Zebra puzzle-ök modellezése

A bevezetőben bemutatott Einstein-féle úgynevezett Zebra Puzzle-típusú fejtörők modellezése és megoldása tette ki kutatási tevékenységünk nagy részét. Célunk az volt, hogy minél több féle adatszerkezettel és esetleg egy adatszerkezethez is minél több féle módon leírható korlátozásokat készítsünk. A könnyebb kezelhetőség illetve az átláthatóság érdekében a korlátozásokat felépítésük alapján kategorizáltuk, így az ugyan más tartalmú, de hasonló logikai felépítésű kifejezések megírásakor jelentős mennyiségű időt takaríthattunk meg.

A következőkben részletesen bemutatjuk ezen példák megoldására készített modelleket. A két példa, amelyekkel részletesen foglalkoztunk a „Movies Night” névre hallgató „könnyű” nehézségi szintű és a „Fundraising Dinner” nevű „nagyon nehéz” nehézségű voltak. Továbbá mindegyikhez két különböző adatszerkezeti megvalósítást, illetve ezeken belül további kétféle korlátozás-leírási módot használtunk.

Lássuk is az első 4x5-ös nagyságú példát: adott 4 barát, akik moziba mennek egyik este. A moziban egymás mellé ülnek le. Mindegyikük azon felül, hogy hányadik széken ül, 5 tulajdonsággal jellemezhető:

* Milyen színű inget visel: **fekete, kék, zöld, piros**
* Keresztnév: **Daniel, Joshua, Nicholas, Ryan**
* Milyen típusú film a kedvence: **akció, vígjáték, horror, thriller**
* Nassolnivaló: **chips, süti, cracker, popcorn**
* Életkor: **11, 12, 13** és **14** éves

Ahhoz, hogy megtudjuk, melyik széken ki ül és milyen tulajdonságokkal rendelkezik, a rendelkezésünkre áll 13 állítás, amelyeket modellezésünk során korlátozásokként(constraint) kezeltünk.

Az első verziójú modell esetében a 4 fiú adatainak tárolására tulajdonságtípusonként egy-egy bináris értékeket tartalmazó 2 dimenziós mátrixot deklaráltunk:

%Bemenő paraméter és segédhalmaz az indexeléshez: hány ember van

int: db = 4;

set of int: PEOPLE = 1..db;

enum NAMES = {Daniel, Joshua, Nicholas, Ryan};

enum MOVIES = {action, comedy, horror, thriller};

[...]

array[NAMES, PEOPLE] of var {0, 1}: name;

array[MOVIES, PEOPLE] of var {0, 1}: movie;

[...]

A MiniZinc nyelv sajátosságait kihasználva a sorok indexelésére egy-egy felsorol(enum) típusú tömböt használtunk. Ennek a megoldás kiíratásánál illetve a korlátozások megfogalmazásánál lesz különös szerepe. Ebből következik, hogy a mátrix sorai az egyes tulajdonságok konkrét értékeit az oszlopai pedig sorrendben a fiúk moziban elfoglalt helyét jelölik. Amelynek indexelésére pedig szintén a leíró nyelv sajátossága miatt egy {1..4} értékkészletű segédhalmazt használtunk, ahogy az a fenti ábrán is látható. Vegyünk egy példát: amennyiben a movie mátrix 3. sorának és 1. oszlopának metszetében az 1-es érték szerepel(movie[horror, 1]=1) azt jelenti, hogy az első helyen ülő fiúnak a horrorfilm a kedvence.

Belátható, hogy mivel egy tulajdonság kategóriából minden fiúhoz(egész pontosan ülőhelyhez, mivel még nem tudjuk, ki hol ül) pontosan egy értéket rendelünk hozzá, ezért az összes bináris mátrix sor- és oszlopösszegének egyenlőnek kell lennie 1-gyel. Ezt hivatottak biztosítani az alábbi korlátok:

% Oszlopösszeg:

constraint forall(p in PEOPLE)(sum(n in NAMES)(name[n, p])=1);

% Sorösszeg:

constraint forall(n in NAMES)(sum(p in PEOPLE)(name[n, p])=1);

[...]

A forall biztosítja, hogy minden sorra(ill. oszlopra) teljesüljön az, hogy a sum függvény visszatérési értéke(amely az oszlopot ill. sort összegzi) 1.

Ezek után következhetett a 13 korlátozás megfogalmazása. Ezen példa esetében 6 különböző típusú korlátozást azonosítottunk, amelyek esetében a kifejezések szerkezete változatlan csak az adatok(egész pontosan az indexelés) változik.

