DIPLOMATERV

Kovács Dániel László

2003.

Intelligens Ágensek Evolúciója

(Evolution of Intelligent Agents)

Készítette: ***Kovács Dániel László***

Budapest Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai kar

Műszaki Informatika szak

V. évfolyam

Konzulens: ***Dr. Dobrowiecki Tadeusz***

Budapest Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

2003. Budapest.

Diplomaterv feladatkiírás

A számítástechnika folyamatos térhódításával felmerül az igény olyan „intelligens eszközök” iránt, melyek képesek megkönnyíteni a mindennapjaink során felmerülő problémák azon részét, ahol az ember szerepe – legalább részint – kiváltható. Az ilyen típusú eszközöknek rendelkeznie kell egyfajta „adaptív-feladatmegoldó” készséggel, amiben központi szerepet játszik a tervkészítés. Máig sem megoldott feladat az „optimális” terv elkészítésének – apriori ismeretek nélkül történő – automatizálása.

A diplomaterv munka célja olyan rendszer elvének kidolgozása, illetve – az elvet szemléltető – olyan szoftver implementálása, amely alkalmas az „optimális” tervkészítés folyamatának automatizálására úgy, hogy a tervkészítés módszerei (sémái) apriori nem adottak, azaz a rendszer maga találja meg őket.

A diplomaterv során megoldandó feladatok:

- Feladat informális megfogalmazása, majd adott tématerületre való leszűkítése.
- Konkrét témakör irodalmazása. A témakörön belüli újabb kutatási eredmények megismerése.
- Formális problémafelvetés. Megoldási javaslat megfogalmazása. A megoldás elvi megalapozása.
- Javasolt megoldás empirikus vizsgálati lehetőségeinek megteremtése. A lehetőségek vizsgálata.
- Empirikus vizsgálatokra alkalmas teszt-környezet specifikációja és implementációja.
- Tapasztalatok összefoglalása és értelmezése. Tesztek értékelése.
- További kutatási lehetőségek vizsgálata.

Nyilatkozat

Alulírott, *Kovács Dániel László*, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva, más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

.....

Kovács Dániel László

Abstract

Evolution of intelligent agents, a new approach to automatic planning, is presented as a new technique for evolving systems capable of generating “optimal” plans without any prior knowledge of the environment, or any method (i.e. schemata) concerning plan design. The first half of the paper gives a detailed overview of the theories linked to this subject, namely: agents, planning and evolutional computation. The second half discusses the new idea presented.

Agents are treated from two points of view: the first gives an abstract definition of agent architectures by trying to familiarize their fundamental ideas, and by considering their purpose and designation. The second considers a possible concretization of the architectures shown in the previous section. Abstract architectures, such as reflexive agents, agents with an internal state, goal- and utility based agents are presented. Concrete architectures, such as logic based, reactive, belief-desire-intention and layered architectures are outlined. This is followed by a comprehensive discussion of planning. After introducing the main concepts, i.e. the notion of plan and planning, problems and solutions, the paper gives a detailed description of the most popular planning methods. Partial order planning, hierarchical task network decomposition and conditional planning is discussed. Consequently the integration of these methods with agents follows. Planning agents, such as simple, conditional, replanning and situated planning agents are next presented. The paper then proceeds with a short summary of evolutional computation based on genetic algorithms. The classical principle is shown to be extended by the latest developments in the field, such as genetic programming and gene expression programming. This is followed by an accurate analysis of the problem stated in the master’s thesis, i.e. generally “optimal” planning. Results are shown concerning the complexity of the problem. The first half of the paper then concludes with a summary of the hitherto used planning methods and their respective properties.

The second half of the paper describes the solution proposed to the problem mentioned, starting with by setting forth the idea of it, i.e. a multi-layered system-architecture. This system is then viewed as an evolutionary environment of some specially situated planning agents. The concept extends the classical model of system-environment interaction in a way by making it more “natural”. The paper then proceeds with a detailed elaboration of the idea, first converting it to an abstract agent-architecture, then to a realizable, concrete architecture. Later on issues of implementing the system are discussed in detail with the help of some very useful hints gathered from the examination of a possible (currently Prolog-based) implementation. Analyzing the difficulties and possibilities of testing the system follows this. After finding a compromise for realizing a thorough system measurement, a detailed description of the testing process is devised: the power of the new approach is demonstrated by testing it on several micro-world problems, by comparing it to other planning methods, and finally, by measuring its performance against human opponents. Test results are then evaluated, and conclusions are drawn concerning the effectiveness of the concept.

