http://serendip.brynmawr.edu/complexity/life.html

http://www.komal.hu/cikkek/kg/sejtauto/kutyak.h.shtml

POLYGON
VI. kötet 1. szám 1996.
9-21.old.

Az élet játék(a)

TOTIK VILMOS

John Conway* „élet" („life") játéka 25 éves. Megalkotása után rövid idő alatt meghódította a számítógép-rajongókat; rengeteg újságcikk témája lett; life hálózat és magazin született, life klubok alakultak. E dolgozatban röviden ismertetjük a játékkal kapcsolatos legfontosabb ismereteket. Reméljük, hogy a dolgozat és a mellékelt program eléri célját, és hazánkban is nagyobb figyelem irányul e valóban rendkívül izgalmas és sok örömet okozó játékra.

Köszönetet mondok Totik Zoltánnak, aki a dolgozat ábráit készítette, továbbá Al Henselnek, aki a mellékelt programot önzetlenül a Polygon rendelkezésére bocsátotta.

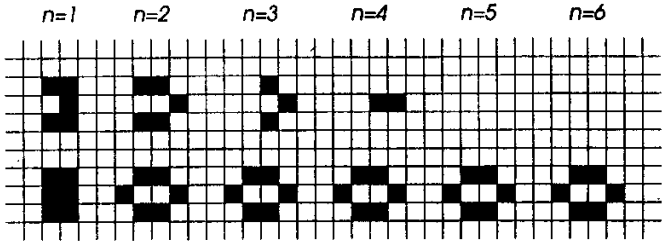
A játék. A life tulajdonképpen nem is játék. Képzeljük el az egységnyi oldalú négyzetekre osztott síkot! Nevezzük a négyzeteket celláknak, és két cellát tekintsünk szomszédosnak, ha van közös pontjuk. Tehát minden cellának nyolc szomszédja van. Egy cellának kétféle állapota lehet: aktív és passzív. Ezenkívül szükségünk van még egy szinkronizáló órára. E nagy óra ütéseire az egyes cellák állapotot változtatnak (mind egyszerre). Egy adott cella állapota időegységenként változik a saját és a szomszédos cellák állapotának megfelelően az alábbi szabály szerint:

(i) Egy aktív cella az $(n+1)$-edik időpillanatban aktív marad, ha az n-edikben pontosan kettő vagy három aktív szomszédja volt, egyébként passzívvá válik.
(ii) Egy passzív cella az $(n+1)$-edik időpillanatban aktívvá válik, ha az n-edikben pontosan három aktív szomszédja volt, egyébként passzív marad.

A „játék" most már a fenti szabályok betartásával a cellák állapotának fejlődése egy kezdeti (aktív-passzív) konfigurációból (mintából) kiindulva. Tehát a mi feladatunk mindössze a kezdeti konfiguráció megadása, innen tovább minden automatikusan megy az (i)—(ii) szabályok szerint. Az időbeni lefutás papíron is követhető (vigyázat, egyetlen hiba tökéletesen más fejlődésmenetet eredményezhet!), de maga a játék könnyen programozható, és igazán izgalmassá a számítógép

* John H. Conway, a XX. század kiemelkedő matematikusa. Kutatási területe többirányú, elsősorban a csoportelmélet, extremális geometria (gömbi pakolások), a csomók elmélete, játékok elmélete és transzfinit aritmetika köré csoportosul, de nagyon sok „szórakoztató" feladat is származik tőle. 1937-ben Liverpoolban született, jelenleg a princetoni egyetem professzora.

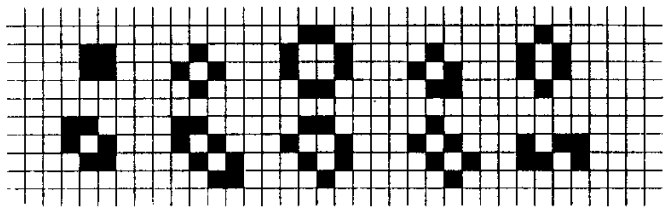
képernyőjén válik. Számítógépet használva tehát a kezdeti konfiguráció megadása után a dolgunk mindössze annyi, hogy kényelmesen hátradőlve élvezzük a képernyőn kibontakozó nyüzsgő életet. Az 1. ábra két egyszerű kezdeti állapot fejlődését mutatja.



1. ábra

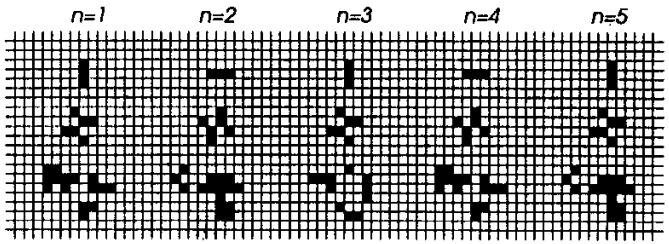
Maga a life (élet) elnevezés onnan ered, hogy az aktív cellákat szokás sejteknek hívni; ekkor a sejtek élhetnek és elpusztulhatnak, és az (i) szabály a túlélés szabálya, míg a (ii) a születésé. Az időbeni lefutás különböző generációk egymás utáni születéseként és átalakulásaként megy végbe. Egy sejt pusztulása kétféle okból következhet be: magányosságból (≤ 1 szomszéd), illetve túlzsúfoltságból (≥ 4 szomszéd).

