq i \leq |V|$) azon csúcsok számát, melyek elérhetők s-ből legfeljebb d_i lépésben. Nyilvánvalóan $d_0 = 1$ és $d_{|V|-1} = c$. Továbbá, ha ismerjük a d_i ($0 \leq i \leq |V| - 2$) értéket, akkor M a következőképpen számolja ki d_{i+1}-t. M a G összes $v \neq s$ csúcsára a következőt teszi. Először egy d számláló értékét nullára állítja, majd a G összes u csúcsára nemdeterminisztikusan megsejti, hogy a csúcs elérhető-e s-ből legfeljebb i lépésben. Ha azt sejti, hogy igen, akkor egy nemdeterminisztikus logaritmikus tárigényű számítással ellenőrzi, hogy helyes-e a sejtés. Ha nem helyes, elutasítja a bemenetet, egyébként növeli a d-t eggyel és megnézi, hogy van-e (u, v) él G-ben. Ha igen, akkor növeli a c_{i+1} értékét eggyel és kezdi az új számítást a következő v-vel. Ha nincs ilyen él, akkor megy tovább a következő u-ra. Ha nem talált olyan u-t amiből vezetne él az aktuális v-be, akkor két eset van. Ha $d \neq c_i$, akkor az u-k nemdeterminisztikus kiválogatása során nem találtunk meg elég u-t, így ebben az esetben M elutasítja a bemenetet. Ha $d = c_i$, akkor az összes olyan u kiválasztásra került, ami elérhető legfeljebb i lépésben s-ből, azaz az aktuális v ebben az esetben nem érhető el $i+1$ lépésben s-ből. Ekkor c_{i+1} értéke nem változik, és M folytatja a számítást a következő v-vel. Az M fent vázolt működését részletesebben a 4. algoritmusban mutatjuk be.

Tudjuk tehát, hogy $\overline{\text{Elérhetőség}} \in \mathbf{NL}$. Mivel Elérhetőség **NL**-teljes (3.44. tétel), a 3.26. tétel alapján $\overline{\text{Elérhetőség}}$ **coNL**-teljes. Ebből, mivel a 3.42. tétel miatt **NL** zárt a logaritmikus táras visszavezetésre, a 3.28 tétel segítségével kapjuk, hogy $\mathbf{NL} = \mathbf{coNL}$. □

A fejezet befejezéseként azt mutatjuk meg, hogy 2sat **NL**-teljes.

3.47. Tétel

2sat **NL**-teljes.

Bizonyítás. Először azt mutatjuk meg, hogy $2\text{sat} \in \mathbf{NL}$. Ehhez azt látjuk be, hogy $\overline{2\text{sat}} \in \mathbf{NL}$, amiből a 3.46. tétel alapján következik az állításunk. Legyen φ a $\overline{2\text{sat}}$ egy bemenete. Konstruáljuk meg a következő G_φ gráfot. G_φ csúcsai a φ-beli változók és ezek tagadásai, az élei pedig a következőképpen adódnak. Minden $\alpha \vee \beta$ φ-beli klózra vegyük fel a $(\bar{\alpha}, \beta)$ és $(\bar{\beta}, \alpha)$ éleket G_φ-be, ahol a felülvonás a literálok komplementerét jelöli (egy x ítéletváltozó komplementere a $\neg x$, míg a $\neg x$ komplementere az x). Belátható, hogy G_φ logaritmikus tárral meg-

Algoritmus 4 Az ELÉRHETŐSÉG-et logaritmikus tárral eldöntő M nemdeterminisztikus Turing-gép

