Szemantikus és deklaratív technológiák oktatási segédlet

Zombori Zsolt, *zombori@cs.bme.hu*
Szeredi Péter, *szeredi@cs.bme.hu*

Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi és Információelméleti Tanszék

2012. május 23.

Bevezetés

Az alábbi segédlet a BME mérnök informatikus mesterképzésének Számításelmélet szakirányán oktatott *Szemantikus és deklaratív technológiák laboratórium* című tárgyához készült. A laboratórium két korábbi szakirányos tantárgy, a *Bevezetés a szemantikus technológiákba* és a *Nagyhatkonyságú deklaratív programozás* folytatásának tekinthető, ennek megfelelően két jól elkülönülő részből áll. Az első részben azt vizsgáljunk, hogy hogyan készül el egy leíró logikai ontológia. A segédlet illusztrálja az onológiakészítés folyamatát egy egyszerű társasjáték modellezésén keresztül. A tárgy második felében logikai programozással, azon belül korlát logikai programozással foglalkozunk. Alaposan megvizsgáljunk a SICStus Prolog CLPFD könyvtárában található kombinatorikai korlátokat, melyek segítségével egész értékű változókra lehet komplex korlátokat megfogalmazni. A kombinatorikai korlátok legfőbb jelentősége, hogy bonyolult korlátkendszereket tudunk egyben, egyetlen globális korlátként megfogalmazni, így sokkal hatékonyabb szűkítési mechanizmust lehet elérni. A segédlet tartalmazza néhány korlát használati útmutatóját, valamint olyan programozási feladatokat, melyeket az adott korlátok segítségével kell megoldani. A feladatok célja, hogy a félév végére a hallgatók gyakorlatot szerezzenek a kombinatorikai korlátok használatában különféle korlát kielégítési problémák megoldásához.

1. fejezet

Modellezés leíró logikai ontológiák segítségével

Ebben a fejezetben példát mutatunk arra, hogy hogyan készül el egy leíró logikai ontológia. A modellezendő világ az *Aranyásók* nevű egyszerű társasjáték, melynek leírása a függelékben olvasható.

Ontológiákat elsősorban arra használunk, hogy egy adott tárgyterület szavainak, fogalmainak jelentését formálisan meghatározzuk. A jelentés formális meghatározását pedig úgy érjuk el, hogy a fogalmak egymással való kapcsolatrendszerét írjuk le. A fogalmak jelentése nem változik, így az ilyen leírások eredendően statikusak és ritkán tartalmaznak eseményeket, változásokat. Ennek ellenére az itt bemutatott modellezendő világ egy társasjáték, melyben keverednek a statikus elemek (a játékban használt fogalmak jelentése valamint a játékbeli entitások tulajdonságai) és dinamikus elemek (mi mi után következik, hogyan alakul az állás az idő múlásával, stb.). A célterület választását az motiválta, hogy íly módon jobban érzékeltetni tudjuk, hogy mit lehet és mit nem lehet megfogalmazni leíró logikai ontológiák segítségével.

1.1. Mit szeretnénk modellezni?

A modellezendő világ minden aspektusát valószínűleg nem tudjuk modellezni, és valószínűleg nincs is rá szükség. Így rögtön az elején döntéseket kell hozzunk: mi kerüljön bele a modellbe és mi maradjon ki? A döntéseket befolyásolja, hogy mire szeretnénk felhasználni a modellt és milyen elvárásaink vannak vele szemben. Ezen felül fontos figyelembe venni, hogy a modellezésre használt eszköz (esetünkben leíró logika) milyen jellegű modell építésére alkalmas.

Leíró logikák segítségével többnyire statikus leírásokat lehet készíteni, melyek meghatározzák bizonyos fogalmak jelentését az egymáshoz való viszonyuk alapján. Egy társasjátékra jellemző dinamizmust (mi mi után következik, az idő múlásával hogyan alakul a helyzet stb.) nem, vagy csak nagyon körülményesen lehet megjeleníteni egy leíró logikai tudásbázisban. Ezért úgy határozunk, hogy egyetlen játékmeneten belül kialakuló pillanatnyi állás jellemezésére készítünk modellt. Célül tűzzük ki, hogy a modell alapján meg tudjunk válaszolni olyan kérdéseket, mint például „Nyertek-e már az aranyásók?” vagy „Érvényesen vannak-e lerakva a járatkártyák?” stb. A dinamizmus modellezése nélkül nem fogunk tudni olyan kérdésekre válaszolni, hogy „Ki áll nyerésre?” vagy „Ha letiltom a szomszédomat, nyerhetnek-e még a szabotőrök?”

Az a tervezői döntés, hogy egy pillanatnyi állást jellemző modellt hozzunk létre, maga után vonja, hogy a játék bizonyos aspektusait figyelmen kívül lehet hagyni. Egy játékmenetben nincsenek játékosok, csak aranyásók és szabotőrök. Így azon jellemzőket, melyek a játékosokra vonatkoznak, de az éppen általuk megtestesített szereplőkre nem, ki fogjuk hagyni a modellből. Ilyen például a játékosok által bírtokolt aranyak száma. Hasonló módon, míg útkártyák meg fognak jelenni a modellben, aranyásókártyák nem lesznek, hiszen azokat a modellen kívüli világban használtuk arra, hogy eldöntsük, kik népesítsék be a modellbeli világot.

1.2. A modell szintaktikai építőköveinek meghatározása

Döntsük el, hogy milyen szintaktikai elemek jelenjenek meg a modellben. Esetünkben ez a leíró logikai fogalmak és szerepek meghatározását jelenti.

Olvassuk végig a játékleírást figyelmesen, mondatról mondatra! Egyesével döntsük el, hogy az adott mondat tartalmaz-e olyan információt, amit bele szeretnénk építeni a modellbe. Ha nem, húzzuk ki a mondatot. Sok oka lehet annak, hogy egy mondatot kihúzunk. A természetes nyelvű leírások gyakran redundánsak és ugyanazon információ többszörös előfordulásából nekünk elég egyet megtartani. Ezen kívül kihúzhatjuk azokat a részeket, melyek a játék modellen kívül eső részére vonatkoznak. Az is előfordulhat, hogy egy mondatnak semmi érdekes információtartalma sincsen, vagy túl általános és később úgyis pontosításra kerül.

A megmaradt mondatokból ezután gyűjtsük össze, hogy milyen fogalmakra és szerepekre lesz szükségünk az információ modellezéséhez. Sok esetben ez egybeesik a szövegben előforduló fontosabb főnevekkel (fogalmak) és igékkel (szerepek). Ez egy előzetes lista, melyet az ontológia axiómáinak felvételekor még módosíthatunk.

> A játékosok lehetnek szorgos aranyászók ~~akik csákányokkal lemennek a hegyek tárnáiba arany után kutatni~~, vagy pedig szabotőrökként játszanak, ~~megakadályozva és hátráltatva az aranyászókat. A két csapat tagjainak támogatniuk kellene egymást akkor is, ha csak sejtik melyik játékos melyik csapatba tartozik.~~ Ha az aranyászóknak sikerül eljutniuk az aranyhoz, akkor arany a jutalmuk, ~~a szabotőrök pedig üres kézzel mehetnek haza.~~ Ha ez nem sikerül nekik, akkor a szabotőrök győznek, ~~övék a jutalom arany, az aranyászók pedig nem kapnak semmit. Csak akkor derül ki, hogy melyik játékos melyik csapathoz tartozik, amikor az arany elosztására kerül sor. A játékot az a játékos nyeri, akinek 3 fordulóban a legtöbb aranyat sikerül összegyűjtenie.~~

Az egyes játékosok ismereteit nem fogjuk modellezni, így lényegtelen, hogy kiről mikor derül ki, hogy szabotőr vagy aranyász. Mivel a modellünk csak egy játékmeneten belüli világról szól, így az is érdektelen, hogy 3 forduló alatt ki mennyi aranyat gyűjt. Ebből a bekezdésből fontos, hogy vannak játékosok, aranyászók, szabotőrök valamint megtudunk valamit arról, hogy kik lesznek a nyertesek.

Fogalmak: Játékos, Aranyász, Szabotőr, Nyertes

> ~~A játékosok szétválogatják az~~ utikártyákat, az akciókártyákat, ~~az aranykártyákat, és külön válogatják azokat a kártyákat is, amiken a törpék vannak.~~
>
> ~~A játékosok számának megfelelő darabszámú aranyász és szabotőr kártya kerül a játékba. A maradék törpés kártyát a játékosok félreteszik.~~
>
> - 3 játékos esetén 1 szabotőr és 3 aranyász.
> - 4 játékos esetén 1 szabotőr és 4 aranyász.
> - 5 játékos esetén 2 szabotőr és 4 aranyász.
> - 6 játékos esetén 2 szabotőr és 5 aranyász.
> - 7 játékos esetén 3 szabotőr és 5 aranyász.
> - 8 játékos esetén 3 szabotőr és 6 aranyász.
> - 9 játékos esetén 3 szabotőr és 7 aranyász.
> - 10 játékos esetén az összes kártya (4 szabotőr és 7 aranyász) kerül a játékba.
>
> ~~A játékosok választanak egy osztót, aki a „törpés" kártyákat megkeveri és lefordítva elosztja a játékosok között. A játékosok megnézik a kapott kártyát, majd lefordítva maguk elé teszik. A játékosok nem mondhatják el egymásnak, hogy milyen szerep jutott nekik ebben a fordulóban. Egy kártya az osztónál marad. Ezt~~

az osztó játékos lefordítva félreteszi. A játékosok csak a forduló végén fordíthatják meg kártyáikat.

A 44 kártya között van egy startkártya (ennek mindkét oldalán egy létra látható) és három célkártya. Az egyik célkártyán van az arany, a másik két célkártyán csak egy-egy kő látható. Az osztó játékos megkeveri a célkártyákat, majd lefordítva az asztalra teszi azokat a startkártyával együtt, a képen látható módon. A játék folyamán a startkártyától kiindulva egy labirintusjárat alakul ki. A labirintusjárat kialakításakor az útkártyák kilóghatnak az ábrán látható határokból.

Egy labirintust fogunk építeni útkártyákból. A labirintusnak van néhány kitüntetett része, mint a start-, illetve célkártyák.

Fogalmak: Útkártya, Startkártya, Célkártya, Aranykártya, Kőkártya, Labirintuskártya

Az osztó játékos összekeveri a maradék 40 útkártyát és minden akciókártyát, majd lefordítva elosztja a játékosok között.

- 3-5 játékos esetén minden játékos 6 kártyát kap a kezébe.
- 6-7 játékos esetén minden játékos 5 kártyát kap a kezébe.
- 8-10 játékos esetén minden játékos 4 kártyát kap a kezébe.

Az osztó játékos a megmaradt kártyákból egy lefordított paklit képez az asztalon a célkártyák mögött, ez lesz a húzó-pakli.

Az osztó játékos megkeveri az aranykártyákat, majd egy lefordított paklit képez belőlük a megmaradt törpés kártyapakli mellett.

A játékot a legfiatalabb játékos kezdi, a játék további menete az óramutató járásával megegyező irányú.

A játék előkészületei nem érdekesek a játékállás jellemzése szempontjából. A játékosok ülésrendje annyiban érdekes, hogy meghatároz egy sorrendet. Ezt egy bináris relációval írjuk le.

Szerepek: következőJátékosa, vanKártyája

Az éppen soron következő játékosnak ki kell játszania egy lapot a kezéből, amit a következők szerint tehet:

- A játékos vagy egy útkártyát illeszt a labirintushoz,
- vagy egy akciókártyát tesz egy játékostársa elé,
- vagy passzol. Ebben az esetben egy kártyát dob a lerakó-paklira.

Ezután a játékos új lapot húz a húzó-pakliból, ami a kezébe vesz. Ezzel a játékos lépése véget ér, a játékot a következő játékos folytatja.

Figyelem: Abban az esetben, ha elfogynak a húzó-pakliból a lapok, a játékosok nem tudnak új lapot húzni, már csak a kezeikben tartott lapokat tudják kijátszani.

Nem modellezzük, hogy egy kártya kijátszása milyen változásokhoz vezet, csupán azt, hogy a soron következő játékosnak mindig ki kell játszania egy kártyát.

Fogalmak: SoronkövetkezőJátékos

Szerepek: kijátszottKártya

~~Az útkártyákból áll össze az út a startkártyától a célkártyáig.~~ Útkártyát csak úgy tehet le a játékos, hogy annak illeszkednie kell legalább egy, már korábban lerakott útkártyához. Az útkártyán levő járatoknak és a már meglevő járatoknak illeszkedniük kell egymáshoz. Az útkártya nem rakható le oldalt forgatva. Az aranyásóknak az a céljuk, hogy kialakítsanak olyan járatot, amely megszakítás nélkül elvezet a startkártyától a célkártyáik valamelyikéig. A szabotőrök ezt megpróbálják megakadályozni. ~~A szabotőrök ezt persze igyekeznek nem feltűnően csinálni, nehogy lelepleződjenek.~~

Ahelyett, hogy megszorításokat tennénk arra, hogy hogyan lehet lerakni egy útkártyát (dinamikus modell), inkább megszorításokat teszünk arra, hogy hogyan nézhet ki egy lerakott labirintus (statikus modell). Az útkártyák összeillesztéséhez fontos, hogy a megfelelő oldalakon az utak illeszkedjenek, valamint az aranyhoz való eljutáshoz fontos, hogy egy kártyán belül mely oldalak vannak úttal összekötve. Ezért bevezetjük az **Oldal** és **ÚttalanOldal** fogalmakat. Egy kártyát és annak oldalait az **oldala** szerep kapcsol össze. Oldalak között pedig értelmezzük az **elérhető** relációt.