Tartalmi kivonat

Az „Intelligens Ágensek Evolúciója” az automatizált tervkészítés egy olyan új megközelítése, mely lehetővé teszi olyan rendszerek létrehozását, melyek már a környezet, vagy a cselekvések modelljeinek (sémáinak) előzetes ismerete nélkül is képesek az „optimális” terv – tetszőleges környezetben történő – előállítására. Míg a dolgozat első fele az elképzeléssel kapcsolatos főbb alapfogalmakat vezeti be, nevezetesen: az ágensek, a tervkészítés és az evolúciós számítási modellek fogalmát, addig a második fele már magával az elképzeléssel foglalkozik.

Az ágenseket két szempontból vizsgáljuk: egyrészt az absztrakt ágens-architektúrák, másrészt a konkrét ágens-architektúrák szemszögéből. Az absztrakt ágensek közül a reflexív, a benső állapottal rendelkező, a cél- és a haszon-orientált architektúrákat különböztetjük meg, míg a konkrét ágensek közül a logikai-alapú, a reaktív, a hiedelem-vágy-szándék alapú és a rétegelt architektúrák elve kerül bemutatásra. Ezt követi a tervkészítés részletes elemzése, amely az alapfogalmak (terv, tervezés, probléma, megoldás, stb.) bevezetésén túl a legelterjedtebb tervkészítési algoritmusok működésébe enged betekintést. Említésre kerül a részben rendezett tervkészítés, a hierarchikus dekompozíció és feltételes tervkészítés elve. A dolgozat további része e módszerek ágensekkel történő integrációját: a tervkészítő ágenseket mutatja be. Ezek közül az egyszerű, a feltételes, az újratervező és a beágyazott tervkészítő ágens elve kerül ismertetésre. Az evolúciós számítási modellek bemutatása csak ezután következik. Ezen belül csak a genetikus algoritmusokon alapuló módszerek alapjaira, és legújabb fejleményeire: a genetikus programozásra és a génkifejező programozásra térünk ki részletesebben. Ezt követi a tervkészítés – diplomaterv feladatkiírásban megadott – problémájának megfogalmazása, formalizálása, illetve a probléma bonyolultságának vizsgálata. A dolgozat első felét a korábbi tervkészítők felsorolása zárja.

A dolgozat második fele foglalkozik a fenti probléma megoldását jelentő központi elv kidolgozásával, mely során egy olyan rendszert mutatunk be, amelyet speciálisan beágyazott tervkészítő ágensek evolúciós környezetének tekintve, a klasszikus rendszer-környezet interakció egy „természetes” kiterjesztéséhez jutunk. Az ötletet a kidolgozás során először absztrakt, majd pedig – egy már megvalósítható – konkrét ágens-architektúrává alakítjuk. A rendszer megtervezését követően a megvalósítás leírása következik, mely egy lehetséges (jelenleg Prolog-alapú) implementáció alapján mutatja be az elv megvalósíthatóságának lehetőségeit. Ezt követi a rendszer tesztelésével kapcsolatos lehetőségek és nehézségek vizsgálata. A megfelelő kompromisszum megtalálását követően a tesztelés folyamatának részletes bemutatása következik. A megközelítés létjogosultságát körültekintő mérésekkel igazoljuk, amik során a rendszert kisebb, ámde kellően reprezentatív mikro-világokon, más hasonló módszerek képességeivel összevetve, végül pedig emberekkel versenyeztetve teszteljük. A teszt-eredményeket értékeljük, majd összefoglalva tapasztalatainkat, kitekintést adunk a jövőbeni kutatási irányokra nézve.