Bemenet: $G = (V, E), s, t \in V$;
Kimenet: *igen*, ha nincs út s-ből t-be, és *nem* egyébként;
$m \leftarrow |V| - 1$;
$c_0 \leftarrow 1$;
for all $i = 0$ to $m - 1$ **do**
  $c_{i+1} \leftarrow 1$;
  for all $v \in V, v \neq s$ **do**
    $d \leftarrow 0$
    for all $u \in V$ **do**
      Nemdeterminisztikusan hajtsuk végre vagy hagyjuk ki a következő lépéseket:
        Nemdeterminisztikusan keressünk legfeljebb i hosszú utat s-ből u-ba;
        if nem jutottunk u-ba **then**
          Kimenet: *nem*
        end if
        $d \leftarrow d + 1$;
        if $(u, v) \in E$ **then**
          $c_{i+1} \leftarrow c_{i+1} + 1$;
          Lépjünk ki ebből a ciklusból
        end if
    end for
    if $d \neq c_i$ **then**
      Kimenet: *nem*
    end if
  end for
end for
$d \leftarrow 0$;
for all $u \in V, u \neq t$ **do**
  Nemdeterminisztikusan hajtsuk végre vagy hagyjuk ki a következő lépéseket:
    Nemdeterminisztikusan keressünk legfeljebb m hosszú utat s-ből u-ba;
    if nem jutottunk u-ba **then**
      Kimenet: *nem*
    end if
    $d \leftarrow d + 1$
end for
if $d \neq c_m$ **then**
  Kimenet: *nem*
end if
Kimenet: *igen*

konstruálható. Vegyük észre, hogy mivel egy $\alpha \vee \beta$ klóz ekvivalens az $\bar{\alpha} \to \beta$ és $\bar{\beta} \to \alpha$ formulákkal, a G_φ élei tulajdonképpen a φ klózaival ekvivalens implikációs formulákat kódolják. Ebből már látható az is, hogy φ akkor és csak akkor kielégíthetetlen, ha van olyan kör G_φ-ben, amely érint egy x ítéletváltozóval címkézett csúcsot és egy $\neg x$-el címkézett csúcsot is. Válasszunk nemdeterminisztikusan egy x-el címkézett csúcsot G_φ-ben és futtassuk az ELÉRHETŐSÉG problémát logaritmikus tárral eldöntő nemdeterminisztikus Turing-gépet (lásd a 3.44. tétel bizonyítását) a $\langle G, x, \neg x\rangle$, majd a $\langle G_\varphi, \neg x, x\rangle$ bemeneteken. Ha a gép mindkét bemenetet elfogadja, akkor ez azt jelenti, hogy G_φ-ben van olyan kör, ami érint egy x ítéletváltozót és annak tagadását is. Ily módon a φ kielégíthetetlenségének eldöntését hatékonyan visszavezettük egy nemdeterminisztikus logaritmikus tárral eldönthető problémára, azaz beláttuk, hogy $\overline{\text{2SAT}} \in \mathbf{NL}$.

Most azt mutatjuk meg, hogy 2SAT **NL**-nehéz. Ehhez, a 3.26. és a 3.46. tételek alapján elég megmutatni, hogy $\overline{\text{2SAT}}$ **NL**-nehéz. A 3.42. tétel bizonyítása alapján könnyen belátható, hogy a logaritmikus tárkorlátos visszavezetések zártak a kompozícióra. Ezért a $\overline{\text{2SAT}}$ **NL**-nehézségének belátásához elég megmutatni, hogy ELÉRHETŐSÉG $\leq_l \overline{\text{2SAT}}$.

Legyen $\langle G, s, t\rangle$ az ELÉRHETŐSÉG probléma egy bemenete valamely $G = (V, E)$ irányított gráfra. Megadunk egy φ formulát úgy, hogy $\langle G, s, t\rangle \in$ ELÉRHETŐSÉG akkor és csak akkor, ha $\langle\varphi\rangle \in \overline{\text{2SAT}}$. Jelöljük $G' = (V', E')$-vel azt a gráfot, melyben az s és t csúcsokat rendre kicseréljük x-re és $\neg x$-re (ahol x egy új, a V-beli csúcsok címkéitől különböző címke). Ezek után legyen φ a következő formula: $\varphi = (x \vee x) \bigwedge_{(u,v)\in E'} (\bar{u} \vee v)$.

Először tegyük fel, hogy G-ben van út s-ből t-be. Ekkor van út G'-ben x-ből $\neg x$-be. Legyen egy ilyen G'-beli út az $u_1 u_2 \ldots u_k$ ($k \geq 2$) csúcssorozat, ahol $u_1 = x$ és $u_k = \neg x$. Megmutatjuk, hogy φ kielégíthetetlen. Az világos, hogy csak olyan I elégítheti ki φ-t, melyre $I(x) = igaz$. Másrészt, mivel van $u_1 u_2 \ldots u_k$ út G'-ben, a φ definíciója miatt az $\bar{u}_1 \vee u_2 = \neg x_1 \vee u_2, \bar{u}_2 \vee u_3 \ldots, \bar{u}_{k-1} \vee u_k = \bar{u}_{k-1} \vee \neg x$ klózok mind szerepelnek φ-ben. Könnyen látható, hogy ha $I(x) = igaz$, akkor ezen klózok közül valamelyiknek *hamis*-nak kell lennie I-ben, azaz φ valóban kielégíthetetlen.