A 2. ábrán néhány egyszerű konfigurációt mutatunk be, amelyek időben nem változnak. A 3. ábra 2 db kettő periódusú és egy három periódusú mintát mutat be fejlődésükkel együtt. Ezek tehát néhány lépésen belül visszatérnek a kiindulási állapothoz, ezért az ezalatt előállt konfigurációk periodikusan ismétlődnek.

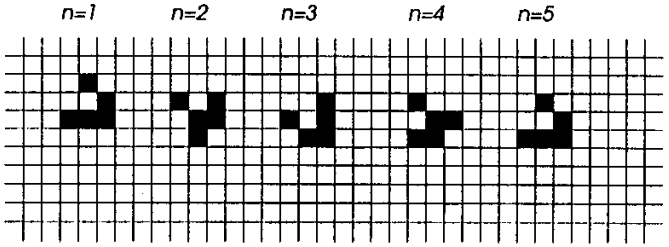


2. ábra

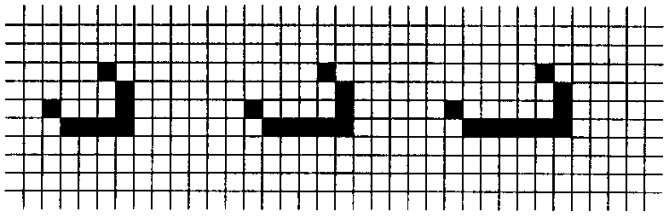
Igazán izgalmassá a life akkor válik, amikor megjelennek „mozgó" alakzatok. Olyan mintákról van szó, amelyek valamilyen k-ra k generáció után megismétlődnek, de az eredeti helyzetükhöz képest kissé eltolva. A legegyszerűbb és egyben legfontosabb ezek közül az ún. **golyó** (angolul glider ~ sikló), amely (megfelelő gyorsasággal futtatva a programot) 45 fokos szögben végigrepül a képernyőn. A 4. ábra a **golyó** időbeni fejlődését mutatja. Az 5. ábra a **golyó**-hoz hasonló ún. urhajókat ábrázol.



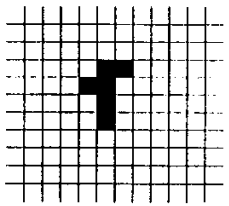
3. ábra



4. ábra

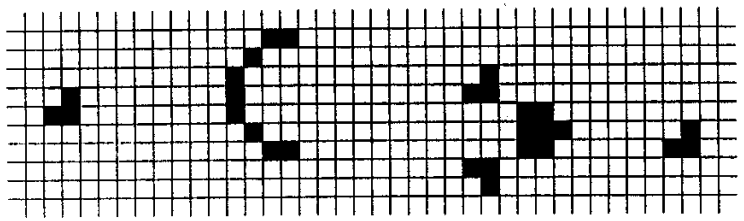


5. ábra



6. ábra

A life megjelenése előtt is több hasonló játék volt ismeretes (ld. a [4], [5] munkákat, továbbá a [2] könyvet, amely sok-sok érdekes játékkal ismerteti meg az olvasót), és Conway ezeket jól ismerte. Mielőtt azonban 1970-ben publikálta a játékot, rendkívül alaposan mérlegelte a születések és pusztulások szabályait. Tudatosan olyan szabályokat választott, amelyekkel a játék nem válik triviálissá sem azért, mert túl gyakran nő a populáció minden határon túl, sem pedig azért, mert túlságosan gyorsan tűnik el. Mint látni fogjuk, ezen alapos mérlegelés olyan játékot eredményezett, amely minden kezdeti várakozást felülmúl. Maga Conway végigelemezte a legegyszerűbb kezdő konfigurációkat, így többek között a hat összefüggő aktív cellából felépülő minták jövőjét is. Az „r" betű formájú kivételével (ld. a 6. ábrát) mindegyik stabilizálódott 10 lépésen belül, az „r" betű formájú azonban különlegesen nehéznek bizonyult, és 460 lépés után a követése túlságosan bonyolulttá vált. Ne feledjük, hogy ez még a számítógépek őskorában volt, amikor képernyő helyett legfeljebb oszcilloszkópra tudták a képet kivarázsolni. A technika fejlődésének köszönhetően ma már persze ez az „r" alakú konfiguráció sem okoz nehézséget, és még egy kisteljesítményű gépen is pár másodperc alatt adódik a „végső" állapot, amely 1103 lépés után éretik el úgy, hogy közben 6 golyó is kirepül.