Fogalmak: Oldal, ÚttalanOldal
Szerepek: elérhető, oldala

~~Az akciókártyát a játékos felfordítva teszi le maga elé. A játékos egy akciókártyát kijátszhat maga előtt, vagy valamelyik játékostársa elé is lerakhatja. Akciókártya a játékosokat segítheti, vagy akadályozhatja is. A játékos egy akciókártyával eltávolíthat lapot a labirintusból, vagy éppen információt szerezhet valamelyik célkártyáról.~~

Ez a bekezdés csak felvázolja, hogy mi mindenre lehet akciókártyákat használni, nem elég pontos ahhoz, hogy formalizálni lehessen. A későbbiekben tisztázodni fog az akciókártyák használata.

~~Ha egy játékos lerak egy játékostársa elé egy akciókártyát (kijátssza ellene), az a játékostárs nem rakhat le útkártyát addig, amíg az akciókártya előtte fekszik. Más kártyát természetesen kijátszhat.~~ Egy játékos előtt maximum három akciókártya lehet, ~~minden fajtából egy.~~ Az a játékos tehet le útkártyát, aki előtt lépése megkezdésekor nincs akciókártya.

A középső sorban látható kártyákkal tudja a játékos eltávolítani az előtte levő akciókártyát. A játékos rátehet ilyen lapot egy saját maga elött fekvő akciókártyára, vagy egyik játékostársa előtt fekvő akciókártyára. ~~Miután a játékos a kártyát kiátszotta, mindkét kártya a lerakó-paklira kerül. Ha például egy törött csille akciókártya van a játékos előtt, akkor azt egy jó csillét ábrázoló kártyával eltüntetheti.~~

~~Az alsó sorban látható akciókártyákon két-két tárgy is látható. Abban az esetben, ha a játékos egy ilyen kártyát játszik ki, akkor a kártyán levő két tárgy bármelyikét megjavíthatja – de csak az egyiket, mindkettőt nem!~~

Figyelem: Minden javító kártyát csak olyan játékos ellen lehet kijátszani, aki előtt a szükséges akadályozó kártya van.

A modell egyszerűsítése érdekében nem különböztetjük meg a három különböző féle akadályozó- illetve javító kártyát. Ez egy önkényes döntés, mely információvesztéshez vezet. A modell felhasználásától függ, hogy mennyire megengedhető.

Fogalmak: Javítókártya, Akadályozókártya
Szerepek: előttelevőkártya

Akkor, mikor a játékos „kőomlás" kártyát játszik ki, egy tetszőleges útkártyát (kivéve a start és célkártya) kivehet a labirintusból. ~~A játékos a labirintusból kivett lapot a kijátszott kőomlás akciókártyával együtt a lerakó-paklira dobja.~~

Ennek az akciókártyának a segítségével a szabotőr megszakíthatja a célkártya felé haladó labirintus járatot. ~~Az aranyásó pedig kivehet egy zsákutcát a labirintusból. A kőomlás akciókártyával kivett útkártya helyét a játékosok később egy megfelelő útkártya elhelyezésével betölthetik.~~

Megintcsak a dinamizmus kihagyása miatt, nem tudjuk leírni, hogy milyen hatást idéz elő egy kőomlás kártya. Az viszont fontos, hogy ennek a kártyának köszönhetően elképzelhető, hogy kialakul olyan részlabirintus, ami nem kapcsolódik a startkártyához. Ilyen a járatkártyák lerakása közben nem jöhet létre, csak kőomlás segítségével.

Fogalmak: Kőomláskártya

~~Akkor, amikor a játékos „térkép" kártyát játszik ki, megnézhet egyet a három célkártya közül. A játékos a megnézett célkártyát visszarakja a helyére és senkinek nem mondja el, amit rajta látott, de ő már tudni fogja, hogy érdemes-e efelé a célkártya felé járatot építeni. Mert a három célkártyából csak az egyik célkártyán van aranyrög. Végül a játékos akciókártyáját a lerakó-paklira teszi.~~

Nem modellezzük a játékosok ismereteit.

Fogalmak: Térképkártya

~~Abban az esetben, ha egy játékos nem tud vagy nem akar kártyát kijátszani, akkor passzol. A játékos kezében tartott lapok közül egyet lefordítva a lerakó-pakli tetejére rak anélkül, hogy megmutatná játékostársainak azt. A lerakó-pakliban levő lapok egy része lefordítva egy része pedig felfordítva van. Egy forduló végénél előfordulhat az is, hogy egy játékosnak nincs már a kezében több kártya, ilyenkor a játékosnak passzolnia kell.~~

~~Akkor, amikor egy játékos útkártyával elér egy célkártyát, és az így lezárult járat megszakítás nélkül a startkártyától kiindulva a célkártyáig vezet, a játékos felfordítja azt a célkártyát.~~

- ~~Ha a célkártyán aranyrög található, a forduló véget is ért.~~
- ~~Ha a célkártyán kőszikla van, a forduló tovább folytatódik. A felfordított célkártyát úgy kell elhelyezni, hogy annak járata csatlakozzon az utóljára lerakott útkártya járatához.~~

~~Figyelem: Ritkán előfordul, hogy valamelyik útkártya járata nem illeszthető a már kialakított járathoz. Ez a célkártyáknál kivételesen megengedett.~~

A forduló abban az esetben is véget ér, ha már egyik játékosnak sincs kijátszható kártya a kezében.

~~Ebben az esetben a játékosok felfordítják a törpés kártyáikat. Na, ki volt az aranyásó, és ki a szabotőr?~~

A leírás vége már a játékmenet utáni teendőkről szól (aranyak elosztása, új játékmenet indítása), melyek modellünkön kívül vannak. Összességében az alábbi fogalmakat és szerepeket gyűjtöttük ki eddig:

- **Fogalmak**: Játékos, Aranyásó, Szabotőr, Nyertes, Útkártya, Startkártya, Célkártya, Aranykártya, Kőkártya, Labirintuskártya, SoronkövetkezőJátékos, Oldal, ÚttalanOldal, Javítókártya, Akadályozókártya, Kőomláskártya, Térképkártya
- **Szerepek**: következőJátékosa, vanKártyája, kijátszottKártyája, elérhető, oldala, előttelevőKártya

1.3. T-doboz készítés

Az előzőekben összegyűjtöttünk néhány fogalom és szerepnevet, valamint kihúztuk a játékleírás nagyrészét. Koncentráljunk most a megmaradt leírásra és formalizáljuk leíró logikai axiómákkal. Az axiómák megadják, hogy milyen jelentésbeli kapcsolat van az összegyűjtött fogalom- és szerepnevek között. Természetesen a formalizálás során megjelenhetnek új nevek is.

1.3.1. Könnyen fordítható részek

Bizonyos részeket egyből le tudunk fordítani axiómákra. Ezeket a részeket aláhúzással jelöljük és utánaírjuk a megfelelő axiómákat. Ahol a fordítás nem teljesen egyértelmű, az egyelőre ugorjuk át!

A játékosok lehetnek szorgos aranyászók, vagy pedig szabotőrök.

$$Jatekos \equiv Aranyaso \sqcup Szabotor$$

Ha az aranyászóknak sikerül eljutniuk az aranyhoz, akkor arany a jutalmuk. Ha ez nem sikerül nekik, akkor a szabotőrök győznek.

- **3 játékos esetén 1 szabotőr és 3 aranyász.**
- **4 játékos esetén 1 szabotőr és 4 aranyász.**
- **5 játékos esetén 2 szabotőr és 4 aranyász.**
- **6 játékos esetén 2 szabotőr és 5 aranyász.**
- **7 játékos esetén 3 szabotőr és 5 aranyász.**
- **8 játékos esetén 3 szabotőr és 6 aranyász.**
- **9 játékos esetén 3 szabotőr és 7 aranyász.**
- **10 játékos esetén 4 szabotőr és 7 aranyász.**

A játékhoz valamennyi játékos tartozik. Bevezetünk egy új fogalmat, melynek egyetlen példánya a *jatek* egyed, és ehhez kapcsolódnak *jatekosa* szerepekkel a játékosok.

$$\top \sqsubseteq \forall jatekosa.Jatekos$$
$$\top \sqsubseteq \forall jatekosa^{-}.\{jatek\}$$
$$\{jatek\} \equiv (\geq 3 jatekosa.\top) \sqcap (\leq 10 jatekosa.\top)$$
$$\leq 4 jatekosa.\top \sqsubseteq \leq 1 jatekosa.Szabotor$$
$$(\geq 5 jatekosa.\top) \sqcap (\leq 6 jatekosa.\top) \sqsubseteq (\geq 1 jatekosa.Szabotor) \sqcap (\leq 2 jatekosa.Szabotor)$$
$$(\geq 7 jatekosa.\top) \sqcap (\leq 9 jatekosa.\top) \sqsubseteq (\geq 2 jatekosa.Szabotor) \sqcap (\leq 3 jatekosa.Szabotor)$$
$$(= 10 jatekosa.\top) \sqsubseteq (\geq 3 jatekosa.Szabotor) \sqcap (\leq 4 jatekosa.Szabotor)$$

A 44 kártya között van egy startkártya és három célkártya. Az egyik célkártyán van az arany, a másik két célkártyán csak egy-egy kő látható. A játék folyamán a startkártyától kiindulva egy labirintusjárat alakul ki.

Az egyes kártyák számosságát úgy fejezzük ki, hogy korlátozzuk a *jatek* egyedből kimenő *jatekKartyaja* szerepkapcsolatokat.

$$Startkartya \sqsubseteq Utkartya$$
$$Celkartya \sqsubseteq Utkartya$$
$$Celkartya \equiv Aranykartya \sqcup Kokartya$$
$$\{jatek\} \sqsubseteq= 44jatekKartyaja.Utkartya$$
$$\{jatek\} \sqsubseteq= 1jatekKartyaja.Startkartya$$
$$\{jatek\} \sqsubseteq= 2jatekKartyaja.Kokartya$$
$$\{jatek\} \sqsubseteq= 1jatekKartyaja.Aranykartya$$

- 3-5 játékos esetén minden játékos 6 kártyát kap a kezébe.
- 6-7 játékos esetén minden játékos 5 kártyát kap a kezébe.
- 8-10 játékos esetén minden játékos 4 kártyát kap a kezébe.

$$(\leq 5jatekosa.\top) \sqsubseteq (\forall jatekosa.(\leq 6vanKartyaja.Kartya))$$
$$(\geq 6jatekosa.\top) \sqcap (\leq 7jatekosa.\top) \sqsubseteq (\forall jatekosa.(\leq 5vanKartyaja.Kartya))$$
$$(\geq 8jatekosa.\top) \sqcap (\leq 10jatekosa.\top) \sqsubseteq (\forall jatekosa.(\leq 4vanKartyaja.Kartya))$$
$$Kartya \equiv Utkartya \sqcup Akciokartya$$

Mivel a játék végén a játékosok kártyái fokozatosan elfogynak, ezért a fenti axiómák csak felső korlátot adnak a játékosok kártyaszámára vonatkozóan (hiszen az axiómáknak igaznak kell lenniük a játék minden fázisában).

Az éppen soron következő játékosnak ki kell játszania egy lapot a kezéből, amit a következők szerint tehet:

- A játékos vagy egy útkártyát illeszt a labirintushoz,
- vagy egy akciókártyát tesz egy játékostársa elé,
- vagy passzol.

$$SoronkovetkezoJatekos \sqsubseteq Jatekos$$
$$SoronkovetkezoJatekos \sqsubseteq (\leq 1kijatszottKartya.Kartya)$$
$$kijatszottKartya \sqsubseteq vanKartyaja$$

Útkártyát csak úgy tehet le a játékos, hogy annak illeszkednie kell legalább egy, már korábban lerakott útkártyához. Az útkártyán levő járatoknak és a már meglevő járatoknak illeszkedniük kell egymáshoz. Az útkártya nem rakható le oldalt forgatva. Az aranyásóknak az a céljuk, hogy kialakítsanak olyan járatot, amely megszakítás nélkül elvezet a startkártyától a célkártyák valamelyikéig. A szabotőrök ezt megpróbálják megakadályozni.