Tartalom

TARTALOM **1**

1. BEVEZETÉS **9**

2. ÁGENSEK – TERVEK – EVOLÚCIÓ **10**

2.1. ÁGENSEK 10

2.1.1. Absztrakt ágens-architektúrák *12*

2.1.1.1. Egyszerű reflex-szerű ágens 13

2.1.1.2. Állapottal rendelkező reflex-szerű ágens 14

2.1.1.3. Cél-orientált ágens 15

2.1.1.4. Haszon-orientált ágens 16

2.1.2. Konkrét ágens-architektúrák *17*

2.1.2.1. Logikai-alapú architektúrák 17

2.1.2.2. Reaktív architektúrák 18

2.1.2.3. Hiedelem-Vágy-Szándék (Belief-Desire-Intention) architektúrák 21

2.1.2.4. Rétegelt architektúrák 24

2.2. TERVKÉSZÍTÉS 25

2.2.1. Probléma és Terv *26*

2.2.2. Reprezentációk és Algoritmusok *27*

2.2.2.1. RRT 30

2.2.2.2. RRT-DUNF 33

2.2.2.3. HD-RRT 34

2.2.2.4. FRRT 36

2.2.3. Tervkészítő Ágensek *38*

2.2.3.1. Egyszerű Tervkészítő Ágens 39

2.2.3.2. Feltételes Tervkészítő Ágens 39

2.2.3.3. Egyszerű Újratervező Ágens 39

2.2.3.4. Beágyazott Tervkészítő Ágens 40

2.3. EVOLÚCIÓS SZÁMÍTÁSI MODELLEK 41

2.3.1. Genetikus Algoritmusok (GA) *41*

2.3.2. Genetikus Programozás (GP) *43*

2.3.3. Génkifejező Programozás (GEP) *45*

3. TERVKÉSZÍTÉS PROBLÉMÁJA **47**

3.1. AZ OPTIMÁLIS TERV FOGALMA 48

3.2. A FELADAT NEHÉZSÉGE 50

4. TERVKÉSZÍTŐ MÓDSZEREK 54

4.1. KORÁBBI MEGKÖZELÍTÉSEK 54

4.2. MEGKÖZELÍTÉSEK FAJTÁI 59

5. MEGOLDÁSI JAVASLAT: EVOLÚCIÓS ÁGENS 60

5.1. AZ ALAPÖTLET 60

5.2. MEGOLDÁS KIDOLGOZÁSA 61

5.2.1. Evolúciós ágensek absztrakt architektúrája 62

5.2.2. Evolúciós ágensek konkrét architektúrája 63

5.3. LEHETSÉGES MEGVALÓSÍTÁS 70

5.3.1. Implementáció tervezése 70

5.3.2. Implementáció megvalósítása 78

6. ELEMZÉS 86

6.1 A TESZTELÉS NEHÉZSÉGEI 86

6.2 A TESZTELÉS MENETE 89

6.2.1. Összevetés az eddigi módszerekkel 89

6.2.2. Mikro-világ tesztek 91

6.2.2.1. Asztal-világ 91

6.2.2.2. Wumpus világ 94

6.3 ÉRTÉKELÉS ÉS KITEKINTÉS 98

7. ÖSSZEFOGLALÁS 99

8. KÖSZÖNETNYÍLVÁNÍTÁS 100

9. IRODALOMJEGYZÉK 101

10. FÜGGELÉK 105

10.1. TERVKÉSZÍTÉSI MÓDSZEREK FELSOROLÁSA 105

10.2. SUSSMAN ANOMÁLIA 106

10.3. ÁGENS-ÁGENS TESZTEK A WUMPUS-VILÁGON 107

10.4. EMBER-ÁGENS TESZTEK A WUMPUS-VILÁGON 108

1. Bevezetés

Az intelligens rendszerek alapfeladata, hogy képesek legyenek önálló, autonóm módon történő tervkészítésre (Russell és Norvig, 1995). Ennek egyik kulcsfontosságú motívuma az, amikor a tervek kialakulnak. Eddig csak olyan tervkészítő algoritmusok születtek, melyek a terv kialakításának sémáit készen kapják, azaz vagy a terv kialakításának módja, vagy a tervben szereplő cselekvések és egyéb alkotóelemek választéka, vagy a környezet, illetve egyéb tényezők modellje **adott** az algoritmus számára. Ez igencsak leszűkíti a megoldható feladatok körét, hiszen a valóban „intelligens" rendszernek a sémákat is magának kell kialakítania ahhoz, hogy képes legyen olyan környezetekben is boldogulni, melyekről a rendszer tervezőjének is csak hiányos ismeretek álltak a rendelkezésére. A valós környezetek tipikusan ilyenek. Ekkor a sémák apriori megadása nem lehetséges, s így a rendszer helyes (előirt) működése sem garantálható általános esetben. A sémák automatikus megalkotása – bár voltak rá kísérletek – eddig még nem eredményezett olyan stabil tervkészítő rendszert, amely képes lett volna általános esetben garantáltan megbízható, más szóval „optimális" tervek előállítására. Dolgozatom ennek a feladatnak a megoldásával foglalkozik. Fő önálló eredményeim: Egy heurisztikus rendszer-architektúra a feladat elvi megoldására, valamint az architektúra egy konkrét (ám részleges) megvalósítása.