7. ábra

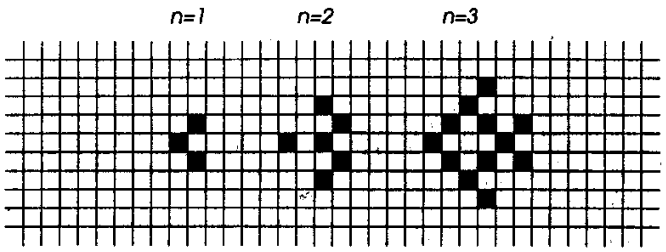
Két golyót más-más irányban elindítva könnyű olyan kezdeti állapotot megadni, amely örökké él, de nem ismétli önmagát. Conway ezen túlmenően azt kérdezte, hogy lehet-e olyan konfigurációt megadni, amelyből kiindulva az aktív cellák száma minden határon túl nő (azt sejtette, hogy nem lehet). Az erre kiírt 50 dolláros díjat a life játék, és általában a komputermanipulálás nagymestere, R. William Gosper kapta, aki a 7. ábrán látható ún. ágyú-t alkotta meg, amely minden 30 generáció után visszatér kiindulási állapotába úgy, hogy közben egy golyó-t röpít ki. Mint látni fogjuk, ez az ágyú a life történetében meghatározó szerepet játszott.

Sejtautomaták, univerzális kiszámíthatóság és a megállási probléma. A jelen fejezet kissé mélyebb területet érint mint a dolgozat többi része, de ez a terület rendkívül izgalmas. Aki csak a life-fal szeretne tovább ismerkedni, az ezt a részt átugorhatja. A fejezettel kapcsolatosan a kiszámíthatóság részletesebb és matematikailag precízebb (bár kissé eltérő) tárgyalására nézve ld. a [3] és [4] munkákat. Az előbbi mindenki által könnyen olvasható, míg az utóbbi tárgyalja a Turing-gépekkel és a rekurzivitással való kapcsolatot is.

A life egy tipikus sejtautomata. Általában sejtautomatának nevezünk identikus sejtek egy rendszerét (a life esetében a négyzetháló négyzeteit), ahol minden sejt véges sok állapot valamelyikében van (esetünkben aktív-passzív). A sejtek között némelyeket szomszédosaknak tekintünk (esetünkben az érintkezőket). A sejtek egy óra ütéseire állapotot változtatnak adott szabály szerint, mely a cella állapotát és a vele szomszédos cellák állapotát veszi figyelembe.

A sejtautomaták elmélete Neumann János munkásságával kezdődött, aki élete utolsó éveit önreprodukáló automatáknak szentelte. Neumann először egy olyan automatát konstruált, amely egy nagy raktárban rohangálva megfelelő alkatrészek kiválogatásával és összeillesztésével képes a saját pontos mását megalkotni. S. Ulam javaslata alapján e fizikai gép hamarosan átkerült a már jólismert négyzethálós síkra, ahol is Neumannak sikerült olyan sejtautomatát konstruálnia, amely minden elképzelhető függvényt képes volt kiszámítani (univerzális számológép, részleteket lásd alább). Ezután következett az ún. univerzális alkotó gép, amely karokat kinyújtva a síkba bármely elképzelhető alakzatot képes megépíteni; így saját magát is. Ezzel Neumannak sikerült egy olyan sejtautomatát megadnia, amely előállítja saját maga hasonmását. Neumann gépében egy négyzet szomszédait a hozzá oldalakkal kapcsolódó négy négyzet adja, a belső állapotok száma 29 volt, és a gép maga kb. 200 000 aktív sejtből állt. A gép magában hordozott egy univerzális számológépet, amely az ideális PC-nek (Personal Computer, személyi számítógép) tekinthető. A konstrukciót sokan egyszerűsítették, ma már vannak értelmes önreprodukáló gépek mindössze négy belső állapottal. Egyébként az is kiderült, hogy önreprodukcióhoz a gépnek nem kell értelmesnek lennie, egészen buta gépek (univerzális számoló rész nélkül) is képesek saját magukat sokszorosítani (pontosan úgy, mint az állatok vagy a kristályok). Szórakoztató feladat pl. azt megmutatni, hogy az alábbi szabály tetszőleges mintát megismétel négy példányban (balra, alul, felül és jobbra) 2^n lépés után (n függ a mintától): a szomszédság a Neumann-szomszédság (szomszédos celláknak van közös oldaluk), és a szabály az, hogy az $(n+1)$-edik időpillanatban egy cella akkor és csakis akkor aktív, ha az n-edikben páratlan sok aktív szomszédja volt (ld. a 8. ábrát).

Sajnos, a sejtautomaták működését (mármint a globális működést) még nem értjük, a probléma a mai matematikai teljesítőképesség határain túl van. A hatvanas években úgy gondolták, hogy a sejtautomaták fogják a számítógépek új generációját eredményezni. Azóta már több számítógép-generáció megszületett, de a sejtautomatákra alapulóra még várni kell. Ez azért is sajnálatos, mert még mindig sokan vélik, hogy az igazán gyors gépek a sejtautomatákban meglévő óriási párhuzamosíthatóság felhasználásával születnek majd meg.



