Előszó a "Bevezetés a programozáshoz" jegyzethez és a relációs programozási modellhez

> Ha valami újat akarsz elmondani, és nem tudod kétféleképpen elmondani, akkor még magad sem érted.
> A jó programspecifikáció és program ugyanazt mondja el kétféleképpen.

Amikor programozásról akarunk beszélni, programozással akarunk foglalkozni, kezdetben intuitív fogalmakra hivatkozhatunk, feltételezhetően többségünknek van ösztönös programozási hajlama.

Ne is a programozáson kezdjük, hanem azt képzeljük el, hogyan fut le egy képzeletbeli gépen egy program. Ne foglalkozzunk először azzal, miért így fut, pusztán kívülről figyeljük, mi történik. Amit nézhetünk, az a gép állapotának, - leegyszerűsítve - memóriájának változási sorozata. Ami a változást okozza, vezérli, az a program.

Az általánosság korlátozása nélkül - digitális technikát feltételezve - a memória lehetséges állapotai legyenek a nem negatív egész számok, amelyeknek halmazát jelölje N. Egy S program lefutásai a memória kezdeti állapotából kiinduló véges, vagy végtelen sorozatok, tehát $S \subseteq N \times N^{**}$ részhalmazként jellemezhető egy program. Valós lehetőségünk ebből mindig csak egy véges rész megfigyelése. Minden, amit még tudhatunk a lehetséges lefutásokról ezen felül, az már csak a program felépítésének ismeretén alapulhat.

Amikor programozunk, tulajdonképpen választunk a lehetséges programok közül. Nem céltalanul választunk, hiszen meg akarunk oldani a programmal egy feladatot, amit az adott lehetőségre korlátozódva egyszerűen úgy kell tekintenünk, hogy milyen memóriaállapotból milyen állapotokba akarunk eljutni. Ezzel a szemléltetéssel egy F feladat $N \times N$ részhalmaza.

Minthogy a végtelen állapottéren végtelen lefutásokra képes programok viselkedése véges megfigyelések alapján teljesen nem ismerhető meg, a program kiválasztását sem tudjuk általában közvetlenül $S \subseteq N \times N^{**}$ alakban megadni. Szükség van arra, hogy a lefutások valamilyen szabályosságát adjuk meg a választás jellemzésére, s tulajdonképpen ez a programozás lényege.

Hasonló mondható el a feladatról is, véges jellemezhetőség nélkül a feladat sem adható meg végességre kárhoztatott lények/gépek számára. A teljes nagy programozási "játék" arra megy ki, hogy végesen jellemezhető feladatokhoz olyan végesen jellemezhető programokat adjunk meg, amelyek végrehajtása megoldja a feladatot. Érezhető az eddigiekből, hogy programot sem választhatunk tetszőlegesen, és a programmal megoldható feladat sem lehet tetszőleges. (A matematika sem tud mást kezdeni a végtelennel, mint véges jellemzéseket elemezni, ekvivalenciájukat keresni.)

A programok és a programmal megoldható feladatok világa egymást kölcsönhatásban határozza meg, hasonló módon építjük a megoldható feladatok világát, mint a megoldó programokét, a feladatot gyakran algoritmus, egymásra építés adja meg, a feladat $N \times N$ alakban önmagában is tekinthető programnak. Alapvetően építkezéssel ismerünk meg egyre nagyobb részt az uralmunk alá vonható feladatokról és programokról. (A matematika világa is építkezéssel válik egyre szélesebben ismertté és használhatóvá.)

Nézzünk két egyszerűnek tűnő példát.

Legyen az állapottér a TÍZ $= \{0,1,2,3,4,5,6,7,8,9\}$ halmaz. Az $S_{\text{valós}}$ program legyen minden bemenetre az összes lehetséges véges vagy végtelen decimális sorozat. A program neve is azt jelzi, hogy a lehetséges lefutások halmaza megfeleltethető a [0,1) intervallum valós számainak. A véges, és észrevehető lefutások viszont a véges tizedes-törteket adják meg. A program által megoldott feladat mindössze a TÍZ $\times$ TÍZ feladat, és minden megoldás végtelen sokféleképpen áll elő.

