5.4. Megállási probléma

Az előzőekben láttuk, hogy a feladatok túlnyomó többsége algoritmikusan nem megoldható, és általában annak eldöntése, hogy egy feladat algoritmikusan megoldható-e egyáltalán nem nyilvánvaló. A kérdéskört tovább vizsgálva az a természetes gondolat vetődik fel az emberben, hogy ha ennyire általános feladat nem is, de talán az megoldható, hogy egy konkrét T Turing-gépről és egy w szóról megállapítsuk, hogy a T elfogadja-e w -t vagy sem. Másképpen: el tudjuk dönteni, hogy egy adott algoritmus megold-e egy konkrét problémát vagy sem. Ennek vizsgálatához először is pontosan meg kell fogalmaznunk, a megoldandó feladatot.

5.21. Megállási probléma

Döntsük el a P_T programjával adott T Turing-gépről és $w \in \Sigma^*$ szóról, hogy T megáll-e a w bemeneten vagy sem.

A megállási problémának két alapvető megközelítése létezik: egy-egy konkrét esetre, vagy teljesen általánosan szeretnénk erre választ adni.

Az első esetben egyszerűbb a dolgunk, hiszen egyedi módszereket alkalmazhatunk minden alkalommal, speciálisan az adott problémához igazítva.

A második eset sokkal bonyolultabbnak tűnik, hiszen olyan bizonyítási eljárást kell találnunk, amelyik minden Turing-gépre és bemenetre működik. Mivel a bizonyítás lényegében logikai lépések sorozata, így lényegében egy egységes módszert - algoritmust - kell találnunk, ami eldönti az argumentumairól, hogy megfelelnek vagy sem. Ennek alapján a probléma általános megoldhatóságáról a következő állítást fogalmazhatjuk meg és bizonyíthatjuk be.

5.22. Tétel

Nem létezik olyan Turing-gép, amelyik minden P_T programjával adott T Turing-gépről és $w \in \Sigma^*$t szóról eldönti, hogy T megáll-e a w bemeneten vagy sem.

Bizonyítás

A tételt indirekt módon igazoljuk.

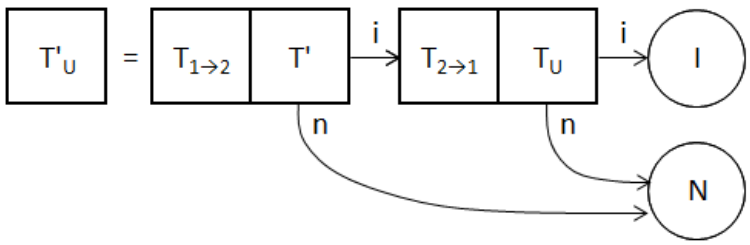
Tegyük fel, hogy létezik T'Turing-gép, amelyik minden $P_T$programjával adott TTuring-gépről és $w \in \Sigma^*$ szóról eldönti, hogy T megáll-e a w bemeneten vagy sem. Ha T megáll a w bemeneten i, egyébként n válasszal tér vissza. Az egyszerűség kedvéért azt is feltételezhetjük, hogy T' bemenetét $P_T \# w$ alakban adjuk meg.

A korábbi jelöléseknek megfelelően legyen

- T_u az univerzális Turing-gép,

- $T_{x \to y}$ egy Turing-gép, ami az x. szalag tartalmát az y. szalagra írja.

Definiáljuk T'_U-t a következőképpen:



T'_U pontosan azokat a w szavakat fogadja el, amelyeket T_U, azaz $L(T'_U) = U$, hiszen azokat a szavakat, amelyeken meg sem áll természetesen nem fogadja el T_U. Mivel T' már csak azokat a w-ket engedi át bemenetként T_U számára, amelyekről kiderítette, hogy meg fog állni rajta, ezért a hátralevő számítás mindenképpen befejeződik, azaz T'_U minden bemenő w szón megáll. Ez azt jelentené, hogy $U \in R$, amiről már beláttuk, hogy nem igaz. Az ellentmondás oka a hibás (indirekt) feltételezésünk. ✓

Az eldönthetőségi problémának megfogalmazható egy egyszerűbb változata is.

5.23. Megállási probléma 2

Döntsük el a P_T programjával adott T Turing-gépről, hogy megáll-e az üres bemeneten vagy sem.

Bár lényegesen egyszerűbbnek tűnik, erre a feladatra is hasonló állítást lehet bizonyítani.