A dolgozat felépítése a következő: először bevezetem a megoldás értelmezéséhez szükséges alapfogalmakat, bizonyos fogalmak ismeretét adottnak feltételezve. Ezt követően ismertetem a megoldás alapelvét, majd pedig rátérek az implementációs részleteinek leírására. A megoldás leírását követően a tesztelés kerül terítékre. Először a nehézségeket tárgyaljuk, majd vázoljuk a tesztelés menetét is. A tesztelés során született eredményeket értelmezzük, és röviden értékeljük is.

2. Ágensek – Tervek – Evolúció

Ez a fejezet arra szolgál, hogy megalapozza a dolgozat központi tárgyát képező új elgondolás elméleti hátterét, szemléletmódot adva a későbbiek mélyebb megértéséhez. Az áttekintés során vázoljuk a központi kérdéskört érintő elméleti megfontolásokat és a hozzájuk tartozó tudományos hátteret. Bevezetésre kerülnek az „ágensekkel” és a „tervkészítéssel” kapcsolatos főbb alapfogalmak. A fejezetet az „evolúció” számítógépes modelljeinek tárgyalása zárja.

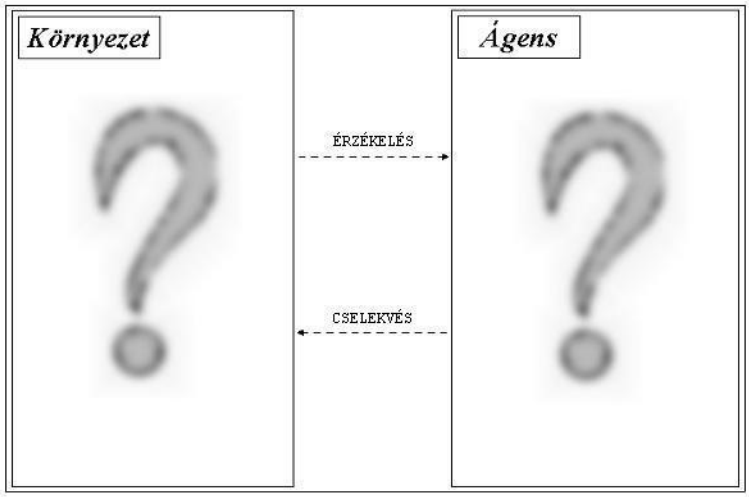
2.1. ÁGENSEK

A számítógépek önmagukban nem jeleskednek az önálló döntéshozatalban: minden véghezvitt utasítást előre le kell programozni, számításba kell venni. A szokványos alkalmazások körében (pl. gépi könyvelés, stb) ez elegendőnek szokott bizonyulni. Ámde újabban jelentős igény mutatkozik olyan rendszerek iránt, amelyek *maguk képesek eldönteni*, hogy mit kell tenniük ahhoz, hogy teljesítsék a rájuk szabott feladatot. Az ilyen rendszereket nevezzük *„ágensnek”*.

Bár az *„ágens”* kifejezésnek többféle definíciója is létezik a tudományos köztudatban, jelentését illetően mégsem alakult ki általános konszenzus. Szakterületről szakterületre változik az *„ágensekkel”* kapcsolatos jellemzők fontossága. Bizonyos alkalmazásokban elengedhetetlen, hogy az *„ágens”* képes legyen *„tanulni”*, mígnem másokban szó szerint kerülendő. Talán csak az *„autonómia”*, mint központi megfontolás, tekinthető az *„ágensek”* általánosan elfogadott, egyetlen közös jellemzőjének. Mivel azonban valamiféle definícióra mindenképp szükségünk lesz a későbbiekben, kezdetnek két neves szerző egymásnak nem ellentmondó javaslatát vesszük alapul:

„Egy ágens bármi lehet, amit úgy tekinthetünk, mint ami az érzékelői segítségével érzékeli a környezetét, és beavatkozó szervei segítségével megváltoztatja azt.” (Russell és Norvig, 1995)

„Ágensnek nevezzük azt a rendszert, mely adott környezetbe helyezve képes autonóm cselekvésre céljai elérése érdekében.” (Wooldridge, 1999)