Most tegyük fel, hogy G-ben nincs út s-ből t-be, azaz G'-ben nincs út x-ből $\neg x$-be. Legyen $V_{\neg x}$ azon V' beli csúcsok halmaza, melyekből elérhető $\neg x$ és legyen $V_x = V' - V_{\neg x}$. Ekkor a feltevésünk szerint nincs olyan él G'-ben, ami V_x-beli csúcsból $V_{\neg x}$-beli csúcsba vezet. Vegyük azt az I interpretációt, ami az összes V_x-beli csúcsnak megfelelő φ-beli ítéletváltozóhoz *igaz*-at, a $V_{\neg x}$-beli csúcsoknak megfelelő ítéletváltozókhoz pedig *hamis*-at rendel. Mivel $x \in V_x$, $I(x) = igaz$ és ezért $I \models (x \vee x)$. Továbbá, mivel nincs olyan (u, v) él G'-ben, hogy $u \in V_x$ és $v \in V_{\neg x}$, nem lehet olyan $\bar{u} \vee v$ klóz φ-ben, hogy $I(u) = igaz$ és $I(v) = hamis$. Ebből azt kapjuk, hogy $I \models \varphi$, azaz φ kielégíthető. □

3.3. Összefoglalás

3.3.1. Könnyű vagy nehéz?

A jegyzetben számos olyan problémát láttunk, amit nehéznek tekintünk, mert tudjuk, hogy **NP**-teljes. Ugyanakkor észrevehetjük azt is, hogy némely nehéz probléma egyszerű módosítással könnyűvé válik. Például a 3SAT nehéz, míg a 2SAT már könnyű, adott két csúcs közötti Hamilton-út keresés egy gráfban nehéz, de tetszőleges út keresése már könnyű, korlátozott feszítőfa keresése egy gráfban nehéz, de egy tetszőleges feszítőfa hatékonyan megkonstruálható. Ha tehát a gyakorlatban azt tapasztaljuk, hogy egy adott problémát nem tudunk hatékonyan megoldani, akkor érdemes végiggondolni, hogy nem tudjuk-e a problémát egyszerűbben, de még mindig céljainknak megfelelően megfogalmazni.

3.3.2. A PSPACE osztályon túl

Legyen $(\mathbf{N})\mathbf{EXPTIME} = \bigcup_{k=1}^{\infty} (\mathbf{N})\mathbf{TIME}(2^{n^k})$. A jegyzetben látott bonyolultsági osztályok egymáshoz való viszonyait az alábbiakban foglaljuk össze.

3.48. Következmény

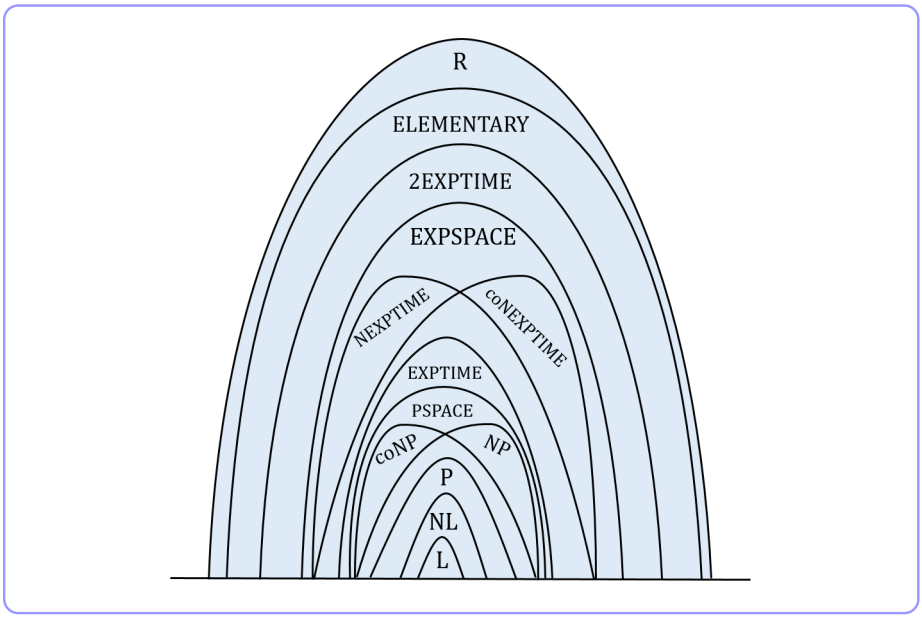
$\mathbf{L} \subseteq \mathbf{NL} = \mathbf{coNL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME}$.