8. ábra

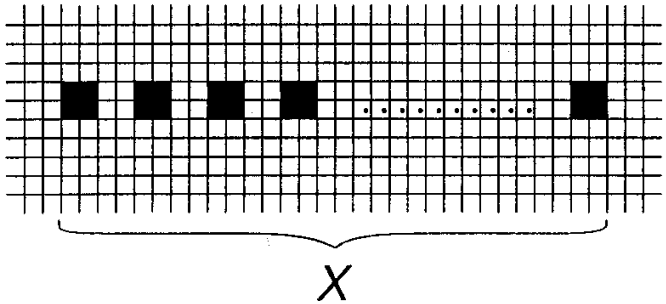
Mielőtt továbbmennénk, szükséges pár szót szólni a kiszámíthatóságról. Számunkra a legkényelmesebb definíció az, ha azt mondjuk, hogy azok a kiszámítható függvények, amelyek kiszámítására egy PC-n program írható. Magát a gépet végtelen memóriával és háttértárral ellátottnak tételezzük fel (ideális PC); tehát nincs megkötés sem az ábrázolható számok nagyságára, sem pedig a program hosszára. A program tetszőleges nyelven íródhat; egyszerűség kedvéért tételezük fel, hogy BASIC-ben íródik. Tehát programon utasítások egy véges listáját értjük, pl. egy tipikus utasítás egy adott változó értékének eggyel való növelése ($X = X + 1$). Mivel csak véges sok utasítás van, adott hosszúságú, pl. 1000 hosszú programok száma véges. A gép a programot soronként hajtja végre, kivéve amikor egy ugró utasításhoz ér, amikor is egy adott programsorra ugrik. A gép bemenetén érthetjük pl. az X változó kezdőértékét, és a kimenetén az X értékét akkor, amikor megáll. Hangsúlyozzuk, hogy a gép nem feltétlenül áll meg egy adott bemenetre; ekkor úgy tekintjük, hogy nincs kiszámított érték. Mint látni fogjuk, az, hogy megáll-e a gép valamikor, az egész elmélet szempontjából döntő jelentőségű kérdés.

Mármost a szuperszámítógépek és pl. a Commodore-ok jelentősen eltérnek egymástól, azonban abból a szempontból ekvivalensek, hogy ugyanazok a függvények számíthatók ki velük (még egyszer megismétlem, hogy végtelen memóriát és programtárat feltételezve). Tehát a szuperszámítógép teljes egészében szimulálható egy Commodore-on, azaz a szuperszámítógép tetszőleges programjára írható olyan Commodore program, amelynél egy adott bemeneten a két gép ugyanazt csinálja (csak persze nem ugyanannyi idő alatt), azaz az egyik pontosan akkor áll meg, ha a másik megáll, és megállás esetén a kimenetek is ugyanazok lesznek. Általában egy gépet (amelyen valamilyen értelemben definiált számolás végezhető) univerzálisnak nevezünk, ha tetszőleges PC program futása szimulálható rajta (a fenti értelemben).

A fenti kiszámíthatósági modell ekvivalensnek bizonyult minden eddigi precízen definiált és elegendően értelmes kiszámíthatósági modellel. Ezért általában elfogadjuk az ún. Church-tézist, amely azt mondja ki, hogy azon függvények, amelyek kiszámítására van eljárás (más néven algoritmus), azok éppen az ideális PC-n kiszámítható függvények. Ez tehát egy munkahipotézis, melyet eddig még minden egyes esetben sikerült igazolni, mihelyt a „kiszámítására van eljárás" fogalmát

pontosan definiálták.

Ezen előkészítés után most már kimondhatjuk a cikk egyik legmeglepőbb állítását, nevezetesen, hogy *a life univerzális gép*; azaz vele bármi kiszámítható, amely más gépekkel kiszámítható. Sőt, bármely PC-n programozható feladatra a life-nak van olyan kezdeti állapota, hogy onnan indulva a life ugyanazt csinálja (megfelelő interpretációval), mint a PC az adott programmal. Persze még azt sem mondtuk meg, hogy mi számítható ki a life-fal! Ahelyett, hogy ezt formálisan definiálnánk, tekintsünk egy konkrét példát! A Goldbach-sejtés a számelmélet egyik nagy megoldatlan problémája, és azt állítja, hogy bármely páros szám előállítható két prímszám összegeként. Eddig még ellenpéldát nem sikerült találni, de az állítást sem sikerült igazolni (megjegyezzük, hogy az ún. páratlan Goldbach-sejtést, amely szerint szerint minden páratlan n természetes szám előáll legfeljebb három prímszám összegeként, Vinogradov igazolta minden elég nagy n páratlan számra). Azonban minden további nélkül írhatunk olyan BASIC programot, amely ellenpéldát keres: X = 2-ből indulunk, majd egy ciklust szervezünk, amely az X értékét kettővel növeli, és (egy újabb cikluson keresztül) megnézi az összes $X = p + q$ felbontást, és ellenőrzi, hogy ebben mindkét tag prím-e (a prímtesztet egy külön szubrutin csinálhatja). Ha nem talál ilyen felbontást, akkor megtalálta a legkisebb ellenpéldát, és a gép megáll. Mármost ugyanez a life-fal is megtehető: van olyan kiindulási minta, hogy onnan elindítva a life-ot, akkor és csakis akkor lesz valamikor az egész sík üres, kivéve sorban egymás mellé helyezett X darab blokkot (ld. a 9. ábrát), ha X a legkisebb ellenpélda a Goldbach-sejtésre. Sőt, a kezdeti minta a fenti Goldbach-programból algoritmikusan megkapható. Magyarán, van olyan számítógépprogram, amely a PC egy tetszőleges P programjához elkészíti azt a life mintát, amiből kiindulva a life ugyanazt az eredményt produkálja, mint a PC a P programmal.