Egy játékos előtt maximum három akciókártya lehet. Az a játékos tehet le útkártyát, aki előtt lépése megkezdésekor nincs akciókártya. A középső sorban látható kártyákkal tudja a játékos eltávolítani az előtte levő akciókártyát. A játékos rátehet ilyen lapot egy saját maga elött fekvő akciókártyára, vagy egyik játékostársa előtt fekvő akciókártyára. Figyelem: Minden javító kártyát csak olyan játékos ellen lehet kijátszani, aki előtt a szükséges akadályozó kártya van. Akkor, mikor a játékos „kőomlás" kártyát játszik ki, egy tetszőleges útkártyát (kivéve a start és célkártya) kivehet a labirintusból. Ennek az akciókártyának a segítségével a szabotőr megszakíthatja a célkártya felé haladó labirintus járatot.

$Akciokartya \equiv Akadalyozokartya \sqcup Javitokartya \sqcup Koomlaskartya \sqcup Terkepkartya$
$Jatekos \sqsubseteq (\leq 3 elottelevoKartya.Akciokartya)$
$(\exists kijatszottKartya.Utkartya) \sqsubseteq (\forall elottelevoKartya.\bot)$
$\exists kijatszottKartya.Javitokartya \sqsubseteq \exists jatekosa^{-}.\exists jatekosa.\exists elottelevoKartya.Akadalyozokartya$
$\top \sqsubseteq (\forall elottelevoKartya.Akadalyozokartya)$

A forduló abban az esetben is véget ér, ha már egyik játékosnak sincs kijátszható kártya a kezében.

1.3.2. A hátralevő rész formalizálása

Mostanra a játékleírás nagy részét formalizáltuk, ráadásul viszonylag mechanikusan. Lássuk, mi van még hátra:

> **Ha az aranyásóknak sikerül eljutniuk az aranyhoz, akkor arany a jutalmuk. Ha ez nem sikerül nekik, akkor a szabotőrök győznek. A játék folyamán a startkártyától kiindulva egy labirintusjárat alakul ki.**
>
> **Útkártyát csak úgy tehet le a játékos, hogy annak illeszkednie kell legalább egy, már korábban lerakott útkártyához. Az útkártyán levő járatoknak és a már meglevő járatoknak illeszkedniük kell egymáshoz. Az útkártya nem rakható le oldalt forgatva. Az aranyásóknak az a céljuk, hogy kialakítsanak olyan járatot, amely megszakítás nélkül elvezet a startkártyától a célkártyák valamelyikéig. A szabotőrök ezt megpróbálják megakadályozni.**
>
> **A forduló abban az esetben is véget ér, ha már egyik játékosnak sincs kijátszható kártya a kezében.**

Két dolog hiányzik még a leírásunkból: a nyerés feltétele, valamint a lerakott útkártyák egymáshoz illesztésének megkötései. Az utóbbival kezdjük, hiszen a nyertes megállapításához éppen a labirintus tulajdonságait kell vizsgáljuk.

A labirintusban levő útkártyáknak illeszkedniük kell, így a modellben meg kell, hogy jelenjen, hogy a kártyák egyes oldalain van-e út vagy nincs. Másrészt az is fontos, hogy az egyes oldalak hogy vannak összekötve. Nem mindegy, hogy a kártyán egy négyes útkereszteződés van vagy pedig négy zsákutca. Erre szolgál az *Oldal* fogalom, illetve annak részfogalma, az *Uttalanoldal*. Az *elerheto* szerep oldalakat kapcsol össze. Minden útkártyának négy oldala van, a szélrózsa négy irányának megfelelően. Az *elerheto* szerepről előírjuk, hogy legyen tranzitív, reflexív és szimmetrikus. Ezen szerep segítségével nem tudjuk megkülönböztetni a zsákutcákat az úttalan oldalaktól, ezért fontos, hogy megjelenjen az *Uttalanoldal* mint atomi fogalom.

$$\top \sqsubseteq (\forall oldala.Oldal)$$
$$\top \sqsubseteq (\forall oldala^{-}.Utkartya)$$
$$eszakiOldala \sqsubseteq oldala$$
$$deliOldala \sqsubseteq oldala$$
$$keletiOldala \sqsubseteq oldala$$
$$nyugatiOldala \sqsubseteq oldala$$
$$Utkartya \sqsubseteq (= 1 eszakiOldala.Oldal)$$
$$Utkartya \sqsubseteq (= 1 deliOldala.Oldal)$$
$$Utkartya \sqsubseteq (= 1 keletiOldala.Oldal)$$
$$Utkartya \sqsubseteq (= 1 nyugatiOldala.Oldal)$$

$$Trans(elerheto)$$
$$Refl(elerheto)$$
$$elerheto \sqsubseteq elerheto^{-}$$
$$\top \sqsubseteq \forall elerheto.Oldal$$
$$\top \sqsubseteq \forall elerheto^{-}.Oldal$$

A kártyák között meg kell különböztessük azokat, amelyek le vannak rakva. Ezek alkotják a *Labirintuskartya* fogalmat. Egy labirintuskártyának lehet legfeljebb egy északi, déli, keleti valamint nyugati szomszédja, melyekhez illeszkednie kell.

$$Labirintuskartya \sqsubseteq Utkartya$$
$$\top \sqsubseteq (\forall szomszedja.Labirintuskartya)$$
$$\top \sqsubseteq (\forall szomszedja^{-}.Labirintuskartya)$$
$$szomszedja \sqsubseteq szomszedja^{-}$$
$$eszakiSzomszedja \sqsubseteq szomszedja$$
$$deliSzomszedja \sqsubseteq szomszedja$$
$$keletiSzomszedja \sqsubseteq szomszedja$$
$$nyugatiSzomszedja \sqsubseteq szomszedja$$
$$Labirintuskartya \sqsubseteq (\leq 1 eszakiSzomszedja.\top)$$
$$Labirintuskartya \sqsubseteq (\leq 1 deliSzomszedja.\top)$$
$$Labirintuskartya \sqsubseteq (\leq 1 keletiSzomszedja.\top)$$
$$Labirintuskartya \sqsubseteq (\leq 1 nyugatiSzomszedja.\top)$$
$$Uttalanoldal \sqsubseteq (\forall eszakiOldala^{-}.(\forall eszakiSzomszedja.(\forall deliOldala.Uttalanoldal)))$$
$$Uttalanoldal \sqsubseteq (\forall deliOldala^{-}.(\forall deliSzomszedja.(\forall eszakiOldala.Uttalanoldal)))$$
$$Uttalanoldal \sqsubseteq (\forall keletiOldala^{-}.(\forall keletiSzomszedja.(\forall nyugatiOldala.Uttalanoldal)))$$
$$Uttalanoldal \sqsubseteq (\forall nyugatiOldala^{-}.(\forall nyugatiSzomszedja.(\forall keletiOldala.Uttalanoldal)))$$

Ha két labirintuskártya illeszkedik egymáshoz, akkor a megfelelő oldalak elérhetőek. Kicsit zavaró lehet, hogy ennek értelmében két illeszkedő oldal, melyeken nincs út szintén elérhetőek egymásból, de ez nem fogja rontani a modell használhatóságát, hiszen az ilyen oldalpárok sehonnan máshonnan nem érhetőek el, így biztosan nem tudjuk felhasználni a köztük levő „virtuális" összeköttetést. Megtehetnénk ugyan, hogy csak akkor tekintjük a szomszédos oldalakat eléhetőnek, ha azok nem úttalanok, de ennek megfogalmazásához szerep szűkítést kellene használni, és ha egy mód van rá, inkább kerüljük az összetett szerepkonstruktorok használatát, hiszen ezek könnyen elrontják a modell eldönthetőségét. Viszont szerepszűkítés nélkül is szükségünk van egy szerepkonstruktorra, a kompozícióra:

$$eszakiOldala^{-} \circ eszakiSzomszedja \circ deliOldala \sqsubseteq elerheto$$
$$deliOldala^{-} \circ deliSzomszedja \circ eszakiOldala \sqsubseteq elerheto$$
$$keletiOldala^{-} \circ keletiSzomszedja \circ nyugatiOldala \sqsubseteq elerheto$$
$$nyugatiOldala^{-} \circ nyugatiSzomszedja \circ keletiOldala \sqsubseteq elerheto$$

Most már azt is le tudjuk írni, hogy mikor érhető el egy oldalról az aranykártya:

$$Aranyhozkotott \equiv Oldal \sqcap (\exists elerheto.(\exists oldala^{-}.Aranykartya))$$

Az aranyászók pontosan akkor nyernek, ha a startkártya egyik oldaláról elérhető az aranykártya. A szabotőrök pedig akkor nyernek, ha egyetlen játékosnak sincs már lapja.

$$Aranyaso \sqcap \exists jatekosa^{-}.\exists jatekKartyaja.(Startkartya \sqcap \exists oldala.Aranyhozkotott) \sqsubseteq Nyertes$$
$$Szabotor \sqcap \exists jatekosa^{-}.\forall jatekosa.\forall vanKartyaja.\bot \sqsubseteq Nyertes$$

1.3.3. A kapott T-doboz

Elkészült egy leíró logikai T-doboz, mely az eredeti játék szabályrendszerének egy közelítése. A T-doboz az alábbi axiómákat tartalmazza:

$$Jatekos \equiv Aranyaso \sqcup Szabotor$$
$$\top \sqsubseteq \forall jatekosa.Jatekos$$
$$\top \sqsubseteq \forall jatekosa^{-}.\{jatek\}$$
$$\{jatek\} \equiv (\geq 3 jatekosa.\top) \sqcap (\leq 10 jatekosa.\top)$$
$$\leq 4 jatekosa.\top \sqsubseteq \leq 1 jatekosa.Szabotor$$
$$(\geq 5 jatekosa.\top) \sqcap (\leq 6 jatekosa.\top) \sqsubseteq (\geq 1 jatekosa.Szabotor) \sqcap (\leq 2 jatekosa.Szabotor)$$
$$(\geq 7 jatekosa.\top) \sqcap (\leq 9 jatekosa.\top) \sqsubseteq (\geq 2 jatekosa.Szabotor) \sqcap (\leq 3 jatekosa.Szabotor)$$
$$(= 10 jatekosa.\top) \sqsubseteq (\geq 3 jatekosa.Szabotor) \sqcap (\leq 4 jatekosa.Szabotor)$$
$$Startkartya \sqsubseteq Utkartya$$
$$Celkartya \sqsubseteq Utkartya$$
$$Celkartya \equiv Aranykartya \sqcup Kokartya$$
$$\{jatek\} \sqsubseteq = 44 jatekKartyaja.Utkartya$$
$$\{jatek\} \sqsubseteq = 1 jatekKartyaja.Startkartya$$
$$\{jatek\} \sqsubseteq = 2 jatekKartyaja.Kokartya$$
$$\{jatek\} \sqsubseteq = 1 jatekKartyaja.Aranykartya$$
$$(\leq 5 jatekosa.\top) \sqsubseteq (\forall jatekosa.(\leq 6 vanKartyaja.Kartya))$$
$$(\geq 6 jatekosa.\top) \sqcap (\leq 7 jatekosa.\top) \sqsubseteq (\forall jatekosa.(\leq 5 vanKartyaja.Kartya))$$
$$(\geq 8 jatekosa.\top) \sqcap (\leq 10 jatekosa.\top) \sqsubseteq (\forall jatekosa.(\leq 4 vanKartyaja.Kartya))$$
$$Kartya \equiv Utkartya \sqcup Akciokartya$$
$$SoronkovetkezoJatekos \sqsubseteq Jatekos$$
$$SoronkovetkezoJatekos \sqsubseteq (\leq 1 kijatszottKartya.Kartya)$$
$$kijatszottKartya \sqsubseteq vanKartyaja$$
$$Akciokartya \equiv Akadalyozokartya \sqcup Javitokartya \sqcup Koomlaskartya \sqcup Terkepkartya$$
$$Jatekos \sqsubseteq (\leq 3 elottelevoKartya.Akciokartya)$$
$$(\exists kijatszottKartya.Utkartya) \sqsubseteq (\forall elottelevoKartya.\bot)$$
$$\top \sqsubseteq (\forall elottelevoKartya.Akadalyozokartya)$$
$$\exists kijatszottKartya.Javitokartya \sqsubseteq \exists jatekosa^{-}.\exists jatekosa.\exists elottelevoKartya.Akadalyozokartya$$
$$\top \sqsubseteq (\forall oldala.Oldal)$$
$$\top \sqsubseteq (\forall oldala^{-}.Utkartya)$$
$$eszakiOldala \sqsubseteq oldala$$
$$deliOldala \sqsubseteq oldala$$
$$keletiOldala \sqsubseteq oldala$$
$$nyugatiOldala \sqsubseteq oldala$$
$$Utkartya \sqsubseteq (= 1 eszakiOldala.Oldal)$$