5.24. Tétel

Nem létezik olyan Turing-gép, amelyik minden P_T programjával adott T Turing-gépről eldönti, hogy T megáll-e a λ bemeneten vagy sem.

Bizonyítás

A tételt most is indirekt módon igazoljuk.

Tegyük fel, hogy létezik T' Turing-gép, amelyik minden P_T programjával adott T Turing-gépről eldönti, hogy T megáll-e λ bemeneten vagy sem.

Legyen T_w egy olyan Turing-gép, amelyik a bemenő szalagjára felírja a w $ szót. Nagyon egyszerű ilyen Turing-gépet létrehozni, hiszen csak $n = 2 \cdot l(w)$ darab állapot kell hozzá, például $q_1, \ldots, q_n$, az átmenetfüggvénye pedig úgy definiálható, hogy $\delta(q_i, *) = (q_{i+1}, b_i)$ ahol $b_i = w\left[\frac{(i+1)}{2}\right]$, ha i i páratlan és $b_i = \Rightarrow$, ha i páros.

Legyen T_{prog} egy olyan Turing-gép, ami a bemenő w szóhoz elkészíti T_w programját. A konstrukció során a további szokásos Turing-gépeket fogjuk használni:

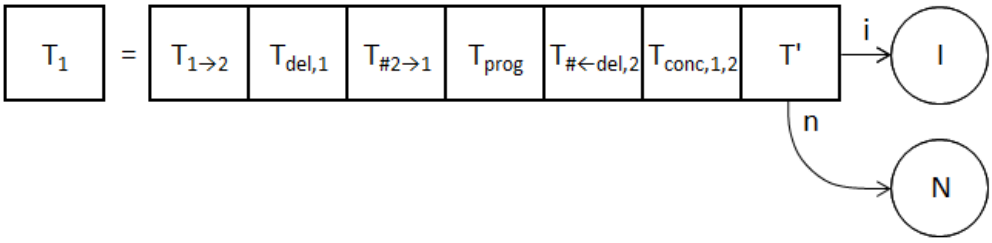
- $T_{x \to y}$ egy Turing-gép, ami az x. szalag tartalmát az y. szalagra írja

- $T_{\#x \to y}$ az x. szalag végétől indulva az első talált $\#$ jelig átmásolja a talált szimbólumokat az y. szalagra

- $T_{\#\leftarrow del,x}$ az x. szalag végétől visszatöröl az első megtalált $\#$ jelig. (A $\#$ szimbólumot is törli.)

- $T_{conc,x,y}$ ha az x.és y. szalagon a T_x és T_y Turing-gép programok találhatók, előállítja az x. szalagra a $T_x \cdot T_y$ Turing-gép programját. Ez szintén nem bonyolult feladat, hiszen lényegében csak össze kell fűzni a két programot, illetve az első végállapotaiból a második kezdőállapotába való átmenetet kell definiálni.

Ezek után definiáljuk T_1-t a következőképpen:



Világos, hogy T_1 minden bemeneten megáll.

Ha bemenetként egy $P_T \# w$ alakú szót adunk meg, akkor T_1 a következőket csinálja:

1. átmásolja $P_T \# w$-t a 2. szalagra,

2. a 2. szalag $\#$ utáni részét, azaz w-t átmásolja az 1. szalagra

3. az első szalagon w-hez elkészíti a T_w Turing-gép programját.

4. a 2. szalagon eltávolítja a $\#$ utáni részt, azaz csak P_T marad

5. az 1. és 2. szalagon levő Turing-gép programokból elkészíti az 1. szalagra az összefűzött programot.

6. megvizsgálja, hogy az 1. szalagon levő Turing-gép megáll-e az üres bemeneten.

Az 5. lépésben létrehozott Turing-gép működése olyan, hogy ha bemenetként az üres szót kapja, akkor először felírja a szalagjára a w szót, majd úgy működik, mint az eredeti T. Vagyis pontosan akkor áll meg az üres bemeneten, ha T megáll a w bemeneten.

Ez azt jelenti, hogy T_1 pontosan azokat a $P_T \# w$ szavakat fogadja el, amelyekre T megáll w-n. Ekkor viszont T_1 megoldaná az általános megállási problémát, amiről az előző tételben bizonyítottuk, hogy lehetetlen. Az ellentmondás oka az indirekt feltételezésünk.

✓