1. típus: konkrétan megmondják, hogy mely pozícióban milyen tulajdonságú ember ül:

%A 14 éves fiú a harmadik helyen ül.

constraint age[fourteen,3]=1;

1. típus: hasonló az elsőhöz, de itt a széksor valamelyik széléről nyilatkozunk:

%Joshua valamelyik szélen ül.

constraint name[Joshua,1]+name[Joshua,db]=1;

Itt kihasználjuk hogy bináris mátrixszal dolgozunk, ezért két elem összege 1 kell hogy legyen, miszerint vagy az egyik szélen vagy a másikon ülhet Joshua.

1. típus: két tulajdonságot kapcsol össze egymással:

%Joshua a horrorfilmet szereti.

constraint forall(szek in PEOPLE)(name[Joshua,szek] = movie[horror, szek]);

1. típus: valamely tulajdonságú ember mellett közvetlenül balra(illetve jobbra) ülő emberről állít valamit:

%A fekete inget viselő fiú közvetlenül a thrillert szerető ember bal oldalán ül.

constraint movie[thriller,1]=0;

constraint forall(szek in 2..db)(movie[thriller,szek]=1 -> shirt[black, szek-1]=1);

Itt viszont ki kell kötnünk, hogy az első helyen(tehát a bal szélen) nem ülhet a thriller kedvelő, mivel az ő bal oldalán már nem ülhet senki. A korlátozás megfogalmazásánál ez esetben az implikációs operátort(->) használtuk, amely kimondja, hogy ha megtaláltuk a thrillerkedvelőt, akkor fekete inges ül mellette(igaz állításból csak igaz következhet), egyéb esetben nem történik semmi.(hamis állításból bármi következik)

Ennek kiváltására használtunk egy továbbfejlesztett korlátozás-leírási mód esetén tisztán relációs operátorokat:

constraint forall(szek in 2..db)(movie[thriller,szek] = shirt[black, szek-1]);

1. típus: adott személy valamely tulajdonságú ember mellett balra(ill. jobbra) tetszőleges pozícióban foglal helyet

%A fekete inges fiú a legfiatalabbtól valamelyik balra eső helyen ül.

constraint forall(x in PEOPLE)(age[eleven,x]=1 -> sum(szek in 1..x-1)(shirt[black, szek])=1);

Szintén relációs operátor az implikáció kiváltására:

constraint forall(x in PEOPLE)(age[eleven, x] <= sum(szek in 1..x-1)(shirt[black, szek]));

1. típus: egy ember valamely két másik között tetszőleges pozícióban foglal helyet:

%A piros inget viselő fiú valahol a 13 éves és az akció kedvelő között ül, ebben a sorrendben.

constraint forall(x in PEOPLE, y in PEOPLE)(age[thirteen,x]=1 /\ movie[action,y]=1 -> sum(szek in x+1..y-1) (shirt[red,szek])=1);

Ebben az esetben viszont a logikai operátor relációsra cseréléséhez két részre kellett bontanunk az állítást:

constraint forall(x in PEOPLE)(movie[action, x] <= sum(szek in 1..x-1)(shirt[red, szek]));

constraint forall(x in PEOPLE)(age[thirteen, x] <= sum(szek in x+1..db)(shirt[red, szek]));

A kimenet képzése a nyelvnek megfelelő szintaktikával valósult meg a következőképpen:

output [format(width, show(n)) ++ "\t" | t in PEOPLE, n in NAMES where fix(name[n,t])=1]++["\n"];

[...]

A format függvény pusztán esztétikai formázást, a show karakterlánccá konvertálást, a fix függvény pedig a változók értékének „stabilizálását” végzi. A where záradékban pedig azt vizsgáljuk, hogy a bináris mátrixban hol található 1-es érték, mert csak azokat az értékeket íratjuk ki.

Az eddigiekben részletezett bináris mátrix mellett egy másik, ettől merőben eltérő adatszerkezettel rendelkező implementációt is készítettünk. Itt tulajdonságkategóriánként egy-egy felsorol(enum) típusú egydimenziós tömböt készítettünk, amelyet a már említett segédhalmazzal indexeltünk:

array[PEOPLE] of var NAMES: name;

array[PEOPLE] of var MOVIES: movie;

[...]