9. ábra

A life univerzalitásának bizonyítása túlmegy e dolgozat keretein (megtalálható az [1] könyvben), azonban a fő gondolatot vázolni tudjuk: az **ágyú golyó**-k sorát bocsátja ki. Két vagy több **golyó** ütközésénél a **golyó**-k megsemmisülhetnek. A **golyó**-k játsszák az 1 szerepét, míg a **golyó** hiánya a sorban a 0-t jelenti.

Így golyó-k és lyukak sorozatával minden 0-1 sorozat modellezhető, és mivel alkalmasan elhelyezett ágyúk-kal és különféle elnyelőkkel és ütköztetőkkel a 0-1 sorozatokon logikai operációk is végezhetők, végül is minden elvégezhető, ami egy PC belsejében történik (ne feledjük, hogy a közönséges asztali számítógép meg a szuperszámítógép is 0-kal és 1-esekkel operál). Ami mindezt lehetővé teszi, az az ágyú létezése. Mármost az ágyú létezését nem lehet igazolni, annak megtalálásához rendkívüli szerencse, esetleg nem mindennapi intuíció kellett. Ez szép példája annak, hogy egy, a számítógépen véletlenül megtalált konfiguráció ad lehetőséget arra, hogy egy tisztán elméleti állítást igazoljunk.

Ha lenne eljárásunk annak eldöntésére, hogy egy kiindulási konfigurációból indulva a life megáll-e, akkor el tudnánk dönteni minden aritmetikai állításról, hogy igaz-e. Pl. a Goldbach-problémánál a program úgy is modellezhető a life-fal, hogy ha talált ellenpéldát, akkor megáll (stabilizálódik) a gép az említett módon, egyébként nem. Mármost ekkor a Goldbach-sejtés megoldásához mindössze a fenti eljárással kellene tesztelnünk a Goldbach-konfigurációt, azaz az eljárásunkkal megnézni, hogy ebből a konfigurációból kiindulva a life megáll-e. Szerencsére (vagy sajnos?) nincs, és nem is lehet eljárás bármely aritmetikai állítás eldöntésére (erről alább bővebben szólunk), ezért nem lehet eljárás a life jövőjének megjósolására sem. Azaz a life viselkedése valóban kiszámíthatatlan.

D. Hilbert az 1900-ban tartott matematikai kongresszuson huszonhat probléma köré csoportosította azokat a kérdéseket, amelyek szerinte az elkövetkezendő (XX.) században a matematika fejlődését meg fogják határozni. Ezek lettek a híres Hilbert-problémák. A 23. közülük azt kérte, hogy adjunk meg olyan mechanikus eljárást, amellyel minden aritmetikai (számelméleti) állításról eldönthető, hogy igaz-e vagy sem. A válasz csak 31 évvel később született meg, és alapjaiban megrázta a matematikát, és az ismeretelméletben is új irányt nyitott. Ugyanis 1931-ben Kurt Gödel, a princetoni kutatóintézet fiatal professzora igazolta, hogy bármilyen axiómákból is indulunk ki, mindig lesz olyan aritmetikai állítás, amely igaz, de nem bizonyítható (és mivel igaz, nem is cáfolható). Ez valami olyasmi, mintha azt mondanánk, hogy bár a Goldbach-sejtés igaz a természetes számokon (azaz valóban minden páros szám két prímszám összege), de ez a tény a felvett axiómákból nem igazolható. Persze más axiómákból igazolható lesz, de akkor meg egy másik igaz állítás nem lesz bizonyítható. Gödel tétele egyben a Hilbert-program végét is jelentette, mivel egy mechanikus eljárás létezése, mely minden állításról eldöntené, hogy az igaz-e, egyben eszközt adna az igaz állítások bizonyítására. Gödel után a figyelem a bizonyítható állításokra fordult azt remélve, hogy legalább a bizonyítható állítások felismerhetők mechanikusan, azaz lesz eljárás annak megítélésére, hogy egy állítás bizonyítható-e. Ez gyengébb, mint amit az eredeti Hilbert-féle program kért, hiszen bizonyítható állításból kevesebb van, mint igaz állításból (ami bizonyítható, az automatikusan igaz is), és már maga a bizonyítás fogalma is egyfajta mechanikus eljárást jelent. Jegyezzük meg mindjárt, hogy a bizonyítható állítások listáját minden további nélkül elkészíthetjük: tudunk olyan programot írni, amely sorban elvégzi a bizonyításokat (először az 1 hosszúakat, majd a 2 hosszúakat stb.), és a kapott tételeket felírja egy listára. A fenti eljárás