$Utkartya \sqsubseteq (= 1 deliOldala.Oldal)$
$Utkartya \sqsubseteq (= 1 keletiOldala.Oldal)$
$Utkartya \sqsubseteq (= 1 nyugatiOldala.Oldal)$
$Trans(elerheto)$
$Refl(elerheto)$
$elerheto \sqsubseteq elerheto^{-}$
$\top \sqsubseteq \forall elerheto.Oldal$
$\top \sqsubseteq \forall elerheto^{-}.Oldal$
$Labirintuskartya \sqsubseteq Utkartya$
$\top \sqsubseteq (\forall szomszedja.Labirintuskartya)$
$\top \sqsubseteq (\forall szomszedja^{-}.Labirintuskartya)$
$szomszedja \sqsubseteq szomszedja^{-}$
$eszakiSzomszedja \sqsubseteq szomszedja$
$deliSzomszedja \sqsubseteq szomszedja$
$keletiSzomszedja \sqsubseteq szomszedja$
$nyugatiSzomszedja \sqsubseteq szomszedja$
$Labirintuskartya \sqsubseteq (\leq 1 eszakiSzomszedja.\top)$
$Labirintuskartya \sqsubseteq (\leq 1 deliSzomszedja.\top)$
$Labirintuskartya \sqsubseteq (\leq 1 keletiSzomszedja.\top)$
$Labirintuskartya \sqsubseteq (\leq 1 nyugatiSzomszedja.\top)$
$Uttalanoldal \sqsubseteq (\forall eszakiOldala^{-}.(\forall eszakiSzomszedja.(\forall deliOldala.Uttalanoldal)))$
$Uttalanoldal \sqsubseteq (\forall deliOldala^{-}.(\forall deliSzomszedja.(\forall eszakiOldala.Uttalanoldal)))$
$Uttalanoldal \sqsubseteq (\forall keletiOldala^{-}.(\forall keletiSzomszedja.(\forall nyugatiOldala.Uttalanoldal)))$
$Uttalanoldal \sqsubseteq (\forall nyugatiOldala^{-}.(\forall nyugatiSzomszedja.(\forall keletiOldala.Uttalanoldal)))$
$eszakiOldala^{-} \circ eszakiSzomszedja \circ deliOldala \sqsubseteq elerheto$
$deliOldala^{-} \circ deliSzomszedja \circ eszakiOldala \sqsubseteq elerheto$
$keletiOldala^{-} \circ keletiSzomszedja \circ nyugatiOldala \sqsubseteq elerheto$
$nyugatiOldala^{-} \circ nyugatiSzomszedja \circ keletiOldala \sqsubseteq elerheto$
$Aranyhozkotott \equiv Oldal \sqcap (\exists elerheto.(\exists oldala^{-}.Aranykartya))$
$Aranyaso \sqcap \exists jatekosa^{-}.\exists jatekKartyaja.(Startkartya \sqcap \exists oldala.Aranyhozkotott) \sqsubseteq Nyertes$
$Szabotor \sqcap \exists jatekosa^{-}.\forall jatekosa.\forall vanKartyaja.\bot \sqsubseteq Nyertes$

Vegyük észre, hogy bár a modellezés elején felsorolt szerepnevek között szerepel a *kovetkezoJatekosa* szerep, viszont ez sehol nem jelenik meg az axiómákban. Bár a tényleges játékban van jelentősége annak, hogy ki után ki következik, az egyszerűsített statikus modellben ez a jelentőség elveszik. Ezért ezért a következőJátékosa szerepet kihagyjuk a modellből.

1.4. A-doboz készítés

Egy leíró logikai tudásbázis két részből áll: T-dobozból és A-dobozból. Az előző bekezdésben összegyűjtöttük a T-doboz axiómáit, ezek azok a részek, melyek játékról játékra változatlanok. Ezzel szemben az A-doboz tartalma az aktuális játékállást tükrözi. Az elkészített ontológiának tipikus célja lehet, hogy a T-doboz segítségével ellenőrizze az A-doboz helyességét, azaz hogy a kialakult játékhelyzet szabályos. A helyességvizsgálaton túl persze felmerülhetnek olyan kérdések is, hogy egy adott állás alapján ki nyert, vagy mely oldalakról érhető el az aranykártya.

Tekintsünk egy olyan A-dobozt mely az aranyások számára a lehető legyegyszerűbb nyertes állást írja le: összesen hét lerakott labirintuskártya van a középső sorban és mindegyiken keletről nyugatra

megy út. Az aranykártya pedig éppen középen van. Legyen három aranyásónk és két szabotőrünk. Az A-dobozban felsoroljuk, hogy az egyedek milyen fogalomba tartoznak, valamint, hogy milyen szerepkapcsolatokban vesznek részt. Az öt játékosról az alábbi állításokat tesszük:

$$Aranyaso(aranyaso1),\ jatekosa(jatek, aranyaso1)$$
$$Aranyaso(aranyaso2),\ jatekosa(jatek, aranyaso2)$$
$$Aranyaso(aranyaso3),\ jatekosa(jatek, aranyaso3)$$
$$Szabotor(szabotor1),\ jatekosa(jatek, szabotor1)$$
$$Szabotor(szabotor2),\ jatekosa(jatek, szabotor2)$$

A játékban van egy startkártya, egy aranykártya és hét labirintuskártya:

$$Startkartya(sk),\ jatekKartyaja(jatek, sk)$$
$$Labirintuskartya(lk1),\ jatekKartyaja(jatek, lk1)$$
$$Labirintuskartya(lk2),\ jatekKartyaja(jatek, lk2)$$
$$Labirintuskartya(lk3),\ jatekKartyaja(jatek, lk3)$$
$$Labirintuskartya(lk4),\ jatekKartyaja(jatek, lk4)$$
$$Labirintuskartya(lk5),\ jatekKartyaja(jatek, lk5)$$
$$Labirintuskartya(lk6),\ jatekKartyaja(jatek, lk6)$$
$$Labirintuskartya(lk7),\ jatekKartyaja(jatek, lk7)$$
$$Aranykartya(ak),\ jatekKartyaja(jatek, ak)$$

Le kell írni, hogy a lerakott kártyák egy kelet-nyugati vonalon vannak egymás mellett:

$$keletiSzomszedja(sk, lk1),\ nyugatiSzomszedja(lk1, sk)$$
$$keletiSzomszedja(lk1, lk2),\ nyugatiSzomszedja(lk2, lk1)$$
$$keletiSzomszedja(lk2, lk3),\ nyugatiSzomszedja(lk3, lk2)$$
$$keletiSzomszedja(lk3, lk4),\ nyugatiSzomszedja(lk4, lk3)$$
$$keletiSzomszedja(lk4, lk5),\ nyugatiSzomszedja(lk5, lk4)$$
$$keletiSzomszedja(lk5, lk6),\ nyugatiSzomszedja(lk6, lk5)$$
$$keletiSzomszedja(lk6, lk7),\ nyugatiSzomszedja(lk7, lk6)$$
$$keletiSzomszedja(lk7, ak),\ nyugatiSzomszedja(ak, lk7)$$

Minden labirintuskártyának van keleti és nyugat oldala, melyek egymásból elérhetőek.

$$keletiOldala(lk1, klk1),\ nyugatiOldala(lk1, nylk1)$$
$$keletiOldala(lk2, klk2),\ nyugatiOldala(lk2, nylk2)$$
$$keletiOldala(lk3, klk3),\ nyugatiOldala(lk3, nylk3)$$
$$keletiOldala(lk4, klk4),\ nyugatiOldala(lk4, nylk4)$$
$$keletiOldala(lk5, klk5),\ nyugatiOldala(lk5, nylk5)$$
$$keletiOldala(lk6, klk6),\ nyugatiOldala(lk6, nylk6)$$
$$keletiOldala(lk7, klk7),\ nyugatiOldala(lk7, nylk7)$$
$$keletiOldala(sk, ksk),\ nyugatiOldala(sk, nysk)$$
$$keletiOldala(ak, kak),\ nyugatiOldala(ak, nyak)$$
$$elerheto(klk1, nylk1)$$
$$elerheto(klk2, nylk2)$$
$$elerheto(klk3, nylk3)$$

$$elerheto(klk4, nylk4)$$
$$elerheto(klk5, nylk5)$$
$$elerheto(klk6, nylk6)$$
$$elerheto(klk7, nylk7)$$

Hasonló módon bele kell venni a két kőkártyát és a játékosok kezében levő kártyákat is az A-dobozba. Ettől az egyszerűség kedvéért most eltekintünk.

1.5. OWL ontológia építése a Protégé program segítségével

Az előző két bekezdésben összeállítottunk egy T-dobozt és egy A-dobozt. Ebből most OWL dokumentumot szeretnénk készíteni. Ehhez egy nagyon kényelmes segédeszköz a Protégé ontológiakészítő program, mi is ezt fogjuk használni.

A program szabadon letölthető a `http://protege.stanford.edu/` címről, ahol különféle felhasználói kézikönyveket is lehet találni. A Protégé használatát nem részletezzük, sok mindenre az ember használat közben rájöhet, illetve érdemes tanulmányozni az említett felhasználói kézikönyvet.

A Protégé segítségével készült ontológia más filozófiát követ, mint ahogy eddig haladtunk. Mi axiómákat gyűjtöttünk össze, hogy a játék szabályait megragadjuk. Protégé-ben az ember először fogalmakat és szerepeket hoz létre. Ezután tulajdonságokat rendelünk hozzájuk. Ha például vesszük az alábbi axiómát:

$$Aranyhozkotott \equiv Oldal \sqcap (\exists elerheto.(\exists oldala^{-}.Aranykartya))$$

Akkor először létrehozzuk az *Aranyhozkotott Oldal* és *Aranykartya* fogalmakat valamint az *oldala* szerepet, majd az *Aranyhozkotott* fogalomhoz hozzárendelünk egy vele ekvivalens fogalmat: $Oldal \sqcap (\exists elerheto.(\exists oldala^{-}.Aranykartya))$.

Ez a megközelítés nem jelent korlátozást, minden axiómát át lehet fogalmazni úgy, hogy az valamilyen fogalom vagy szerep egy tulajdonságaként jelentkezzen. Például tetszőleges fogalomtartalmazási axióma bennsősíthető, ami által kapunk egy olyan fogalmat, amit hozzárendelhetünk a $\top$ fogalomhoz mint befoglaló fogalom. Másrészről a Protégé segít bizonyos hiányok felderítésében. Például azonnal látszik, ha egy szerepnek nem adtuk meg az értelmezési tartományát és értékkészletét. Az eddig elkészített ontológiánk szép számmal tartalmaz ilyen szerepeket, mint például a *kijatszottKartya* vagy *vanKartyaja*. Ezeket érdemes pótolni.

Az elkészült ontológiát el tudjuk menteni közvetlenül OWL formátumban.

1.6. Következtetés

A Protégé támogatást nyújt bizonyos következtetési feladatok elvégzésére. Automatikus következtetést több okból is használnak ontológiákon:

- A következtetés segíti az ontológiakészítést illetve a modellezési hibák felderítését avval, hogy rámutat a modell nem kívánt következményeire.
- Következtetéssel leegyszerűsíthetjük a modellünket. Ha például egy gráfot modellezünk és szeretnénk, hogy a modellben megjelenjen, hogy mely pontok mely másik pontokból érhetőek el, akkor nem kell kézzel felvennünk minden azonos komponensen belüli pontpárra az elérhető szerepet, elég ezt megtenni a szomszédokra és kijelenteni, hogy az elérhető szerep tranzitív. A rendszer ezután már magától fel tudja venni a hiányzó szerepkapcsolatokat. Mindezt úgy is mondhatjuk, hogy ahelyett, hogy egy szabályszerűség példányait kimerítő módon felsorolnánk az A-dobozban, megfogalmazhatunk néhány T-doboz beli állítást, amit a következtető kiterjeszt az A-dobozbeli egyedekre.

- Ha már elkészült az ontológia, akkor a modell felhasználói kérdéseket fogalmazhatnak meg. Tipikus kérdések, lehetnek, hogy két fogalom között milyen kapcsolat van, vagy hogy egy fogalomnak kik az egyedei.

Az Aranyások játékból készített modellünkben találunk példát mindhárom típusú következtetésre.

1.6.1. Modellezési hibák felderítése

Vegyük fel az összegyűjtött axiómákat a Protégé-ben! Ezután a *Reasoner* menűpontban kiválaszthatjuk, hogy melyik külső következtető programot szeretnénk használni. Az alap csomagban már benne van a FaCT++ és a Hermit, de lehet használni más rendszereket is megfelelő plug-in letöltésével. Népszerű leíró logikai következtetők még a Racer és Pellet rendszerek.

Válasszuk ki például a FaCT++ programot. Ezután, ha elindítjuk a következtetőt (*Start reasoner*), akkor érdekes eredményt kapunk. Az általunk eddig épített osztályhierarchia mellett megnézhető a kikövetkeztetett osztályhierarchia. Ebben azt látjuk, hogy $Thing \equiv Oldal$, azaz a hogy minden egyed beletartozik az *Oldal* fogalomba![1] Erre nem számítottunk, úgyhogy valahol hibát követtünk el. Szerencsére a Protégé abban is segít, hogy megmutatja, hogy mit miből következtetett ki (*Explain inference*). Az alábbi üzenetet kapjuk:

$$Reflexive : elerheto$$
$$elerheto\ Range\ Oldal$$

Az elérhető szerepről kijelentettük, hogy az értékkészlete az Oldal fogalom, és azt is, hogy a szerep reflexív. A reflexivitásból következik, hogy minden egyed önmagából elérhető, tehát a szerep tényleges értékkészlete a modell teljes tartománya, azaz $\top$. Ebből pedig tényleg következik, hogy $Thing \equiv Oldal$. A probléma egyszerű megoldása, hogy töröljük a reflexivitásról szóló axiómát.