2.1 ábra: Ágens elvi alapmodellje

Az előbbi két ágens-definíciót foglalja össze az 2.1. ábrán feltüntetett absztrakt ágens-modell. Az *„autonóm”* ágens tehát egy adott környezetbe ágyazva cselekszik, minden cselekvésével meg-megváltoztatva annak pillanatnyi állapotát s így közvetve akár saját észleléseit is. Az *„autonóm”* szó intuitív jelentésén túli konkrét meghatározásához most is az előbbi két szerző javaslatát vesszük figyelembe:

„Egy rendszer olyan mértékig autonóm, amennyire a viselkedését saját tapasztalatai határozzák meg.” (Russell és Norvig, 1995)

„Az autonóm ágensek az ember, vagy más egyéb rendszer beavatkozása nélkül is képesek cselekedni: mind belső állapotukról, mind pedig viselkedésükről maguk rendelkeznek.” (Wooldridge, 1999)

A két definíció közül a dolgozat során ez utóbbi, gyengébb definícióra fogunk támaszkodni, mivel érzésem szerint ez áll közelebb az önállóság intuitív jelentéséhez. Megítélésem szerint az autonómia környezetfüggő, azaz: Egy-egy környezetben akár még tapasztalás nélkül, csupán beidegződések útján is képesek lehetünk arra, hogy önállóan, azaz autonóm módon elboldoguljunk. Más környezetben esetleg olyan behatások is érhetnek, melyekre semmilyen tapasztalat sem készíthet fel eléggé, s így, bár viselkedésünket színtisztán tapasztalataink határozzák meg, most mégis – önállóságunkat vesztve – külső segítségre szorulunk. E gondolatmenetet követve jutunk el a környezetek egyfajta osztályozásáig (Russell és Norvig, 1995):

Hozzáférhető, vagy *nem hozzáférhető*: A környezet hozzáférhető, ha az ágens érzékelői segítségével hozzáférhet a környezet teljes állapotához. Egy környezet ténylegesen hozzáférhető, ha az ágens érzékelése lefedi a környezet mindazon aspektusait, melyek szükségesek lehetnek egy-egy cselekvés kiválasztásához. Minél inkább hozzáférhető egy környezet, annál egyszerűbb felépítésű ágensek lesznek elegendőek hozzá. Egy teljesen hozzáférhető környezetben az ágensnek már benső állapotokra sincs szüksége ahhoz, hogy nyomon kövesse a környezet alakulását. Legtöbbször már a mérsékelten komplex környezetek sem hozzáférhetők. Nem hozzáférhető környezetre lehet példa a fizikai valóság, vagy akár az Internet… stb.

Determinisztikus, vagy *nem determinisztikus*: A környezet determinisztikus, ha tetszőleges állapotában az ágens által végrehajtott tetszőleges cselekvés egyértelműen meghatározza a cselekvés nyomán előálló következő környezeti állapotot. Egy determinisztikus, ámde nem hozzáférhető környezetet az ágens akár nem-determinisztikusnak is érzékelhet, így gyakran jobb a környezetet a determinizmus szemszögéből vizsgálni. A nem-determinisztikus környezetek (pl. fizikai valóság) nagyobb kihívást jelentenek az ágensek (és tervezőik) számára.

Epizódszerű, vagy *nem epizódszerű:* A környezet epizódszerű, ha benne az ágens tapasztalata „epizódokra” bontható. Ilyen környezetben az ágens teljesítménye a diszkrét számú „epizódon” vett teljesítményétől függ, ahol az egyes „epizódokon” vett teljesítmények függetlenek egymástól. Az epizódszerű környezetek egyszerűbbek, mint a nem epizódszerűek, mivel az ágensnek csak az adott „epizódon” belül kell előre tekintenie, hiszen az egyes „epizódokban” végrehajtott cselekvések nem

befolyásolják a következő „epizód” alakulását. Epizódszerű környezetre lehet példa egy sakkverseny, ami játszmák sorozatából áll, ahol az egyes játszmákon belüli cselekvések (lépések) nem befolyásolják a következő játszmák alakulását, így azok a sakkverseny epizódjainak tekinthetők.

Statikus, vagy *dinamikus:* A környezet statikus, ha csak az ágens cselekvéseinek nyomán módosul az állapota, egyébként változatlan. A dinamikus környezetet az ágens hatáskörén kívül eső jelenségek, folyamatok, vagy más egyéb hatások is befolyásolhatják, változtathatják. Szemi-dinamikus környezetről beszélünk, ha az idő előrehaladtával a környezet nem változik, de az ágens által elért teljesítmény mértéke igen. Erre lehet példa egy olyan sakkjátszma, amely időre megy.