A másik példa konstruktív úton adja meg ugyanezt a programot. Induljunk ki abból, hogy a program tartalmazza a TÍZ $\times$ TÍZ feladatot, és rendelkezik a következő tulajdonsággal: bármely véges lefutás folytatható az utolsó értékhez, mint kiinduláshoz tartozó bármely lefutás tetszőleges véges kezdőszeletével, és minden ilyen véges vagy végtelen folytatás lehetséges lefutás. Könnyen látható, hogy ezzel a konstruktív megadással minden véges vagy végtelen lefutás előáll, tehát az $S_{\text{valós}}$ programot kapjuk.

Nézzük meg, hogy egy elvileg (matematikailag) elképzelhető programról mit tudunk megállapítani, mire tudnánk használni a program $N \times N^{**}$ formában adott lefutásait. Azt tételezzük fel, hogy tetszőleges sok kezdőértékből tetszőleges sokszor indíthatjuk a végrehajtást, s bármelyik megkezdett végrehajtást megállíthatunk, majd folytathatunk. Két természetes korlátot nem tudunk túllépni: a gép memóriaváltozásához egy minimális időegység, kvantum szükséges, és véges idő használható a megfigyelésre. Megfigyeléseinket egy kétdimenziós táblázatba foglalhatjuk: egy sor egy adott kiindulási értékhez tartozó memóriaállapotokat tartalmazza.

Szervezzük úgy a megfigyelést, hogy egy megfigyelési szakaszban minden megkezdett lefutást folytatva M hosszban figyeljünk meg, ha még nem ért véget. Minden új megfigyelési szakasznál kezdjünk egy új indítást minden eddigi indulási helyről, és még egy újabb helyről, és M értékét növeljük. (Még annyi "okosságot" tételezzünk fel a gépről, hogy ugyanazt a lefutást nem választja még egyszer. Tehát, ha egy pontból csak véges sok lefutás van, akkor ahhoz a ponthoz csak véges sok sor fog tartozni.)

A következő alakú táblázatot kapjuk:

0	$A_{1,1},$	$(A_{1,2},$	$A_{1,3},$	.....	$,A_{1,M})$
0	$A_{2,1},$	$(A_{2,2},$	$A_{2,3},$	.....	$,A_{2,M})$
1	$A_{3,1},$	$(A_{3,2},$	$A_{3,3},$	.....	$,A_{3,M})$
0	$A_{4,4},$	$(A_{4,2},$	$A_{4,3},$	.....	$,A_{4,M})$
1	$A_{5,1},$	$(A_{5,2},$	$A_{5,3},$	.....	$,A_{5,M})$
2	$A_{6,1},$	$(A_{6,2},$	$A_{6,3},$	.....	$,A_{6,M})$
0	$A_{7,1},$	$(A_{7,2},$	$A_{7,3},$	.....	$,A_{7,M})$
1	$A_{8,1},$	$(A_{8,2},$	$A_{8,3},$	.....	$,A_{8,M})$
2	$A_{9,1},$	$(A_{9,2},$	$A_{9,3},$	.....	$,A_{9,M})$
3	$A_{10,1},$	$(A_{10,2},$	$A_{10,3},$	.....	$,A_{10,M})$
.					
.					
.					
L	$A_{K,1},$	$(A_{K,2},$	$A_{K,3},$	.....	$,A_{K,M})$

A táblázat első oszlopa adja meg, melyik állapotból indul a lefutás. $A_{i,j}$ jelöli, hogy az i-edik megkezdett lefutás a j-edik lépésben milyen memóriaállapotot eredményezett, amennyiben nem állt meg a lefutás a j-edik lépés előtt. Minden megkezdett lefutást M lépésig néztünk meg, és $L+1$ megfigyelési szakaszt hajtottunk végre.