1.6.2. Az A-doboz egyszerűsítése

A fenti A-dobozban felsoroltuk, hogy az i-dik labirintuskártya keleti szomszédja az $i+1$-dik labirintuskártya, valamint hogy az $i+1$-dik labirintuskártya nyugati szomszédja az i-dik labirintuskártya. A második állítás következik az elsőből és sokkal kényelmesebb lenne, ha feleslegesen nem kellene adatállításokat megfogalmazni. Ezért érdemes még tovább „okosítani" a T-dobozt: könnyedén leírhatjuk, hogy az keletiSzomszédja és nyugatiSzomszédja szerepek egymás inverzei. Hasonlóan járunk el az északiSzomszédja és déliszomszédja szereppárral és az alábbi axiómákkal egészítjük ki a T-dobozt:

$$deliSzomszedja \equiv inv(eszakiSzomszedja)$$
$$keletiSzomszedja \equiv inv(nyugatiSzomszedja)$$

Az automatikus következtetésnek köszönhetően elég az egyik kártyáról kijelenteni, hogy a másik szomszédja, fordítva nem kell.

1.6.3. Lekérdezések

Miután elkészült a modell, kérdéseket fogalmazhatunk meg (*DL Query*). Egy kérdés egy tetszőlegesen komplex leíró logikai fogalom lehet, melyről a következtető megállapítja, hogy kik az egyedei, milyen fogalmakat tartalmaz illetve milyen fogalmak tartalmazzák őt. Ha például a kérdés a *Nyertes* fogalom, akkor válaszként azt kapjuk, hogy a három aranyásó (*aranyaso1*, *aranyaso2*, *aranyaso3*) nyertes, valamint hogy minden nyertes része a *Jatekos* fogalomnak.

[1] A Protégé $\top$ és $\bot$ fogalmak helyett *Thing* és *Nothing* fogalmakat használ.

1.7. Összefoglalás

Ebben a fejezetben az Aranyások társasjáték modellezésére készítettünk leíró logikai ontológiát. Összegzésként tekintsük újra a modellezés néhány fontos aspektusát.

Első lépésként meg kell határozni, hogy pontosan mit szeretnénk modellezni, mely részek fontosak és melyeket lehet elhanyagolni. Különböző modellező eszközök különféle modellek készítésére alkalmasak. Leíró logikák segítségével statikus, „világleíró" modelleket lehet kényelmesen készíteni.

Ha már tudjuk, hogy mit szeretnénk modellezni, akkor érdemes összegyűjteni, hogy milyen szintaktikai elemek kerüljenek bele a modellbe. Leíró logikák esetén tehát meg kell határozni, hogy milyen fogalmak, szerepek illetve egyedek forduljanak elő a modellben.

A modellezendő világ informális leírásából érdemes eltávolítani azokat a részeket, melyeket nem kell formalizálni. Gondolhatunk itt a játék modellen kívül eső részére, a redundáns leírásokra, vagy a túl általános részekre, melyek később úgyis pontosításra kerülnek. Ezáltal egy sokkal tömörebb, átláthatóbb leírást kapunk.

Ezután hozzáláthatunk a tényleges modellkészítéshez. A megmaradó leírást próbáljuk minél pontosabban lefordítani a modell nyelvére. A modellezés gyakran iteratív feladat, későbbi részek formalizálásakor kiderülhet, hogy érdemes átírni egy korábbi részt.

Az elkészült modell értékét nagyban növeli, ha van hozzá automatikus következtetési támogatás. Ez talán a legfőbb előnye a leíró logikai ontológiáknak más modellezési nyelvvel szemben. A következtetés segít a modellezési hibák felderítésében, könnyíti a modellépítést azáltal, hogy nem kell minden információt expliciten megjeleníteni, valamint lehetővé teszi, hogy a felhasználók kérdéseket tegyenek fel a modellről.

2. fejezet

Korlátkövetkeztetés gyorsítása kombinatorikai korlátok használatával

A következőkben megismerkedünk a SICStus 4 Prolog CLPFD könyvtárában található legfontosabb kombinatorikai korlátokkal. Ezen könyvtár segítségével egészen változatos globális korlátokat lehet megfogalmazni tetszőlegesen sok változó együttes korlátozására, és a korlát szűkítési mechanizmusát is finomra lehet hangolni. Célunk, hogy megmutassuk, megfelelő hangolással tetszőleges relációt le lehet írni korlátokkal. A fejezet tartalmazza sok korlát rövid leírását, valamint programozási feladatokat, melyek az adott korlát használhatóságát szemléltetik. A korlátok ismertetése helyenként részleges, a teljes leírást a SICstus Prolog felhasználói kézikönye tartalmazza.

2.1. count, global_cardinality

A következőkben az alábbi két korlátot vizsgáljuk meg:

- `count(+Val,+List,+RelOp,?Count)`: A korlát segítségével össze lehet számolni egy elem előfordulásait egy listában. `Val` egy egész szám, `List` egészek vagy egész értékű változók listája, `RelOp` egy összehasonlító szimbólum, `Count` pedig egy egész vagy egész értékű változó. A korlát igaz, ha `Val` `N`-szer fordul elő `List`-ben és teljesül, hogy `N RelOp Count`.
- `global_cardinality(+Xs,+Vals)`: A `count` korlát általánosítása. `Xs=[X1,...,Xd]` egy egészekből vagy egész értékű változókból álló lista, `Vals=[K1-V1,...,Kn-Vn]` párok listája, ahol `Ki` egy egyedi egész szám, `Vi` pedig egész szám vagy egész értékű változó. A korlát igaz, ha `Xs` minden eleme megegyezik valamelyik `Ki`-vel, továbbá `Xs`-nek pontosan `Vi` eleme egyezik meg `Ki`-vel.

1. Feladat. *Birodalmunkat szeretnénk biztonságban tartani. Ez a birodalom egy $n*n$-es négyzetrács, melynek egyes mezőire építhetünk várakat, bástyákat illetve őrtornyokat, a maradék pedig mező marad. Az alábbi elvárásoknak kell eleget tennünk:*

- *Északnyugaton barátaink élnek, akiket nem akarunk megijeszteni, így a bal felső sarok maradjon parlagon.*
- *Északkeleten szövetségeseink vannak, nagyon nem kell tőlük félni, de azért egy őrtoronnyal mindenképp szemmel kell őket tartani a jobb felső sarokban.*
- *Délnyugaton barbár törzsek laknak, ezért a bal alsó sarokba bástyát szeretnénk építeni.*
- *Délkeleten lakik a gonosz maga, így a jobb alsó sarokba erős várat tervezünk.*

Teljesülniük kell továbbá, az alábbi feltételeknek:

- *Minden mező élszomszédságában van vagy legalább 1 vár, vagy legalább 2 bástya vagy 3 őrtorony*
- *Egy őrtorony élszomszédságában pontosan egy darab erőd van, ami lehet vár is és bástya is.*
- *Bástya és vár mellé közvetlenül nem építünk se várat se bástyát.*

Készíts Prolog eljárást a birodalom megtervezésére, az alábbi specifikációk szerint.
`birodalom(+N,+Torony,+Bastya,+Var,+Mod,?Terkep)`: ***`Terkep`*** *egy* ***`N*N`****-es mátrix (***`N`** db **`N`** hosszú listából álló lista), melynek elemei a 0 (mező), 1 (őrtorony), 2 (bástya), 3 (vár) értékeket vehetik fel. Továbbá teljesül, hogy* ***`Terkep`*** *pontosan* ***`Torony`*** *darab őrtornyot,* ***`Bastya`*** *darab bástyát és* ***`Var`*** *darab várat tartalmaz.*

Az eljárást 3 különböző változatban kell elkészíteni, melyek közül a ***`Mod`*** *argumentum segítségével lehet választani.* ***`Mod`*** *értéke 1, 2 és 3 lehet az alábbiak szerint:*

- ***`Mod=1`*** *Az eljárás kötelezően kell, hogy használja a* ***`global_cardinality/2`*** *kombinatorikus korlátot.*
- ***`Mod=2`*** *Az eljárás nem használhat* ***`global_cardinality/2`*** *korlátot, viszont kell használja a* ***`count/4`*** *korlátot.*
- ***`Mod=3`*** *Az eljárás se a* ***`global_cardinality/2`*** *se a* ***`count/4`*** *korlátokat nem használhatja, csak reifikálást.*

Hasonlítsd össze a három megoldás futási idejét, abban a konkrét esetben, mikor egy $6*6$-os birodalmunk van és pontosan 5 őrtorony, 7 bástya és 4 vár építésére van erőforrásunk!

Az elkészült Prolog program egyetlen állományban legyen, melynek neve ***`{Vezetéknév}.pl`****. A programot helyezd el a vezetékneveddel azonos nevű modulba, de ne exportálj semmit. Tehát ha például Zombori Zsolt a neved, akkor a program kerüljön a* ***`zombori.pl`*** *állományba, melynek első sora az alábbi modul deklaráció:*

```
:- module(zombori, []).
```

2.2. case

- `case(+Template,+Tuples,+Dag [,+Options])`: A korlát segítségével tetszőleges n argumentumú relációt le lehet írni egy DAG segítségével. A DAG-nak egyetlen gyökere van és minden levélbe vezető útja n hosszú. A csomópontok a reláció elemeinek felelnek meg. Az éleken `(Min..Max)` tartományok valamint lineáris egyenlőtlenségek szerepelnek. A korlát pontosan akkor teljesül egy tömör számennesre, ha van olyan út a gyökértől egy levélig, ahol a k. elem benne van az út k. élére írt tartományban, valamint az úthoz tartozó lineáris egyenlőtlenségek mindegyike teljesül. `Template` egy Prolog kifejezés, mely n különböző változót tartalmaz. `Tuples` Prolog kifejezések listája, melyeknek a strukturája megegyezik `Template` strukturájával. Nem tartalmazhat közös változót `Template`-tel. `Dag` csomópontok listája. Egy csomópont `node(ID,X,Children)` alakú, ahol `X` egy változó `Template`-ből, `ID` pedig egy egész, mely egyedileg azonosítja a csomópontot. Az első csomópont a gyökér. Ha egy csomópont belső csomópont, akkor `Children` olyan lista, mely `(Min..Max)-ID2` vagy `(Min..Max)-SideConstraints-ID2` alakú kifejezésekből áll, ahol `ID2` egy gyerek csomópontot azonosít. Leveleknél `Children` `(Min..Max)` vagy `(Min..Max)-SideConstraints` alakú kifejezésekből álló lista. `Options` opciólista segítségével különböző ébredési és szűkítési szinteket határozhatunk meg.

A korlát teljes leírása megtalálható a SICStus kézikönyvben.

2. Feladat. *Készíts egy `square/5` eljárást az alábbiak szerint:*
`square(+N,+XSumList,+YSumList,+Mode,?Res)`: `Res` egy `N*N`-es mátrix, `XSumList` és `YSumList` pedig `N` hosszú listák. `Res` mátrix elemei 2 és 9 közötti számjegyek és teljesül, hogy két szomszédos számjegy mindig különböző. `Res` sorainak összegét rendre `XSumList` elemei adják. Hasonlóan, `Res` oszlopainak összegei `YSumList`-ben vannak felsorolva. Továbbá, bármely sorból illetve oszlopból választunk ki három egymást követő A, B, C elemet, teljesül rájuk az alábbi tulajdonság: ha A és B relatív prímek, akkor C páros, különben páratlan.