Diszkrét, vagy *folytonos:* A környezet diszkrét, ha adott az ágens különböző lehetséges cselekvéseinek és észleléseinek egy véges számosságú halmaza. Az említett sakkjátszmában mind a különböző cselekvések, mind pedig a különböző játék-állások száma véges, így diszkrét környezetnek tekinthető. Folytonos környezetre lehet példa a taxisofőr-ágens környezete, ahol az ágens feladata egy taxi elvezetése. Ebben a környezetben mind a taxi állapota, mind pedig a taxi környezetének állapota folytonos tartományban mozog.

A legáltalánosabb környezet-osztály tehát a *nem hozzáférhető*, *nem determinisztikus*, *nem epizódszerű*, *dinamikus* és *folytonos* környezetek osztálya. Látható, hogy amíg az ágenst a környezet szemszögéből definiáltuk, addig a környezetet az ágenséből. Ez a kötődés teszi elválaszthatatlanná a két fogalmat egymástól, sajnos azonban nem ad semmiféle útmutatást sem a megvalósításukra vonatkozólag. A következő alfejezetek tehát az absztrakt és a konkrét ágens-architektúrák bemutatásán keresztül nyújtanak betekintést az ágensek megvalósítási lehetőségeibe. Megírásukkor javarészt Wooldridge e tárgyban íródott összefoglaló jellegű munkáira (Wooldridge, 1995; Wooldridge, 1999) támaszkodtam, mivel világos struktúrájuk és tárgyalásmódjuk jó és korszerű áttekintést nyújt a témában.

2.1.1. Absztrakt ágens-architektúrák

Az eddigiek során bevezetett absztrakt ágens-fogalmat könnyen formalizálhatjuk. Legyen $S = \{s_1, s_2, \ldots\}$ a lehetséges *ágens-környezetek* halmaza, ahol az ágens $A = \{a_1, a_2, \ldots\}$ *cselekvések* végrehajtására képes. Ekkor az ágenst tekinthetjük egy olyan *Cselekvés:* $S^* \to A$ alakú függvénynek, amely környezeti állapotok véges, de nem korlátos sorozataiból képez cselekvésekbe. Magyarán az ágens az általa megtapasztalt környezeti állapotokat használja fel ahhoz, hogy döntéseit meghozza. A *környezet* változását ekkor egy *Környezet:* $S \times A \to \wp(S)$ alakú függvény reprezentálhatja, amely a környezet-halmaz összes részhalmazának $\wp(S)$ halmazába, azaz a környezetek hatványhalmazába képez. A környezetet *determinisztikusnak* tekintjük, ha tetszőleges $s \in S$ és $a \in A$ bemenetre a *Környezet* függvény által adott kimenet egyelemű, egyébként pedig

nem-determinisztikusnak. Az ágens cselekvéseinek nyomán bekövetkezett környezet-változások sorozatát az *ágens történetének* nevezzük, és *h*-val jelöljük:

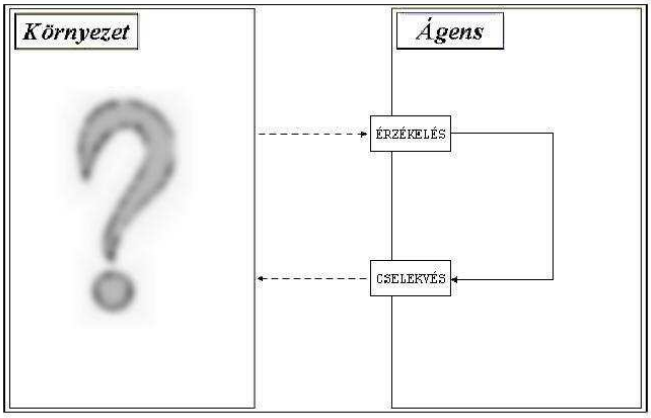
$$h: s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} s_n \xrightarrow{a_n} \cdots$$

Az *ágens karakterisztikus viselkedésének* nevezzük mindazon *h* történetek halmazát, melyekre teljesül, hogy

(1) $\forall n \in N$ *esetén fennáll, hogy* $a_n = Cselekvés(\{s_0, s_1, \ldots, s_n\})$, *és*

(2) $\forall n \in N, n > 0$ *esetén fennáll, hogy* $s_n \in Környezet(s_{n-1}, a_{n-1})$

Legtöbbször azonban az ágens nincs „közvetlen” kapcsolata a környezetével, hanem csak „közvetve”, szenzorok útján érzékeli azt. Ebben az esetben különválik benne az érzékelés és a cselekvés fázisa. Ezt a felosztást szemlélteti a 2.2. ábra.