Bizonyítás. $\mathbf{L} \subseteq \mathbf{NL}$ és $\mathbf{P} \subseteq \mathbf{NP}$ definíció szerint teljesül. Az $\mathbf{NL} = \mathbf{coNL}$ egyenlőséget a 3.46. tételben bizonyítottuk. Az $\mathbf{NL} \subseteq \mathbf{P}$ tartalmazást és a $\mathbf{PSPACE} = \mathbf{NPSPACE}$ egyenlőséget rendre a 3.45. és a 3.34. következményekben láttuk. $\mathbf{NP} \subseteq \mathbf{PSPACE}$ azért teljesül, mert egy polinom időigényű Turing-gép legfeljebb polinom méretű tárat használhat a számításai során (hiszen nincs „ideje” több tárat használni). Végül a $\mathbf{PSPCAE} \subseteq \mathbf{EXPTIME}$ abból következik, hogy egy $p(n)$ polinom tárigényű Turing-gép konfigurációs gráfja $2^{\mathcal{O}(p(n))}$ méretű, és ebben a gráfban kell a kezdőkonfigurációból elfogadó konfigurációba vezető utat keresni, ami megtehető a gráf méretében polinom időben. □

Megjegyezzük, hogy bár egyelőre csak annyit tudunk biztosan, hogy $\mathbf{P} \subset \mathbf{EXPTIME}$ és $\mathbf{NL} \subset \mathbf{PSPACE}$, az a sejtés, hogy a fenti tartalmazások mindegyike valódi.

Végezetül a 3.13. ábrán bemutatjuk, hogyan viszonyulnak egymáshoz a könyvben tanult bonyolultsági osztályok, valamint néhány, a **PSPACE** osztályon túlmutató bonyolultsági osztály. Ez utóbbiak közül az eddig még nem tárgyaltak a következők:

- $\mathbf{EXPSPACE} = \bigcup_{k=1}^{\infty} \mathbf{SPACE}(2^{n^k})$,



3.13. ábra. Ismert bonyolultsági osztályok egymáshoz való viszonyai

- $\mathbf{kEXPTIME} = \mathbf{TIME}(2^{2^{\cdot^{\cdot^{2^{n^k}}}}})$, ahol a kitevőben k-szor szerepel a 2-es,
- $\mathbf{ELEMENTARY} = \bigcup_{k=1}^{\infty} \mathbf{kEXPTIME}(n^k)$.

Az **ELEMENTARY** osztályban lévő függvényeket elemi függvényeknek nevezzük. Egy jól ismert nem elemi függvény az Ackermann-függvény.

Fogalmak és jelölések

Ajánlott irodalom

[1] Ésik Zoltán. *A számítástudomány alapjai*. Typotex, 2011.

[2] Fülöp Zoltán. *Automaták és formális nyelvek*. Polygon, 2005.

[3] J. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[4] M. E. Hopcroft, R. Motwani, J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation, 2nd edition*. Addison-Wesley, 2001.

[5] Lovász László. *Algoritmusok bonyolultsága*. Typotex, 2014.

[6] C. H. Papadimitriou. *Számítási bonyolultság*. Novadat Bt., 1999.

[7] Pásztorné Varga Katalin, Várterész Magda. *A matematikai logika alkalmazásszemléletű tárgyalása*. Panem kft., 2003.

[8] M. Sipser. *Introduction to the Theory of Computation, 3rd edition*. Cengage Learning, 2012.