A feladatnak van egy olyan verziója is, melyben a fenti reláció három egymás után álló elemre visszafelé is teljesül. A relációt le lehet írni egy blokkolt eljárás segítségével, vagy a `case` korláttal. Több megoldást kell elkészíteni, melyek közül a `Mod` paraméter segítségével választunk:

- *`Mod=case(Dir,Together,Options)`*
- *`Mod=block(Dir)`*

`Dir` határozza meg, hogy csak előre, vagy mindkét irányban teljesül a reláció:

- *`Dir=forw`: csak előre*
- *`Dir=both`: mindkét irányba*

`Together` határozza meg, egyetlen nagy `case` korlátot veszünk fel, vagy minden hármashoz külön egyet:

- *`Together=no`: külön `case` korlát minden hármashoz*
- *`Together=yes`: egyetlen `case` korlátot veszünk fel*

`Options` segítségével különböző ébredési szinteket állíthatunk be a `case` korlátnál:

- *`Options=dom`: a `case` korlát akkor ébredjen fel, ha bármelyik változójának tartománya változik.*
- *`Options=minmax`: a `case` korlát akkor ébredjen fel, ha bármelyik változójának intervalluma változik.*
- *`Options=val`: a `case` korlát csak akkor ébredjen fel, ha valamelyik változója behelyettesítődik.*

Példák:

```
square(3,[7,9,9],[7,9,9],case(minmax,forw,no),X).
X = [[2,3,2],
     [3,2,4],
     [2,4,3]]

-------------------------------------------------

X=[[5,_,_,_,_,_,5],
   [_,4,_,_,_,3,_],
   [_,_,_,_,_,_,_],
   [_,_,_,_,_,_,_],
   [_,_,3,_,4,_,_],
   [_,7,_,_,_,2,_],
   [3,_,_,_,_,_,5]],
square(7,[25,26,26,23,24,32,34],[27,32,25,25,22,30,29],block(forw),X).


[5,2,4,3,2,4,5]   [5,2,4,3,2,4,5]   [5,2,4,3,2,4,5]   [5,2,4,3,2,4,5]
[8,4,3,2,4,3,2]   [8,4,3,2,4,3,2]   [6,4,3,2,4,3,4]   [6,4,3,2,4,3,4]
[2,7,2,4,3,2,6]   [2,3,2,4,3,8,4]   [4,3,2,4,3,8,2]   [4,7,2,4,3,4,2]
[3,2,4,3,2,4,5]   [3,2,4,3,2,4,5]   [3,2,4,3,2,4,5]   [3,2,4,3,2,6,3]
[2,4,3,2,4,7,2]   [2,8,3,2,4,3,2]   [2,8,3,2,4,3,2]   [2,4,3,2,4,5,4]
[4,7,6,4,5,2,4]   [4,7,6,4,3,2,6]   [4,7,6,4,3,2,6]   [4,7,6,4,3,2,6]
[3,6,3,7,2,8,5]   [3,6,3,7,4,6,5]   [3,6,3,7,4,6,5]   [3,6,3,7,4,6,5]


[5,2,4,3,2,4,5]   [5,2,4,3,2,4,5]   [5,2,4,3,2,4,5]   [5,2,4,3,4,2,5]
[2,4,3,2,4,3,8]   [8,4,3,2,4,3,2]   [8,4,3,2,4,3,2]   [6,4,3,4,2,3,4]
[4,3,2,4,3,8,2]   [2,7,2,4,5,2,4]   [2,7,2,4,5,2,4]   [4,7,4,2,3,4,2]
[5,2,4,3,2,4,3]   [3,2,4,3,2,4,5]   [3,2,4,3,2,6,3]   [3,4,2,3,2,6,3]
[2,8,3,2,4,3,2]   [2,4,3,2,4,7,2]   [2,4,3,2,4,5,4]   [4,2,3,2,4,5,4]
[6,7,6,4,3,2,4]   [4,7,6,4,3,2,6]   [4,7,6,4,3,2,6]   [2,7,6,4,5,2,6]
[3,6,3,7,4,6,5]   [3,6,3,7,2,8,5]   [3,6,3,7,2,8,5]   [3,6,3,7,2,8,5]


[5,2,4,3,4,2,5]   [5,2,4,3,4,2,5]   [5,2,4,3,4,2,5]
[2,4,3,4,2,3,8]   [6,4,3,4,2,3,4]   [8,4,3,4,2,3,2]
[4,7,4,2,3,4,2]   [4,7,4,2,3,4,2]   [2,7,4,2,5,2,4]
[3,4,2,3,2,6,3]   [3,4,2,3,2,6,3]   [3,4,2,3,2,6,3]
[4,2,3,2,4,7,2]   [4,2,3,2,4,7,2]   [4,2,3,2,4,7,2]
[6,7,6,4,3,2,4]   [2,7,6,4,3,2,8]   [2,7,6,4,3,2,8]
[3,6,3,7,4,6,5]   [3,6,3,7,4,6,5]   [3,6,3,7,2,8,5]
```

2.3. table, automaton

- `table(+Tuples,+Extension [,+Options])`: A korlát segítségével tetszőleges n argumentumú relációt le lehet írni úgy, hogy felsoroljuk a reláció elemeit. `Tuples` n hosszú listák listája, melyben minden elem egy egész szám vagy egész értékű változó. `Extension` n hosszú egész listák listája, azonban egészek helyett állhatnak konstans tartományok is (pl. `(inf..3) \/ {5,8} \/ (12..sup)`). A korlát akkor teljesül, ha `Tuples` minden eleme benne van `Extension`-ben.

- `automaton(+Signature,+SourcesSinks,+Arcs)`: A korlátban egy véges automatát adunk meg, mely meghatározza az elfogadható számsorozatokat. `Signature` egész értékű változók listája. `SourcesSinks` egy lista, melyben `source(Node)` illetve `sink(Node)` alakú kifejezések vannak, meghatározva a véges automata kezdő és végállapotait. `Arcs` élek listája. Egy él `arc(Node1,I,Node2)` alakú, amely azt írja le, hogy az automata `Node1` állapotból `I` egy egész szám érkezése esetén `Node2` állapotba megy át. Az állapotok azonosítói tetszőleges Prolog kifejezések lehetnek. A korlát akkor teljesül, ha `Signature`-ben szereplő bemeneti sorozatot elfogadja a `SourcesSinks` és `Arcs` paraméterek által leírt automata.

Mindkét korlátnak vannak további opcionális argumentumai, melyekről részletek a SICStus kézikönyvben olvashatóak.

3. Feladat. *A "könnyű mint az ABC" feladványban egy NxN-es négyzetes mátrix minden sorában és oszlopában el kell helyezni az ABC első M betűjét pontosan egyszer. Minden sorban és oszlopban így N-M mező marad üresen. Egyes sorok/oszlopok elején/végén meg van adva, hogy az adott irányból nézve melyik betű van az első nem üres mezőben (röviden: melyik betű látható). Példa N = 5, M = 3 esetre (50. sorszámú feladvány):*

```
        C       C
    ----------------
  B |   |   |   |   | C
    |---|---|---|---|---|
  C |   |   |   |   | B
    |---|---|---|---|---|
  B |   |   |   |   | A
    |---|---|---|---|---|
    |   |   |   |   | C
    |---|---|---|---|---|
    |   |   |   |   |
    ----------------
        A   B   C
```

A feladvány egyetlen megoldása:

```
        C       C
    ----------------
  B | B |   |   | A | C | C
    |---|---|---|---|---|
  C |   | C | A | B |   | B
    |---|---|---|---|---|
  B |   | B | C |   | A | A
    |---|---|---|---|---|
    | A |   | B | C |   | C
    |---|---|---|---|---|
    | C | A |   |   | B |
    ----------------
        A   B   C
```

A feladvány ábrázolása Prologban:

`abc(N,M,Bal,Jobb,Felso,Also)`

ahol `N` *és* `M` *a paraméterek,* `Bal, Jobb, Felso, Also` *N hosszú listák, amelyek az adott oldalról látható betűk listája. A betűket az ABC-beli sorszámukkal jelöljük, ahol nincs megadott látvány, ott a 0 szám áll. A fenti példafeladvány ábrázolása:*

`abc(5,3,[2,3,2,0,0],[3,2,1,3,0],[0,3,0,0,3],[0,1,2,3,0])`

Egy egészlista bal látványának az első nem-0 elemét, jobb látványának az utolsó nem-0 elemét hívjuk.

Az ABC feladat megoldásához rendelkezésre áll egy keretprogram (`abc.pl`*). A keretprogram futásához szükség van az ún. látványkorlátok megvalósitására, amelyeket több változatban is megadhatunk. Az egyes változatokat* `latvany_XXX.pl` *nevű állományokban kell elhelyezni. Minden ilyen állomány egy* `XXX` *nevű modult kell definiáljon, amely a látványkorlátok egy vagy több változatának definícióját is tartalmazhatja (de nem szabad ezeket exportálnia).* `XXX` *legyen* `{vezetéknév}_YYY` *alakú, például egy reifikációval elkészített változat kerülhet a* `latvany_zombori_reif.pl` *állományba. A lehetséges látványkorlátok a következőek (a legegyszerűbbtől a legbonyolultabbig):*

- `b_latv(L, B)`*: L bal látványa B.*
- `b_latv_gcc(L, B, M)`*: L bal látványa B, továbbá L-ben az* $1, \ldots, M$ *számok mindegyike pontosan egyszer fordul elő, és a többi elem 0.*
- `bj_latv(L, B, J)`*: L bal látványa B és L jobb látványa J.*
- `bj_latv_gcc(L, B, J, M)`*: L bal látványa B, jobb látványa J és L-ben az* $1, \ldots, M$ *számok mindegyike pontosan egyszer fordul elő, és a többi elem 0.*
- `multi_b_latv(BLs)`*: BLs minden [B/L] elemére fennáll a* `b_latv(L, B)` *predikátum.*
- `multi_b_latv_gcc(BLs, M)`*: BLs minden [B/L] elemére fennáll a* `b_latv_gcc(L, B, M)` *predikátum.*
- `multi_bj_latv(BJLs)`*: BJLs minden [B,J/L] elemére fennáll a* `bj_latv(L, B, J)` *predikátum.*
- `multi_bj_latv_gcc(BJLs, M)`*: BJLs minden [B,J/L] elemére fennáll a* `bj_latv_gcc(L, B, J, M)` *predikátum.*

A B és J látványértékek adott számok, amelyekről feltételezhető, hogy $B > 0, J > 0$*. A keretprogram a bonyolultabb eljárásokat keresi először, ha valamelyik nincs megvalósítva, akkor folytatja a keresést az egyszerűbbekkel. Ha a legegyszerűbb (*`b_latv/2`*) eljárás definiálva van, akkor biztos működőképes a keretprogram.*

A fenti eljárások közüli választást opciókkal is lehet szabályozni. A `{nomulti, nogcc, nobj}` *opciók rendre a nevükben a multi, gcc, bj szavakat tartalmazó eljárások használatát tiltják le. Értelemszerűen több opció is megadható. Ha a fenti három opció mindegyikét megadjuk, akkor kizárólag a* `b_latv/2` *eljárást használja a keretprogram. Persze ugyanez a helyzet, ha csak a* `b_latv/2` *eljárás van definiálva az adott modulban.*

A keretprogram belépési pontja az abc/3 eljárás.

`abc(Feladv, V, Mx)`*: A* `Feladv` *ABC feladvány megoldása a* `V` *variánssal* `Mx`*.*

A `V` *(variáns) argumentum lehet egy atom, ekkor a használandó modul* `XXX` *nevét kell tartalmazza; vagy lehet egy lista, amelynek első eleme a modulnév kell legyen, ezt követi a futást vezérlő opciók listája. A* `{nomulti, nogcc, nobj}` *opciókat a keretprogram veszi figyelembe, a fennmaradó listaelemeket a* `user:latvanykorlat_opciok` *változóba menti a keretprogram, ahonnan a látványkorlátokat megvalósító modul elő tudja venni ezeket.*

Valósítsd meg a fenti látványkorlátokat `table` *és* `automaton` *korlátok segítségével! A megoldások kerüljenek a* `latvany_{vezetéknév}_table.pl` *illetve* `latvany_{vezetéknév}_automaton.pl` *állományokba!*

2.4. geost

- `geost(+Objects,+Shapes [,+Options])`: A korlát segítségével többdimenziós, egymást nem átfedő testek elhelyezkedését lehet korlátozni. `Objects` az elhelyezendő objektumok listája, melyek alakja `object(Oid,Sid,Origin)`, ahol `Oid` egy egész szám, mely egyedileg azonosítja az objektumot, `Sid` egész szám vagy változó, mely az objektum alakját azonosítja, `Origin` egész számok vagy változók listája, mely az objektum kezdőpontját adja meg. Az alakzatok k-dimenziós téglatestekből állnak össze. `Shapes` ilyen k-dimenziós téglatestek listája, melyek alakja `sbox(Sid,Offset,Size)`. `Sid` egy egész szám. Az azonos `Sid`-del rendelkező téglatestek együttesen alkotnak egy alakzatot.`Offset` egy k hosszú egészlista, mely meghatározza, hogy a téglatest kezdőpontja hol van az alakzat kezdőpontjához képest. `Size` szintén egy k-hosszú egészlista, mely a téglatest méretét adja meg az egyes dimenziókban. A korlát akkor teljesül, ha az objektumok egymást nem fedik át a térben.

A korlátot számtalan opcióval lehet meghívni, melyekről részletek a SICStus kézikönyvben olvashatóak.