Itt a tárolt értékek már a konkrét tulajdonságok neveit vehetik fel, sorrendjük pedig meghatározza hogy melyik széken ülő emberre mi jellemző. Egy tömbön belül az értékek egyediségét az alldifferent globális constraint biztosítja:

constraint alldifferent(name);

constraint alldifferent(movie);

[...]

A korábban már részletezett korlátozások típusokat a következőképpen modelleztük jelen adatstruktúra esetében(ugyan azokat a kifejezéseket mutatjuk be, mint az előbb):

1. age[3]=fourteen;
2. name[1]=Joshua xor name[db]=Joshua;
3. forall(szek in PEOPLE)((name[szek]=Joshua) = (movie[szek]= horror));
4. forall(szek in 2..db)(movie[szek]=thriller -> shirt[szek-1]= black);
5. forall(x in PEOPLE)(age[x]=eleven -> count([shirt[szek] | szek in 1..x-1],black)=1);
6. forall(x in PEOPLE)(movie[x]=action -> count([shirt[szek] | szek in 1..x-1], red)=1) /\  
   forall(x in PEOPLE)(age[x]=thirteen -> count([shirt[szek] | szek in x+1..db], red)=1);

A tömbös megvalósításnak a másik korlátozás-leírási módszerét pedig a második, 5x6-os méretű feladatunkon, a „Fundraising Dinner” címűn keresztül fogjuk bemutatni. Néhány szó magáról a feladatról: a leírás szerint 5 jómódú hölgy egy jótékonysági vacsorán vesz részt. Egy egyenes asztalnál, egymás mellett foglalnak helyet és mindegyikük 6 féle tulajdonsággal jellemezhető:

* keresztnév
* viselt ruha színe
* nyakláncukon található drágakő típusa
* életkoruk
* fogyasztott koktéljuk
* adakozott pénzmennyiség dollárban

Ezen módszernél az indexekkel való bejárásnál fogalmaztunk meg egy apróbb szabályt a következőképpen:

forall(a,b in PEOPLE where a<b)(not(cocktail[a]=margarita /\ dress[b]=blue));

Így tudtuk biztosítani a V. típusú korlátozás egy másfajta leírását. Ezen felül ennél a feladatnál azonosítottunk egy újfajta korlátozást, amely egy objektum valamely oldaláról nyilatkozik, viszont nem rögzíti, hogy melyikről. Ezeknél egy új elemet, az úgy nevezett visszafelé következtetést is használtuk. Az alábbi példát megvizsgálva észrevehetjük, hogy abból, hogy Lidia a Cosmopolitant ivó mellett ül, viszont következik az is hogy a Cosmopolitan fogyasztó mellett – üljön ő bárhol is- valamely oldalon kell helyet foglalnia Lidia-nak. Ehhez az xor azaz a kizáró vagy operátort hívtuk segítségül.

%Lidia a Cosmopolitant ívó hölgy mellett foglal helyet.

forall(p in 2..db-1)(name[p]=Lidia -> (cocktail[p-1]=cosmopolitan xor cocktail[p+1]=cosmopolitan)) /\

forall(p in 2..db-1)(cocktail[p]=cosmopolitan -> (name[p-1]=Lidia xor name[p+1]=Lidia)) /\

(name[1]=Lidia -> cocktail[2]=cosmopolitan) /\

(name[db]=Lidia -> cocktail[db-1]=cosmopolitan) /\

(cocktail[1]=cosmopolitan -> name[2]=Lidia) /\

(cocktail[db]=cosmopolitan -> name[db-1]=Lidia);

Azokat az eseteket, amelyeknél az említett személyek valamely szélén ülnek a széksornak, itt is külön kell kezelni, mivel egyrészt ekkor egyértelmű egymáshoz viszonyított helyzetük. Másrészt pedig így elkerülhetjük a tömbök bejárása során a kiindexelést.

A IV. típusú korlátozás leírását pedig az ekvivalencia operátor(<->) használatával valósítottuk meg, a széksor széleit a fenti okok miatt ismét külön kezelve.

%A legidősebb hölgy közvetlenül a Cosmopolitant ivó jobbján ül.

forall(p in 1..db-1)(cocktail[p]=cosmopolitan <-> age[p+1]=sixty) /\

(cocktail[db]!=cosmopolitan) /\

(age[1]!=sixty);

Az eredmény kiíratásánál itt kihasználtuk azt, hogy a tömb elemeit nem kell vizsgálnunk, hanem sorrendben, vizsgálat nélkül kiírathatóak.

output [format(width,show(dress[p])) ++ "\t" | p in PEOPLE]++["\n"];