ennél többet kér, nevezetesen azt, hogy egy adott A állításról mondjuk meg, vajon a listán szerepel-e vagy sem. Megtehetnénk, hogy sorban generálnánk a lista elemeit, és ha megtaláljuk A-t, akkor persze mondhatjuk, hogy igen, A bizonyítható. A probléma azzal van, hogy mikor mondjuk azt, hogy *A nem bizonyítható*? Ha még eddig nem találtuk meg a listán, várjunk-e még két napot, és utána mondjuk, hogy nincs rajta? Lehet, hogy a rákövetkező órában rá került volna! Tehát itt egy tipikus megállási problémával állunk szemben, nem tudjuk, hogy mikor kell, vagy lehet megállnunk a lista átnézésével.

Mint kiderült, a bizonyítható állítások felismerésére sincs mechanikus eljárás; ezt A. Church and A. Turing egymástól függetlenül igazolták. Turing ezirányú munkái során alkotta meg a róla elnevezett univerzális számológépet (Turing-gép), mely a modern matematika-számítástudományban az algoritmusok bonyolultságának vizsgálatánál alapvető szerepet játszik.

Mi itt szorítkozunk a life megjósolhatatlanságára, és igazoljuk, hogy nincs algoritmus annak eldöntésére, vajon egy konfigurációból kiindulva a life megáll-e (a life jövője megjósolhatatlan). Tételezzük fel, hogy lenne erre egy T eljárás. Akkor arra is lenne, hogy egy adott PC egy adott programmal megáll-e (emlékezzünk, a programból a megfelelő life konfiguráció mechanikusan megkapható, és akkor csak erre kellene T-t alkalmazni). Legyen $\sigma(n)$ a lehető legnagyobb szám, amelyet egy n hosszú programmal a PC-n megkaphatunk. Ekkor T segítségével $\sigma(n)$ kiszámítható, hiszen nem kell más tennünk, mint sorban végignézünk az n hosszú programokat (ezekből véges sok van), ezekből kidobjuk azokat, amelyeken a PC nem áll meg (itt van szerepe T-nek), majd a PC-t lefuttatjuk a megmaradt programokkal, és a kapott számok közül kiválasztjuk a lehető legnagyobbat. Mindez minden további nélkül programozható (persze a programban az n input paraméterként szerepel). Mármost a life viselkedésének megjósolhatatlansága abból következik, hogy, mint azt az alábbiakban megmutatjuk, $\sigma(n)$ valójában nem kiszámítható. Először is jegyezzük meg, hogy $\sigma(n)$ szigorúan monoton nő. Valóban, ha egy n hosszú program éppen $\sigma(n)$-et számítja ki, azaz $X = \sigma(n)$-et ad kimenetként, akkor az $X = X + 1$ sort utánaírva olyan $n + 1$ hosszú programot kapunk, amely $\sigma(n) + 1$-et ad kimenetként, azaz $\sigma(n + 1) > \sigma(n)$. Tételezzük mármost fel, hogy a $\sigma(n)$ függvény kiszámítható, pl. az m hosszú P program kiszámítja; azaz P az $X = n$ bemenet esetén az $X = \sigma(n)$ értékkel áll meg. Egy adott n-re tekintsük az alábbi Q_n programot:

$$
\begin{aligned}
&X = 0\\
&X = X + 1\\
&X = X + 1\\
&\vdots\\
&X = X + 1\\
&X = X * X,
\end{aligned}
$$

amely n db $X = X+1$ utasítást tartalmaz. Nyilvánvalóan Q_n éppen n^2-et számítja ki, ezért a Q_n után írva a P programot, egy olyan $n + 2 + m$ hosszú programot kapunk, amely a $\sigma(n^2)$-et számítja ki. De nagy n-re $n^2 > n + 2 + m$, és így $\sigma(n^2)$,

amely a σ szigorú monotonitása miatt nagyobb, mint $\sigma(n+2+m)$, nem számítható ki $n+2+m$ hosszú programmal. A kapott ellentmondás igazolja állításunkat a life kiszámíthatatlanságáról.

További élet kérdések. Megmutatható, hogy life minták nemcsak univerzális számológépként viselkedhetnek, hanem univerzális konstruktőrökként is. Speciálisan, vannak önreprodukáló alakzatok a life-on belül. Most képzeljünk el egy óriási véletlen masszát a life térben (véletlen mintát, ilyet a mellékelt számítógépes programban a „!" paranccsal kaphatunk). Valószínűsíthető, hogy egy idő után megjelennek olyan alakzatok, amelyek bizonyos ideig fenn tudnak maradni, pl golyó-k tömegeit küldve ki a környezetükbe abból a célból, hogy tájékozódjanak, ill. közeledő objektumokat megsemmisítsenek. Ezek közül bizonyosak jobban alkalmazkodnak mint mások, és ha még véletlenül önreprodukálásra is képesek, megindulhat a törzsfejlődés. Az evolúcióhoz szükséges módosító tényezők bőven léteznek ebben az ősmasszában, tulajdonképpen a mutációk létrejötte a tipikus life fejlődés. A legtöbb mutáció persze haszontalan lesz és elpusztul, de néha-néha hasznos mutánsok is létrejöhetnek, amelyeknek a környezetükhöz való alkalmazkodása tökéletesebb. Előbb-utóbb létre fognak jönni olyan alakzatok, amelyek saját hasznukra át is tudják alakítani a környezetüket, sőt a korábbi tapasztalatokat fel is tudják használni (már az univerzális life minta is rendelkezik memóriával). Azaz kialakulhatnak értelmes lények. Innen tovább már csak a képzelet korlátai szabnak határt a fejlődésnek. Mindez nem science fiction, annál sokkal több, mivel pozitív a valószínűsége minden egyes eseménynek, amit leírtunk (precíz definiciók után), feltételezve persze, hogy valóban nagy véletlen mintából indulunk ki, és a rendelkezésre álló idő is korlátlan.