4. Feladat. *A feladat egy egyszerű tetris játék megírása. Ehhez az alábbi predikátumot kell megvalósítani:*
`tetris(+ObjList,+Width,+Height,?Result):-`
`ObjList obj(ID,Shape)` alakú termek listája, melyek az egymás után következő tetris testeket írják le. Ezeket kell elhelyezni egy `Width` szélességű es `Height` magassagu pályára. `Result` maga a helyes lerakás, egy `Height`∗`Width` méretű mátrix. Egy tetris testet egy egyedi azonosító (`ID`) és egy alakzat (`Shape`) segítségével írunk le. Az alakzat az alábbi értékeket veheti fel:

```
%
%
% 1: **   2:  **  3:       4:  *   5: *    6:   *  7: **
%     **     **      ****     ***     ***     ***     **
%
%
```

A lerakás során az alakzatokat tetszőlegesen el lehet forgatni. Az eredménymátrix egyes mezőiben azon test azonosítója szerepel, mely az adott mezőt kitölti. Az üresen maradt mezők értéke 0.
Például az alábbi feladvány egy megoldása:

```
tetris([obj(1,3),obj(2,7),obj(3,5),obj(4,2)],6,3,Result).

Result = [[ 0 , 4 , 4 , 2 , 2 , 3 ],
          [ 4 , 4 , 0 , 2 , 2 , 3 ],
          [ 1 , 1 , 1 , 1 , 3 , 3 ]].
```

FONTOS: a megoldást mindenki egy vezeteknev.pl állományba rakja bele, amiben megtalálható a vezeteknev nevű modul, mely nem exportál semmit. Tehát például Zombori Zsolt megoldása a zombori.pl állományban legyen, melynek első sora az alábbi:

```
:- module(zombori,[]).
```

2.5. cumulative

- `cumulative(+Tasks [,+Options])`: A korlát segítségével ütemezési feladatokat lehet megoldani. `Tasks` elvégzendő feladatok listája `task(Oi,Di,Ei,Hi,Ti)` alakban, ahol `Oi` a feladat kezdési ideje, `Di` a feladat elvégzéséhez szükséges idő, `Ei` a feladat befejezésének ideje, `Hi` a feladat elvégzéséhez szükséges erőforrások száma, `Ti` pedig a feladat azonosító száma. Mindegyik érték egész szám vagy egész értékű, korlátos tartományú változó. A korlát pontosan akkor teljesül, ha a kezdőidők, időtartamok és végidők által meghatározott üzemezés során az erőforrás igény sose haladja meg a limitet. A limit alapból 1, de ezt az opciólista segítségével át lehet állítani. `Options` opciólista az alábbi kifejezéseket tartalmazhatja:

  1. `limit(N)`: az erőforrás limit minden időpontban N egész szám.
  2. `precedences(PS)`: `Ps` egy lista, melynek elemei T_i-T_j `#=` D_{ij} alakúak, ahol T_i és T_j feladat azonosítók, D_{ij} pedig egész szám vagy egész változó. Íly módon megköthetjük, hogy $O_i - O_j = D_{ij}$.
  3. `global(Boolean)`: `Boolean` $\in$ `{true,false}` és amennyiben be van kapcsolva, akkor egy költségesebb, de jobban szűkítő következtető algoritmus működik.

5. Feladat. *Egy kereskedőnek van néhány rabszolgája. Az a terve, hogy végigjárja a Selyemutat és az út mentén lévő városokkal kereskedik. Az egyes városokban különféle árucikkek állnak rendelkezésre és az is meg van adva, hogy mely városokban lehet őket értékesíteni. Adott továbbá, hogy egy áru szállításához mennyi rabszolga szükséges, valamint, hogy milyen értékes az áru. Ahány városnyival visz arrébb egy adott árut, annyiszorosát kapja meg az értéknek. A kereskedő pontosan egyszer szeretne végigmenni a Selyem-úton és az összes létező árut el szeretné szállítani valameddig. A feladat annak eldöntése, hogy ez lehetséges-e, valamint mennyi a maximális haszon. Ehhez el kell készíteni az alábbi* `karavan/4` *eljárást, a lehető leg hatékonyabb módon:*

`karavan(+N,+Aruk,-Utemezes,-Nyereseg)`

`Aruk` *egy lista, elemei* `aru(ID,Start1-Start2,Stop1-Stop2,Jutalom,Rabszolga)` *alakú termek, ahol* `ID` *egy olyan árut azonosít, melyet a* `Start1` *és* `Start2`*-dik települések között lehet megvenni és a* `Stop1` *és* `Stop2`*-dik települések között lehet eladni.* `N` *az összes rendelkezésre álló rabszolga. Az áru eladásáért járó nyereség* `Jutalom` *szorozva a megtett út hosszával (az út hossza alatt az érintett városok számát értjük). Az áru szállításához* `Rabszolga` *darab rabszolga szükséges.* `Utemezes` `aru(ID,Start,Stop)` *alakú termek listája, mely leírja az egyes áruk tényleges vételi és eladási idejeit.* `Nyereseg` *pedig az egyes árukért járó jutalmak összege, a teljes haszon.*
Valósítsd meg a `karavan/4` *predikátumot a* `cumulative` *korlát felhasználásával! A megoldást helyezd egy* `vezeteknev.pl` *állományba, amiben megtalálható a* `vezeteknev` *nevű modul, mely nem exportál semmit. Tehát például Zombori Zsolt megoldása a* `zombori.pl` *állományban legyen, melynek első sora az alábbi:*

```
:- module(zombori,[]).
```

2.6. cumulatives

- `cumulatives(+Tasks,+Machines [,+Options])`: A korlát segítségével olyan ütemezési feladatokat lehet megoldani, ahol több, különböző erőforrással rendelkező gép között kell szétosztani feladatokat. `Tasks` elvégzendő feladatok listája `task(Oi,Di,Ei,Hi,Mi)` alakban, ahol `Oi` a feladat kezdési ideje, `Di` a feladat elvégzéséhez szükséges idő, `Ei` a feladat befejezésének ideje, `Hi` a feladat elvégzéséhez szükséges erőforrások száma, `Mi` pedig a feladatot elvégző gép azonosító száma. Mindegyik érték egész szám vagy egész értékű, korlátos tartományú változó. `Machines` a rendelkezésre álló gépek listája, melyek alakja `machine(Mj,Lj)`, ahol `Mj` a gép azonosítója, `Lj` pedig a gép maximális kapacitása. Mindkét érték egész szám. A kezdő idők, időtartamok, végidők és a gépazonosítók együttesen meghatározzák az egyes gépek terhelését. A korlát pontosan akkor teljesül, ha a terhelés semelyik időpillanatban se kerül a kapacitás alá illetve fölé (ezt az opció listában lehet beállítani). `Options` opciólista az alábbi kifejezéseket tartalmazhatja:

  1. `bound(B)`: Ha `B = lower`, akkor a kapacitások alsó korlátként értelmezendők (ez az alapértelmezett). Ha Ha `B = upper`, akkor a kapcitások felső korlátként értelmezendők.
  2. `prune(P)`: optimalizáció, lásd a manualt.
  3. `generalization(Boolean)`: komplexebb következtető algoritmus, lásd a manualt.
  4. `task_intervals(Boolean)`: komplexebb következtető algoritmus, lásd a manualt.

6. Feladat. *A Selyem-úton néhány karaván vonul végig. Az út mentén lakó városoknak áruik vannak, melyekkel kereskedni szeretnének. Minden árut el szeretnének adni, csak azt kell eldönteniük, hogy melyik karavánra bízzák rá az áru szállítását. Előfordulhat továbbá, hogy egy árut több városban is fel lehet venni, illetve több városban is el lehet adni. Meg van adva az egyes áruk elszállításához szükséges rabszolgaszám, valamint az is, hogy a karavánok mennyi rabszolgával rendelkeznek.*

A városoknak tehát karaván kell választaniuk minden egyes áruhoz. Azonban arra is figyelniük kell, hogy a karavánok elég irigyek egymásra és elég harciasak. Ha egy adott időpillanatban egy karaván azt látja, hogy egy másik karaván legalább 3-mal több árut szállít, akkor mérgében megtámadja és kirabolja a másik karaván. Ilyen nem fordulhat elő, ez árt az üzletnek hosszú távon. Tehát valamennyire muszáj egyenletesen kiosztani a megrendeléseket a karavánok között.
Cél az alábbi `selyemut/4` eljárás minél hatékonyabb megvalósítása:

`selyemut(+AruList,+KaravanList,-Utemezes)`
`AruList aru(ID,Start1-Start2,Stop1-Stop2,Rabszolga)` alakú termek listája, ahol `ID` egy olyan árut azonosít, melyet a `Start1` és `Start2`-dik települések között lehet megvenni és a `Stop1` és `Stop2`-dik települések között lehet eladni. Az áru szállításához `Rabszolga` darab rabszolga szükséges.

`KaravanList karavan(ID,Kapacitas)` alakú termek listája, ahol `ID` egy olyan karaván azonosít, melynek `Kapacitas` darab rabszolgája van.

`Utemezes aru(ID,Start,Stop,KaravanID)` alakú termek listája, mely megadja az egyes áruk tényleges vételi és eladási idejeit, valamint a szállító karaván azonosítóját.
Egy egyszerű példa, melynek három megoldása van:

```
_AruList = [aru(1,1-1,4-4,1),
            aru(2,2-2,5-5,1),
            aru(3,4-4,7-7,1),
            aru(4,3-3,7-7,1)
           ],
_KaravanList = [karavan(1,3),karavan(2,1)],
zombori:selyemut(_AruList,_KaravanList,Utemezes).
Utemezes = [aru(1,1,4,1),aru(2,2,5,1),aru(3,4,7,1),aru(4,3,7,2)] ? ;
Utemezes = [aru(1,1,4,1),aru(2,2,5,2),aru(3,4,7,1),aru(4,3,7,1)] ? ;
Utemezes = [aru(1,1,4,2),aru(2,2,5,1),aru(3,4,7,2),aru(4,3,7,1)] ? ;
no
```

2.7. Fekete Lyuk passziánsz játék

A Fekete Lyuk egy egyszemélyes kártyajáték, melyet az 52 lapos francia kártyával lehet játszani. Az asztal közepén van a *„fekete lyuk”*, ide a játék elején a pikk ászt kell helyezni. A maradkék 51 lapot össze kell keverni és hármas csoportokba osztani. A csoportokat felfordítva el kell helyezni az asztalon a fekete lyuk körül. A játékos innentől kezdve egyesével rak kártyákat a fekete lyukba, két szabály betartásával. Egyrész minden csoportból csak a legfelső lapot lehet rakni (viszont a többit is meg szabad nézni), másrészt a rakott kártya vagy eggyel kisebb, vagy eggyel nagyobb kell legyen, mint a fekete lyuk tetején található kártya. Az ásznál eggyel kisebb lap a király és az eggyel nagyobb a kettes. A játék akkor ér véget, ha már nem lehet lapot tenni a fekete lyukra. A játékos akkor nyer, ha az összes lap elfogyott.

7. Feladat. *A következőkben a fekete lyuk kártyajáték általánosításával foglalkozunk. Paraméterként megadható, hogy hány féle kártyaszín van (nem feltétlen 4), valamint hogy egy színen belül hány kártya van (nem feltétlen 13). A szétrakott paklikban pedig nem feltétlenül 3 lap van, akár paklinként változhat is. Eltérés továbbá, hogy kezdetben nem a pikk ász, hanem egy olyan színű ász van a fekete lyukban, ami a paklikban nem fordul elő. Tehát ha például 5 szín van, akkor a legelső lap egy hatodik színhez tartozó egyetlen (ász) kártya.*

További megkötés, hogy a színek nagyjából egyenletesen következnek. Pontosabban, a játék bármely pillanatában és bármely A, B színekre teljesül, hogy a fekete lyukban levő A és B színű kártyák darabszáma közti különbség legfeljebb 2.

Az $1 \dots k$ színek különböző értékűek és egy megoldás értékét növeli, ha az értékesebb színeket minél később használjuk fel. Ha a paklira n-diknek (nem számítva a kezdőlapot) egy k színű kártya kerül, akkor azért $n * k$ jutalompont jár. A cél egy maximális összpontszámú megoldás előállítása.

A feladat egy `feketelyuk/5` *eljárás elkészítése, mely talál egy nyertes kártyasorrendet egy adott lapkiosztáshoz, maximális összértékkel.*

`feketelyuk(+N,+Szin,+Elrendezes,-Sorrend,-Osszertek)`

A pakliban `Szin` *féle szín van és színenként* `N` *darab kártya.* `Elrendezes` *egy lista, mely megadja a kezdeti lapkiosztást. A lista elemei számlisták és minden szám egy kártyalapot kódol. A lista k-dik eleme felülről a k-dik kártya az adott csoportban.* `Sorrend` *egy számlista, mely leírja, hogy a lapok milyen sorban kerülnek a fekete lyukba. A kezdeti lap (mely minden mástól eltérő színű) nincs benne a listában.*

A lapok kódolásánál azonos színen belül a kódszámok ugyanúgy növekednek mint a lapok, ásztól az N-dik lapig. A különböző színek egymás utáni tartományokban helyezkednek el az alábbiak szerint:

- *1. szín:* $1 \dots N$
- *2. szín:* $N + 1 \dots 2 * N$
- $\dots$
- *k. szín:* $(k - 1) * N + 1 \dots k * N$

Tehát az i-dik szín j-dik kártyáját a $(i - 1) * N + j$ szám kódolja.

