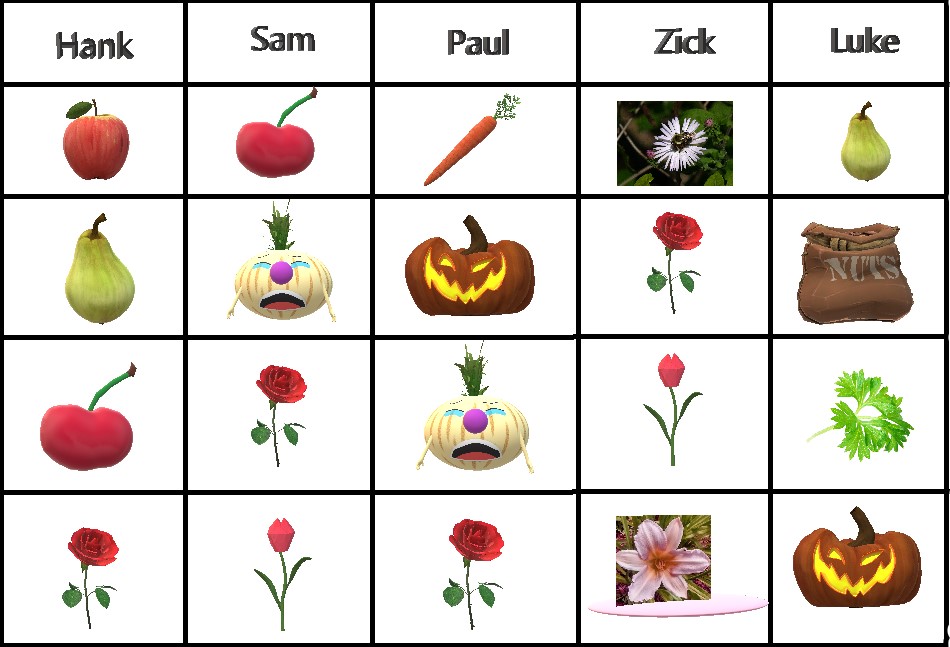
## Gardens

A munkánk során részletesen bizonyos előre „legyártott” feladatokat vizsgáltunk. Az első ilyen neve „Gardens”, azaz „Kertek”, amit az Einstein-féle logikai feladványokhoz sorolnak kategóriája szerint.

Adott öt barát, akiknek a kertjei egymás mellett helyezkednek el. Ezeken a területeken tizenkettő féle növényből termesztenek négyet-négyet fejenként. Azt is tudjuk, hogy ezekből a terményekből négy gyümölcs, négy zöldség és négy virág van. Azt, hogy ki melyik kertben dolgozik és azt, hogy mit tartalmaznak ezek, azt kikötések sora után tudjuk csak meg, aminek a végeredményét az alábbi kép szemlélteti.

Előszőr is az általunk használt parancsokat mutatnám be a kikötések fajtái alapján a megoldásunk első verzióján. Itt bináris mátrix segítségével dolgoztunk, azaz, ha megtalálható valami a kertben, akkor a helyére egyest, ha nem akkor nullát raktunk a táblázatba. A tulajokat egy külön tömbben tároljuk, melyeket összekötöttünk a kertekkel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Hank** | **Sam** | **Paul** | **Zick** | **Luke** |
| **alma** | **1** | 0 | 0 | 0 | 0 |
| **körte** | **1** | 0 | 0 | 0 | **1** |
| **mogyoró** | 0 | 0 | 0 | 0 | **1** |
| **cseresznye** | **1** | **1** | 0 | 0 | 0 |
| **sárgarépa** | 0 | 0 | **1** | 0 | 0 |
| **petrezselyem** | 0 | 0 | 0 | 0 | **1** |
| **tök** | 0 | 0 | **1** | 0 | **1** |
| **hagyma** | 0 | **1** | **1** | 0 | 0 |
| **őszirózsa** | 0 | 0 | 0 | **1** | 0 |
| **rózsa** | **1** | **1** | **1** | **1** | 0 |
| **tulipán** | 0 | **1** | 0 | **1** | 0 |
| **liliom** | 0 | 0 | 0 | **1** | 0 |

A legfontosabb hogy minden utasítást „constraint”-el kell kezdeni. A szó maga is nagyon jól szemlélteti, hogy amit mögé írunk, azt úgymond „kikényszerítjük”, hogy tartsa be a munka során.

A képeken is jól látszik, hogy Zick például csak virágokat termel, míg Hank-nél vagy Sam-nél mind három fajta termény megtalálható, tehát nagyon különbözőek a kertek. Direkt ezekre az esetekre használtuk programunk során az „alldifferent”-et, ami kifejezi, hogy mint a kertek, mint a tulajok csak és kizárólag egyszer szerepelnek és eltérnek egymástól.

% egy embernek pontosan 1 kertje van(azaz minden tulaj különböző)

constraint alldifferent(tulaj);

A leírásban sok olyan kikötés szerepel, ami kijelenti, hogy mi hányszor szerepel. Ennek három speciális esetét különböztettük meg. Az első mikor konkrétan megadják hogy melyik kertben vagy melyik tulaj termeszti/nem termeszti az adott dolgot, de eme módon akár a tulajt is megadhatják. Ilyenkor egyszerűen csak megadjuk az adott cella értékét.

%Paul kertje a középső, liliom nélkül.

constraint termeszt[12,3]=0;

constraint tulaj[3]=4;

A második mikor csak annyit tudunk, hogy az adott növényt hányszor termeljük. Ilyen esetekben azt az előnyt használtuk ki, hogy mivel a táblázatunkban számok vannak, így lehet sima összeadás műveletet használni, így ilyenkor a sorösszegeket adtuk meg neki.