Mindenesetre figyelemreméltó, hogy egy ilyen egyszerű, kétállapotú sejtautomata milyen bonyolult fejlődést eredményezhet. Az eddig vizsgált alakzatok csak valóban a legelemibb szintjét jelölik a lehetséges fejlődésnek.

A fentieknek a valódi élettel való analógiájának nehéz ellenállni. Vannak, akik megfordítják az egészet, és nem zárják ki azt sem, hogy a mi tér-időnk is kvantumos, és az élet nem más, mint egy óriási számítógép által vezényelt szimuláció, amely szubatomi szinten ugyanolyan egyszerű szabályok szerint működik, mint a sejtautomaták.

A life játékkal kapcsolatban sok megoldatlan probléma is van. A life kiszámíthatatlanságát mutatja pl. az, hogy nem tudjuk, hogy mely n-ekre fog az egy sorban egymás mellett levő n db aktív cellából álló minta végül is eltűnni. Az első negyven esetből ez az n hosszú sor végül is eltűnik, ha $n = 1, 2, 14, 15, 18, 19, 23, 24$. Ebből nem kínálkozik semmilyen általános formula a kérdéses n-ekre. Az sem ismert, hogy milyen gyorsan nőhet a populáció száma; mi a lehető legsűrűbb minta, amely még stabil marad; mi a legkisebb elemszámú minta, amelynek növekedése négyzetes; vagy mi a legkisebb minta, amely csak kezdeti konfigurációként állhat elő (ún. „éden kertje").

A szabályokon is lehet változtatni, de eddig még az eredeti Conway-féle szabályok bizonyultak legérdekesebbnek. Hasonlóan, nem kell, hogy az univerzum

négyzetekből álljon, állhat pl. háromszögekből vagy hatszögekből, de ezek valamilyen értelemben mindig szimulálhatók négyzethálós automatákkal. Végezetül természetesen elkészült a life térbeli változata. Itt ábrázolási problémák jönnek elő, hiszen az élő cellák eltakarják a mögöttük levőket. A kétdimenziós life szimulálható a háromdimenzióssal, és megtalálták a golyó valódi térbeli megfelelőjét is. Mindenesetre a térbeli life „még egy dimenzióval" komplikáltabb, mint az eredeti.

A programról. Mint említettük, a life-ot viszonylag könnyű gépre programozni. Azonban a populáció növekedésével a számolási igény hirtelen felugorhat, amely jelentősen lelassíthatja a futást; ezért *jól programozni* nem is olyan könnyű.

A mellékelt lemezen található egy kiváló life szimulátor több mint száz érdekes life mintával együtt. A programot Al Hensen amerikai mérnök fejlesztette ki. Hensen maga is több izgalmas life konfigurációt talált, és még ma is intenzíven foglalkozik a témával. Köszönet illeti, hogy a programot ingyen a Polygon folyóirat rendelkezésére bocsátotta. E programban az univerzum persze nem lehet véges, de olyan nagy (524 288 × 524 288), hogy ez a gyakorlatban nem jelent problémát. Továbbá az univerzum szélét elérő sejtek a másik oldalon felbukkannak, tehát a program tulajdonképpen egy tóruszra íródott.

A lemezen az lf.exe file található. Először hozzunk létre egy könyvtárat, ebbe másoljuk bele a file-okat, majd adjuk ki az lf utasítást. Ez kibontja a tényleges programot (a feltett kérdésre az y billentyű megnyomásával válaszoljunk (yes)). Ezt csak egyszer kell megcsinálnunk, ezután a program a létrehozott könyvtárból bármikor a „life" paranccsal indul DOS promptból (kezdeti mintanév is megadható a „life" parancs után).

A nyilak (vagy egér) mozgatják a képernyőn a kurzort; egy cella aktívvá ill. passzívvá tétele a szóköz leütésével történik. Az „enter" parancs indítja a szimulációt, és bármely betű leütése megállítja. „q" kilép.