A. Függelék

Az aranyásók társasjáték leírása

Aranyásók

Tervezte: Frederic Moyersoen

Játékosok száma: 3-10 játékos részére. ajánlott: 8 éves kortól. játékidő: 30 perc


44 útkártya


27 akciókártya


28 aranykártya


7 aranyásókártya


4 szabotőrkártya

A játék

A játékosok lehetnek szorgos aranyásók akik csákányokkal lemennek a hegyek tárnáiba arany után kutatni, vagy pedig szabotőrökként játszanak, megakadályozva és hátráltatva az aranyásókat. A két csapat tagjainak támogatniuk kellene egymást akkor is, ha csak sejtik melyik játékos melyik csapatba tartozik. Ha az aranyásóknak sikerül eljutniuk az aranyhoz, akkor arany a jutalmuk, a szabotőrök pedig üres kézzel mehetnek haza. Ha ez nem sikerül nekik, akkor a szabotőrök győznek, övék a jutalom arany, az aranyásók pedig nem kapnak semmit. Csak akkor, derül ki, hogy melyik játékos melyik csapathoz tartozik, amikor az arany elosztására kerül sor. A játékot az a játékos nyeri, akinek 3 fordulóban a legtöbb aranyat sikerült összegyűjtenie.

A játék előkészítése

A játékosok szétválogatják az útkártyákat, az akciókártyákat, az aranykártyákat, és külön válogatják azokat a kártyákat is, amiken a törpék vannak.

A játékosok számának megfelelő darabszámú aranyásó és szabotőr kártya kerül a játékba. A maradék törpés kártyát a játékosok félreteszik.

- 3 játékos esetén 1 szabotőr és 3 aranyásó.
- 4 játékos esetén 1 szabotőr és 4 aranyásó.
- 5 játékos esetén 2 szabotőr és 4 aranyásó.
- 6 játékos esetén 2 szabotőr és 5 aranyásó.
- 7 játékos esetén 3 szabotőr és 5 aranyásó.
- 8 játékos esetén 3 szabotőr és 6 aranyásó.
- 9 játékos esetén 3 szabotőr és 7 aranyásó.
- 10 játékos esetén az összes kártya (4 szabotőr és 7 aranyásó) kerül a játékba.

A játékosok választanak egy osztó játékost, aki a „törpés" kártyákat megkeveri és lefordítva elosztja a játékosok között. A játékosok megnézik a kapott kártyát, majd lefordítva maguk elé teszik. A játékosok nem mondhatják el egymásnak, hogy milyen szerep jutott nekik ebben a fordulóban. Egy kártya az osztónál marad. Ezt az osztó játékos lefordítva félreteszi. A játékosok csak a forduló végén fordíthatják meg kártyáikat.

A 44 útkártya között van egy startkártya (ennek mindkét oldalán egy létra látható), és három célkártya. Az egyik célkártyán van az arany, a másik két célkártyán csak egy-egy kő látható. Az osztó játékos megkeveri a célkártyákat, majd lefordítva az asztalra teszi azokat a startkártyával együtt, a képen látható módon. A játék folyamán a startkártyától kiindulva egy labirintusjárat alakul ki. A labirintusjárat kialakításakor az útkártyák kilóghatnak az ábrán látható határokból.

célkártya

Startkártya *7 kártyahely* *1 kártyahely*

célkártya

Az osztó játékos összekeveri a maradék 40 útkártyát és minden akciókártyát, majd lefordítva elosztja a játékosok között.

1 kártyahely

- 3-5 játékos esetén minden játékos 6 kártyát kap a kezébe.
- 6-7 játékos esetén minden játékos 5 kártyát kap a kezébe.
- 8-10 játékos esetén minden játékos 4 kártyát kap a kezébe.

célkártya

Az osztó játékos a megmaradt kártyákból egy lefordított paklit képez az asztalon, a célkártyák mögött, ez lesz a húzó-pakli.

Az osztó játékos megkeveri az aranykártyákat, majd egy lefordított paklit képez belőlük a megmaradt törpés kártyapakli mellett.

A játékot a legfiatalabb játékos kezdi, a játék további menete az óramutató járásával megegyező irányú.

húzó-pakli

A játék menete

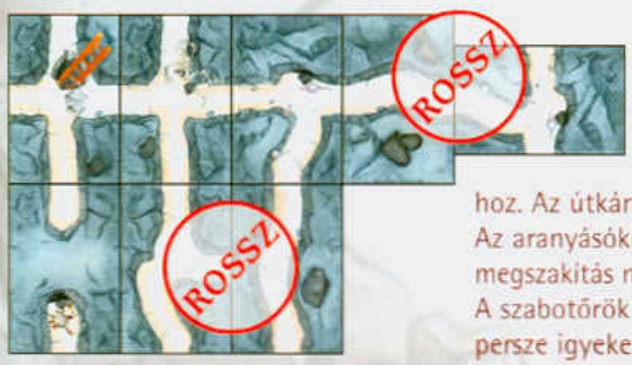
Az éppen soron lévő játékosnak ki kell játszania egy lapot a kezéből, amit a következők szerint tehet:

- A játékos vagy egy útkártyát illeszt a labirintushoz,
- vagy egy akciókártyát tesz egy játékostársa elé,
- vagy passzol. Ebben az esetben egy kártyát dob a lerakó-paklira.

Ezután a játékos új lapot húz a húzó-pakliból, amit a kezébe vesz. Ezzel a játékos lépése véget ér, a játékot a következő játékos folytatja.

Figyelem: Abban az esetben, ha elfogynak a húzó-pakliból a lapok, a játékosok nem tudnak új lapot húzni, már csak a kezeikben tartott lapokat tudják kijátszani.

Útkártya lerakása



Az útkártyákból áll össze az út a startkártyától a célkártyáig. Útkártyát csak úgy tehet le a játékos, hogy annak illeszkednie kell legalább egy, már korábban lerakott útkártyához. Az útkártyán lévő járatoknak és a már meglévő járatoknak illeszkedniük kell egymáshoz. Az útkártya nem rakható le oldalt forgatva. (lásd az ábrán).

Az aranyásóknak az a céljuk, hogy kialakítsanak olyan járatot, amely megszakítás nélkül elvezet a startkártyától a célkártyák valamelyikéig. A szabotőrök ezt megpróbálják megakadályozni. A szabotőrök ezt persze igyekeznek nem feltűnően csinálni, nehogy lelepleződjenek.

Akciókártya kijátszása

Az akciókártyát a játékos felfordítva teszi le maga elé. A játékos egy akciókártyát kijátszhat maga előtt, vagy valamelyik játékostársa elé is lerakhatja. Akciókárty a játékosokat segítheti, vagy akadályozhatja is. A játékos egy akciókártyával eltávolíthat lapot a labirintusból, vagy éppen információt szerezhet valamelyik célkártyáról.



Ha egy játékos lerak egy játékostársa elé egy akciókártyát (kijátssza ellene), az a játékostárs nem rakhat le útkártyát addig, amíg az akciókártya előtte fekszik. Más kártyát természetesen kijátszhat. Egy játékos előtt maximum három akciókártya lehet, minden fajtából egy. Az a játékos tehet le útkártyát, aki előtt lépése megkezdésekor nincs akciókártya.



A középső sorban látható kártyákkal tudja a játékos eltávolítani az előtt lévő akciókártyát. A játékos rátehet ilyen lapot egy saját maga előtt fekvő akciókártyára, vagy egyik játékostársa előtt fekvő akciókártyára. Miután a játékos a kártyát kijátszotta, mindkét kártya a lerakó-paklira kerül. Ha például egy törött csille akciókártya van a játékos előtt, akkor azt egy jó csillét ábrázoló kártyával eltüntetheti.



Az alsó sorban látható akciókártyákon két-két tárgy is látható. Abban az esetben, ha a játékos egy ilyen kártyát játszik ki, akkor a kártyán lévő két tárgy bármelyikét megjavíthatja - de csak az egyiket, mindkettőt nem!

Figyelem: Minden javító kártyát csak olyan játékos ellen lehet kijátszani, aki előtt a szükséges akadályozó kártya van.

Akkor, amikor a játékos „kőomlás" kártyát játszik ki, egy tetszőleges útkártyát (kivéve a start és a célkártya) kivehet a labirintusból. A játékos a labirintusból kivett lapot a kijátszott kőomlás akciókártyával együtt a lerakó-paklira dobja. Ennek az akciókártyának a segítségével a szabotőr megszakíthatja a célkártya felé haladó labirintus járatot. Az aranyásó pedig kivehet egy zsákutcát a labirintusból. A kőomlás akciókártyával kivett útkártya helyét a játékosok később egy megfelelő útkártya elhelyezésével betölthetik.

Akkor, amikor a játékos „térkép" kártyát játszik ki, megnézhet egyet a három célkártya közül. A játékos a megnézett célkártyát visszarakja a helyére és senkinek nem mondja el, amit rajta látott, de ő már tudni fogja, hogy érdemes-e efelé a célkártya felé járatot építeni. Mert a három célkártyából csak egyik célkártyán van aranyrög. Végül a játékos akciókártyáját a lerakó-paklira teszi.

Passzolás

Abban az esetben, ha egy játékos nem tud vagy nem akar kártyát kijátszani, akkor passzol. A játékos kezében tartott lapok közül egyet lefordítva a lerakó-pakli tetejére rak anélkül, hogy megmutatná játékostársainak azt. A lerakó-pakliban lévő lapok egy része lefordítva, egy része, pedig felfordítva van. Egy forduló végénél előfordulhat az is, hogy egy játékosnak nincs már a kezében több kártya, ilyenkor a játékosnak passzolnia kell.

Célkártya

Egy forduló vége

Akkor, amikor egy játékos útkártyájával elér egy célkártyát, és az így lezárult járat megszakítás nélkül a startkártyától kiindulva a célkártyáig vezet, a játékos felfordíthatja azt a célkártyát.

Célkártya aranyröggel

- Ha a célkártyán aranyrög látható, a forduló véget is ért.
- Ha a célkártyán kőszikla van, a forduló tovább folytatódik. A felfordított célkártyát úgy kell elhelyezni, hogy annak járata csatlakozzon az utoljára lerakott útkártya járatához.

Figyelem: Ritkán előfordul, hogy valamelyik útkártya járata nem illeszthető a már kialakított járathoz. Ez a célkártyáknál kivételesen megengedett.

A forduló abban az esetben is véget ér, ha már egy játékosnak sincsen kijátszható kártya a kezében.

Ebben az esetben a játékosok felfordítják a törpés kártyáikat. Na, ki volt aranyász, és ki szabotőr?

Az aranykártyák elosztása

Az aranyászok győztek, ha a járat megszakítás nélkül a startkártyától kiindulva az aranyrögös célkártyáig vezet. Ebben az esetben a célkártyát elérő játékos annyi aranykártyát vesz el lefordítva az arany-pakliból, amennyi a játékosok száma. Ha például a játékosok öten vannak, akkor öt aranykártyát kell elvenni.

Az a játékos, aki útkártyájával elérte az aranyrögös célkártyát, a kezébe vett aranykártyákból választ magának egyet. A megmaradt kártyákat továbbadja a következő aranyászának (nem szabotőrnek), és nem az óramutató járása szerint, hanem azzal ellentétes irányba. Ez a játékos is választ egy aranykártyát, majd továbbadja a megmaradt lapokat a következő aranyászának. Ez addig folytatódik, amíg az aranykártyák el nem fogynak. Könnyen előfordulhat, hogy az egyik aranyász több aranykártyát kap, mint a másik.

Figyelem: Abban az esetben, ha a játékban 10 játékos vesz részt és az aranyászok győznek, akkor csak kilenc aranykártyán kell osztozkodniuk. Akkor, ha a játékban 3 vagy 4 játékos vesz részt és nincs szabotőr és a játékosok nem érték el az aranyrögös célkártyát, egyik játékos sem kap aranykártyát.

A szabotőrök győztek, ha az aranyászok nem érték el labirintusukkal az aranyrögös célkártyát. Ha a fordulóban csak egy szabotőr van, ő 4 arany értékben kap aranykártyát. Két vagy három szabotőr esetén mindegyikük 3 arany értékben kap aranykártyát, négy szabotőr esetén pedig mindegyikük 2 arany értékben.

A játékosok a játék végéig titokban tartják, hogy mennyi aranyuk van.

Új forduló

Az aranykártyák kiosztása után új forduló veszi kezdetét. Az osztó játékos visszateszi a helyére a start- és a megkevert célkártyákat. Ezek után az összes törpés kártyát meg kell keverni, majd minden játékosnak osztani egyet. Az út- és az akciókártyákat össze kell keverni, és ahogy a játék elején történt, ismét annyit kell osztani minden játékosnak. A maradék lapokból új felhúzó-paklit kell képezni.

A játék további menete az első forduló szabályaival megegyezően halad tovább. Azzal a kivétellel, hogy a játékosok maguknál tarthatják az addig megszerzett aranykártyáikat. A megmaradt aranykártyákat, a megmarat törpés kártyák mellé kell megint tenni.

A játékot az a játékos kezdi, aki a bal oldalán ül az előző forduló utolsó lapját kijátszó játékosnak.

A játék vége

A játék a harmadik forduló után végetér. Ekkor a játékosok megszámolják, hogy mennyi aranyat gyűjtöttek. A játékot az a játékos nyeri, akinek a legtöbb aranya van. Egyenlőség esetén minden érintett játékos nyert.

Importálja és forgalmazza:
PIATNIK BUDAPEST KFT.
1034 Budapest, Bécsi út 100.Tel.: 388-4122
email: piatnik@piatnik.hu
Származási hely: EU

Piatnik