A táblázatban $K = (L+1)L/2$, amennyiben egyik sor sem üres. A zárójeles szakaszok véget érhetnek az M-edik lépés előtt, akkor a sor folytatása üres.

Mit tudunk ebből a véges táblázatból megállapítani: mindössze azt, hogy melyik sor ért véget, és azt, hogy melyik kiindulási értékhez nincs már több lehetséges programlefutás.

"Szabványosítsuk" az $S \subseteq N \times N^{**}$ programok megadását a fenti kétdimenziós táblázattal. Így a következő tulajdonságokat állapíthatjuk meg véges idő alatt:

1. Az S program az i bemenetre kiszámolja j-t, ha egy i-hez tartozó sor véges, és utolsó eleme j.
2. Az S program csak véges különböző lefutást rendel az i bemenethez. Speciálisan, az S program az i bemenetre determinisztikus, ha pontosan egy sor tartozik i-hez. (Ez a tulajdonság csak a gép "okosságára" tett feltevés esetén állapítható meg!)

Ez azt jeleni, hogy amennyiben egy programra valamelyik fenti tulajdonság egy adott i-re teljesül, akkor azt véges idő alatt megtaláljuk, észlelhetővé válik egy elég nagy véges ablakban. Ellenkező esetben, ha egy tulajdonság nem teljesül, véges idő alatt ez nem mutatható ki pusztán a táblázat alapján.

Jóllehet ezek a tulajdonságok igen korlátozott megismerést biztosítanak a programokról, mégis lehetőséget adnak a programok osztályokba sorolására, "jó" kiszámítási tulajdonságok jellemzésére. (A "jó" tulajdonságok az 1.-és 2- típusú megfigyelésekből építhetők fel.) A jó tulajdonságú programokhoz bizonyos mértékig meg tudjuk határozni, hogy milyen feladatokat, milyen értelemben oldanak meg. Általában elemi, intuitíven könnyen megadható programokból indulunk ki, amelyeknek a szabványosított táblázaton való kiszámítása megmutatható. A megismert tulajdonságú programokból (vagy bizonyos feladatokhoz kívülről, orákulomokkal megadott programokból) további programokat építhetünk. Amennyiben az építkezés megfelelő szabályok betartásával történik, a keletkező programok jó tulajdonságait garantálni tudjuk.

A programosztályokra, konstrukciókra a programozás fejlődése során számos absztrakció, fogalom, módszer alakult ki már eddig is. A programlefutások táblázatos, explicit megadása mellett számos más módja van programosztályok megadásának. A kiszámítás eltérő módon való megadása más absztrakciókat tesz lehetővé, más módon találhatunk megoldható feladatokat, programkonstrukciókat.

A programozás legvégső célja az, hogy létező gépen működtethető programok készüljenek. Ehhez a realizálható gépek működését is jellemezhetővé kell tenni. A formális számítási modellek által használt gépek mindegyikéről elmondhatjuk, hogy van egy természetes realizálásuk: maga az emberi gondolkodás. Végtelen idő, tér (papír) és anyag (tinta) rendelkezésre állása esetén minden kiszámítást el tudnánk végezni. Ez a realizálás mutatja egyben kiszámítási képességünk elvi lehetőségeinek végességi korlátjait: a táblázatos szemlélettel érzékeltetve igen kis ablakokban nézhetjük a végtelen táblázatot. Kell tehát az emberi gondolkodás helyére valódi gép, ami a lehetőségeket felnagyítja. A korszerű számítógépek adják ezt az egyre bővülő lehetőséget.

Nézzük meg a legtipikusabb Neumann-elvű számítógép felépítését és működését a relációs programmodellel szemléltetve.

A gép építésének első komponensei: néhány véges halmazon értelmezett determinisztikus programot végrehajtó processzor, legyenek a végrehajtott feladatok $F_1, F_2, F_3, , F_k$.