2.2.ábra. Ágens érzékelésének és cselekvésének különválasztása

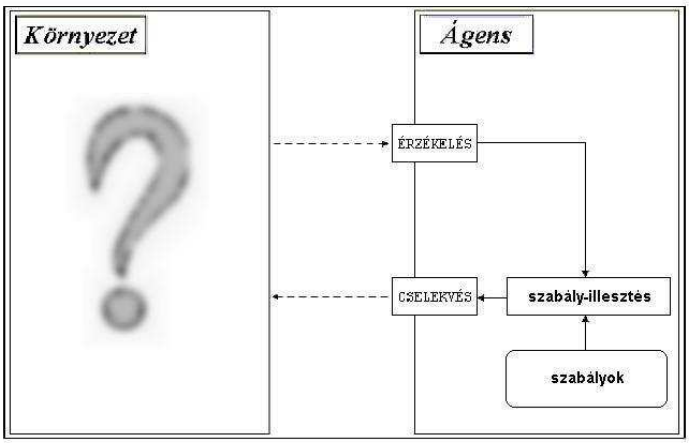
A 2.2. ábrán bemutatott ágens érzékelési mechanizmusát egy *Érzékelés:* $S \rightarrow P$ függvénnyel jellemezhetjük, mely a környezetek *S* halmazából az érzetek *P* halmazába képez; cselekvési mechanizmusát pedig egy *Cselekvés:* $P^* \rightarrow A$ alakú függvénnyel, amely érzetek véges, de nem korlátos sorozataihoz társít egy-egy cselekvést.

Ha az érzékelési mechanizmus „több-egy” leképzést valósít meg, akkor fennáll a lehetősége, hogy egy-egy érzethez több különböző környezet is tartozik. $s_i \in S$ és $s_j \in S$ környezeti állapotokat *megkülönböztethetetlennek* nevezzük, ha $s_i \neq s_j$ esetén *Érzékelés*(s_i) = *Érzékelés*(s_j) teljesül. Mindezen alap-definíciók ismeretében immár áttekinthetjük az absztrakt ágens-architektúrák különböző válfajait.

2.1.1.1. Egyszerű reflex-szerű ágens

A reflex-szerű ágensek – mint ahogy nevük is sejteni engedi – „beidegzett” módon cselekszenek. Minden egyes érzeti benyomáshoz egy-egy meghatározott cselekvést rendelnek. A

cselekvések hozzárendelése általánosságban tekinthető egyfajta szabály-illesztésnek, melyben a szabályok feltétel-része illeszkedik a beérkező érzetekre, törzsük pedig az adott érzethez tartozó cselekvést határozza meg. Egy ilyen elven működő ágens absztrakt architektúráját mutatja a 2.3.ábra. Az ábrán bemutatott egyszerű reflex-szerű ágens „egyszerűségéért" nagy árat fizet, hiszen se nem rugalmas, se nem hatékony.



2.3.ábra. Egyszerű reflex-szerű ágens

Merev, hiszen csak (táblázat-szerűen) előre megadott módon képes cselekedni, továbbá, mivel a szükséges – minden eshetőséget figyelembe vevő – szabályok száma a lehetséges érzetek és cselekvések számával exponenciálisan arányos, „viselkedésének kódolása" sem nevezhető túl hatékonynak. Az egyszerű reflex-szerű ágens működésének formális definíciója a következő:

(1) Érzékeli a környezetét.

Érzékelés: $S \rightarrow P$

(2) Az aktuális érzet és a szabályok halmaza alapján kiválaszt egy szabályt.

Szabály-Illesztés: $P \times \wp(R) \rightarrow R$

(3) Megcselekszi a kiválasztott szabály által leírt cselekvést.