További funkciók:

„o" (one) egy lépést tesz.
„b" (break) a féket állítja be: 0 teljes sebesség, 1-9 pedig növekvő sebességeket jelentenek.
„v" teljes sebesség.
„c" (clear) törli a mintát.
„d" (drop) egy leírást ad az aktuális mintáról (ha van).
NumLock ha be van kapcsolva, akkor a numerikus billentyűkkel lehet az univerzumban mozogni (pl. golyó-kat követni).
„5" visszavisz a középpontba.
„+", „-" a cellák nagyságát állítja (ha a video mód megengedi).
„l" (load) betölt egy mintát (az aktuális könyvtárból, kiterjesztés: .lif). A szóközt megütve a filenév kiegészül az alfabetikusan következő megfelelő névre.
„s" (save) egy mintát elment a kurzor helyével mint középponttal.
„!" véletlen mintát rajzol.

„#" a véletlen minta paramétereit állítja.
„k" megadja az aktív cellák számát.
„r" lehetőséget ad a szabályok változtatására. A formátum: 34/12, ahol a / előtti számok a túléléshez szükséges szomszédos cellák számát, a / utániak pedig a születéshez szükséges szomszédos cellák számát adják meg. Tehát 23/3 az eredeti játék.
„Alt-B" box mód bekapcsolása, amely a cut-paste parancsokhoz hasonlóan lehetőséget ad részek másolására. Az egérrel rajzolhatjuk meg a box-ot, a „szóköz" kivágja, majd újra a „szóköz" bemásolja.
„Alt-D" box mód kikapcsolása.

További információk a lemezen levő life106.doc file-ban találhatók.

Végezetül említsünk meg külön is néhány figyelemre méltó life konfigurációt a lemezen levő készletből (ezek mind .lif kiterjesztésűek).

breeder (és különféle variációi) négyzetes növekedéssel rendelkezik.

cordpull (és variációi mint pl. a **hacksaw**) olyan minta, amelynél a populáció nem korlátos, de nem is tart végtelenbe. Nevezetesen, a $t = 8 \cdot 6^n - 216$ időpillanatban a populáció elemszáma $t/36 + 558$, míg a $t = 32 \cdot 6^n - 429$ időpillanatban a nagyság csak 469.

Az **efence** konfiguráció végeit számítógépes keresőprogrammal találták meg.

Az **irrat4** mintánál egy űrhajó vagy eltalálja az ún. csónakot, vagy egy **golyó** őt eliminálja. Az előbbi lehetőséget jelöljük 1-gyel, az utóbbit 0-val. Így kapjuk az 1010110110... sorozatot. Ha ez elé tesszük az 101-et, akkor megkapjuk az ún. Fibonacci-sorozatot, amelynek definíciója a következő: az 1-ből indulva a képzési szabály az, hogy a már meglévő sorozatban minden 0-t 1-gyel, minden 1-gyet pedig 10-zel helyettesítünk. Így adódik tehát: 1, 10, 101, 10110, 10110101 stb. Könnyen látható, hogy n lépés után az egyesek száma éppen F_n, a 0-k száma pedig éppen F_{n-1}, ahol F_n a Fibonacci-sorozat n-edik tagja. Mivel a Fibonacci-sorozatban az 1-esek sűrűsége $(\sqrt{5}-1)/2$, adódik, hogy az **irrat4** mintánál a t időpontban a populáció nagysága $(8-31\sqrt{5}/10)t$ körül van, azaz a növekedés a t egy irracionális számszorosa.

Megoldatlan az **irrat5** konfiguráció elemszámának kérdése: azt sejtik, hogy páros t-re t lépés után a nagyság $(78\sqrt{2}-73)t/40$, míg páratlan t-re $(82\sqrt{2}-77)t/40$ körül van.

max a leggyorsabban növő ismert konfiguráció.

A **primes** konfiguráció valami hihetetlent csinál: 120 időegységenként balra küld ún. űrhajókat. Ezek az utolsó akadályon akkor és csakis akkor tudnak kb. a $120n+10$ időpillanatban áthaladni, ha n prím.

switchen a legkisebb korlátlanul növő konfiguráció.

a **tiretrax** két potenciális gyújtózsinórt hagy maga után. Rendkívül látványos, ha ezeket meggyújtjuk (egy vagy pár aktív cellát hozzáteszünk).

Látványnak sem lebecsülendők a **crystal, gunstar, makehslr, oscsp2,3, oscspn, pinball, pstrain, pusher, sawtooth3, spriral, still, thue** minták, de a többit is érdemes kipróbálni, ill. módosítani.

Jó szórakozást!

IRODALOM

[1] E. R. Berlekamp, J. H. Conway, R. K. Guy, Winning Ways, Academic Press, 1982.
[2] Csákány Béla, Matematikai Játékok, Polygon Kiskönyvtára (megjelenés alatt)
[3] Gaizer Tamás, Mindenható számítógép?, *Polygon* **1996** VI:1.
[4] Lovász László, Algoritmusok bolyolultsága, ELTE jegyzet, 1994.
[5] R. G. Shrandt and S. M. Ulam, On recursively defined geometrical objects and patterns of growth, 1967.
[6] S. M. Ulam, Essays on cellular automata, University of Illinois Press, 1970.

Tolik Vilmos, JATE Bolyai Intézet, Szeged, Aradi vértanúk tere 1.