A feladatok végrehajtására a gép utasítható, ehhez meg kell tudjuk mondani számára, melyik feladatot hajtsa végre, honnan vegye a bemenő értéket, hova helyezze a kiszámítás eredményét. A gép számára ezért olyan memóriát építünk, amely címezhető rekeszekből

áll, és a cím megadásával kiolvasható a rekesz tatalma, illetve felülírható. A memória és a memóriát olvasó, író berendezés a gép felépítésének (architektúrájának) második alkotója.

Hátra van még a Neumann-elv, a tárolt programú gép elvének felhasználása. A végrehajtandó utasítás három számmal jellemezhető: melyik feladatot kell végrehajtani, melyik memóriarekeszt kell olvasni, és melyik memóriarekeszbe kerüljön a kiszámítás eredménye. Az utasítások ezzel tárolhatóvá válnak a memória rekeszeiben.

Az utasításokat végrehajtó processzorok mellé már csak a központi végrehajtó processzort kell megkonstruálni: a központi processzor egy lépés során megnézi a soron következő memóriarekesz tartalmát, és a tartalom függvényében két lépés valamelyikét hajtja végre:

- A rekeszben tárolt utasítást végrehajtatja a megfelelő processzorral, és a cím szerint következő memóriarekeszre lép. A gép megáll, ha a memória végére ért, és ekkor a memória tartalma lesz a végeredmény.
- A rekeszben egy címet talál, és az a cím lesz a soron következő utasítást tartalmazó rekesz.

Az így felépített gép programozása mindössze annyit jelent, hogy a gép memóriáját feltöltjük.

Látható, hogyha a gép C rekeszből áll, és rekeszenként K különböző érték tárolására képes, akkor "csak" K^C méretű állapottéren képes programokat megvalósítani. Ez is csak egy véges ablak. Természetesen a K, C paraméterek növelésével és végrehajtási idő csökkentésével egyre nagyobb méretű ablakokhoz jutunk. A növekedés üteme - Moore törvény -még mindig exponenciális.

Egy Neumann-elvű adott memóriaméretű gép által realizálható programok relációs megadása elvégezhető, az $F_1, F_2, F_3, , F_k$ egyértelműen meghatározza az összes lehetséges lefutást. A programok determinisztikusak, de nem minden lefutás véges. A nem véges lefutások periodikus ismétlődésekből állnak. Minthogy a memória minden esetben tartalmaz utasításokat is, szét kell még választani a memóriának a tényleges program bemenetét jelentő címtartományát és a végeredményt tartalmazó címtartományát a programutasításokat tartalmazó címtartománytól. Ez azt jelenti, hogy a valódi feladat és program beágyazódik egy bővebb állapottéren lefutó programba.

Rögzítsük a címtartomány adott részének tartalmát, mint utasítások sorozatát. Nevezzük ezt Neumann-programnak. Vizsgálhatjuk ezek után, hogy a memória többi részét tetszőlegesen feltöltve milyen eredménye lesz a lefutásnak. Ez lesz az adott Neumann-programhoz tartozó program a relációs szemléltetésben.

Adott Neumann gép minden Neumann-programjához megadható a fenti értelmű relációs szemléletű program, tehát az is megtalálható, hogy adott véges feladathoz létezik-e a gépen Neumann-program. A Neumann-programok rendkívüli nagy száma miatt azonban ez az út nem járható, programozással kell az adott feladat (véges szeletének) kiszámítását végző Neumann-programot megadni. A programozás jelenlegi feladata - Neumann gépeket feltételezve - mindössze ennyi. Ahhoz, hogy egyáltalán találjunk programot, hogy hatékony programot találjunk, igen gazdag szemléltetési háttérre, és igen precíz, szabatos gondolkodási módra van szükség. Minél többféle módon le tudjuk írni, mi a feladat, hogyan adjuk meg a programot, annál több esélyünk van jó megoldás megtalálására. A programozásra is érvényes bizonyára az, amit a matematikára mondanak: "Nincs királyi út."

Van viszont rengeteg végtelen út, végtelen programlefutás, amit el kell kerülni!

Budapest, 2001. december 2.

Benczúr András