% Az alma csak egy kertben szerepel.

constraint forall(k in Kertek)(sum(k in Kertek)(termeszt[1,k])=1);

A harmadik pedig nagyon hasonlít a másodikhoz, de a szövegben valamilyen formában még jobban hangsúlyozza az állítást. Ilyen esetekben az „exactly”-t használtuk, aminek konkrét felépítési kritériuma van.

% Csak egy kertben található egy fajta növényből mind a négy darab.

constraint exactly(1, [bool2int(sum(z in Zoldsegek)(termeszt[z, k])=4

\/ sum(gy in Gyumolcsok)(termeszt[gy, k])=4

\/ sum(v in Viragok)(termeszt[v, k])=4) | k in Kertek], 1);

Végül, de nem utolsó sorban van még egy kikötési formánk, de ez nem volt elég hatékony megoldás a többi megoldásunknál már nem ezt használtuk.

% 21. Paul pontosan három fajta zöldséget termeszt.

constraint forall(t in Tulaj where tulaj[t]=4)

(sum(z in Zoldsegek)(termeszt[z,t])=3);

Ezt a „where”-es megoldást implikációra cseréltük le a projektünk további részeiben.

constraint forall(t in Tulaj)(tulaj[t]=4 -> sum(z in Zoldsegek)(termeszt[z,t])=3);

A programunk második verziója csupán az ilyen kikötések felépítésében tér el, de mégis sokkal hatékonyabb lett.

A harmadik verzióban már a meglévő mátrix mellé felvettünk még egyet a tulaj-kert kapcsolatokhoz. Az új mátrixot szintén egyesekkel és nullákkal töltöttük fel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Luke** | **Sam** | **Hank** | **Paul** | **Zick** |
| **Kert1** | 0 | 0 | **1** | 0 | 0 |
| **Kert2** | 0 | **1** | 0 | 0 | 0 |
| **Kert3** | 0 | 0 | 0 | **1** | 0 |
| **Kert4** | 0 | 0 | 0 | 0 | **1** |
| **Kert5** | **1** | 0 | 0 | 0 | 0 |

A kikötések felépítésében itt nincs változás, tehát nem sok különbség van a két változat között.

A következő ötletünk az volt, hogy a kétdimenziós tömb helyett halmazokkal reprezentáljuk a megoldást. A tulajokat csak egy egyszerű tömbben tároltuk itt el. Két fontos új kifejezés is megjelent a kikötésekben ennél a munkánál. Az első a „card”, ami megadja a részhalmaz elemszámát.

% Mindenki négy különböző növényt termeszt.

constraint forall(k in Kertek)(card(termeszt[k])=4);

Ehhez a „card”-hoz használtunk még helyenként „intersect” parancsot is, ami kifejezi, hogy az éppen vizsgát halmaz részünk milyen másik halmazzal legyen keresztezve.

% Paul pontosan három fajta zöldséget termeszt.

constraint forall(t in Tulaj)(tulaj[t]=4 -> card (Zoldsegek intersect termeszt[t])=3);

A másik újdonság az „array\_union”, ami igazából egyesíti a kívánt dolgokat egymással.

% Minden egyes variáció minimum egy kertben megtalálható .

constraint card(array\_union(termeszt))=termenyek;

A Kertek utolsó verziójában ismét visszatértünk a mátrixos megoldáshoz, ám itt már nem bináris értékekkel töltöttük fel, hanem konkrétan a terményekkel. A leghasznosabb változtatás itt az volt, hogy a tömböket „enum” kulcsszóval töltöttük fel, így akár maga az objektum nevét, akár csak a sorszámát írtuk le a kikötésben, mind kettő variációt felismerte a rendszer. Egy másik fontos változtatás volt, hogy itt mivel nem tudtunk sorösszegeket számolni, így a „count”-ot kellett használnuk.

% Hank nem termel őszirózsát.

constraint forall(k in Kertek)(tulaj[k]=Hank -> count([termeszt[k, n] | n in Noveny], oszirozsa)=0);

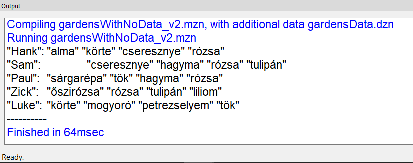
Egy plusz kikötést kellett még írnunk ehhez a változathoz az egy megoldás megtartása érdekében. Le kellett fixálnunk a termények sorrendjét, hogy ne adjon ki több lehetőséget is a megoldó, így a sorszámuk alapján növekvő sorrendben rakosgattuk be a növényeket a helyükre. Ennek a módszernek pontos szemléltetése volt a feladat elején bemutatott kép.

constraint forall(k in Kertek, n in 1..noveny-1)(termeszt[k,n] < termeszt[k,n+1]);

A kimeneteket egységesen minden verziónál formáztuk a jobb olvashatóság érdekében.

output [show(tulajNev[fix(tulaj[1])]) ++ ":\t"]++[show(termenyNev[t]) ++ " "

| t in Termenyek where fix(termeszt[t,1])=1]++["\n"];



"Output"

Ötből három változatban az adatokat külső fájlban tároltuk, így például, ha csak a növények neveit szeretnénk módosítani, nem kell az egész programunkat átírni hanem csak az adatfájlt. Ez a módszer is egy jó megoldás, ha munkát szeretnénk spórolni később.