Cselekvés: $R \rightarrow A$

2.1.1.2. Állapottal rendelkező reflex-szerű ágens

Az egyszerű reflex-szerű ágens hatékonyságát nagyban megnövelheti az, ha kiegészítjük egy *benső állapottal,* ahol információkat tárolhat érzeteivel és cselekvéseivel kapcsolatban. Az állapotot az aktuális érzet és állapot alapján újra és újra frissítheti a működés során, ezáltal mintegy „követve" a környezet alakulását, a pillanatnyilag esetleg nem érzékelhető környezeti jelenségeket, a környezethez való aktuális viszonyát, stb. A 2.4. ábrán látható az állapottal kiegészített egyszerű reflex-szerű ágens architektúrája, melynek formális működése a következő:

(1) Érzékeli a környezetét.

Érzékelés: $S \rightarrow P$

(2) Az aktuális érzet és benső állapot alapján frissíti benső állapotát.

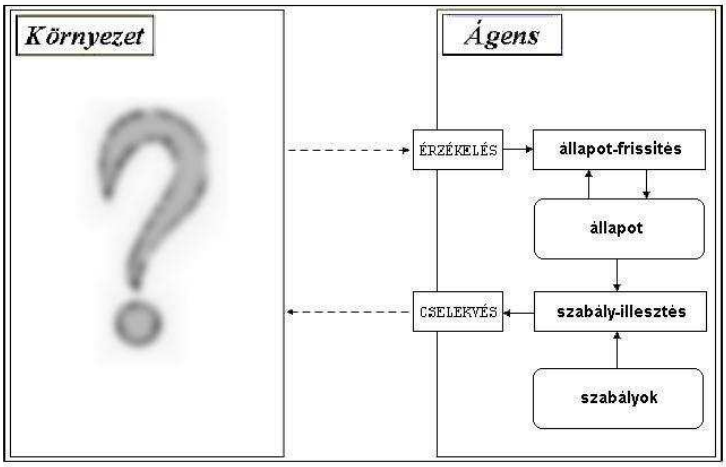
Állapot-Frissítés: $P \times I \rightarrow I$

(3) A benső állapot és a szabályok halmaza alapján kiválaszt egy szabályt.

Szabály-Illesztés: $I \times \wp(R) \rightarrow R$

(4) Megcselekszi a kiválasztott szabály által leírt cselekvést.

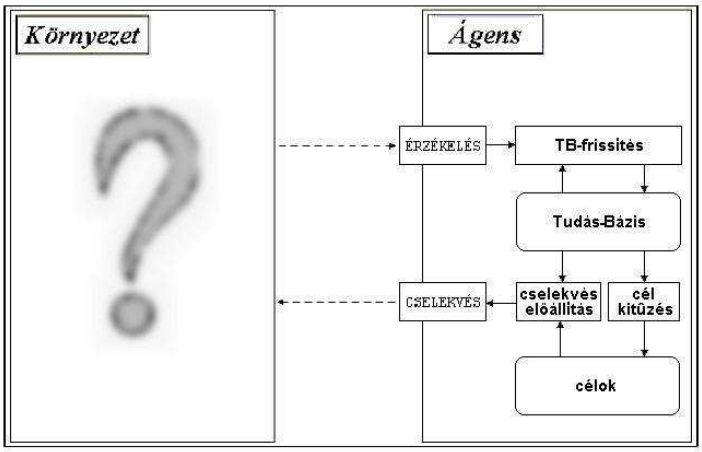
Cselekvés: $R \rightarrow A$



2.4.ábra. Állapottal rendelkező reflex-szerű ágens

2.1.1.3. Cél-orientált ágens

A reflex-szerű ágensek esetében a Tervezőnek kell gondoskodnia arról, hogy meghatározza, hogy *mikor-mi* történjen, s bár az így előálló működés valamiféle cél felé „tendál", a Tervező – főleg komplex esetben – ezt nemigen láthatja előre. Az ágens működése által megvalósított cél megváltoztatásához a szabály-halmaz – akár teljes – módosítása szükséges, ami komplex környezetek, ágensek és célok esetén nem túl célravezető. Ehelyett tekinthetünk a problémára fordítva: nem az ágenst konfiguráljuk a célhoz, hanem a célt az ágenshez. Magyarán, úgy tervezzük meg az ágenst, hogy képes legyen egy-egy adott cél elérésére. Ehhez lehetővé kell tennünk számára a cél eléréséhez szükséges cselekvések önálló előállítását.



2.5.ábra. Cél-orientált ágens működési elve

A cél automatikus elérésére több módszer is ismert, melyek közül a dolgozat az „automatikus tervkészítést” preferálja. Azonban absztrakt architektúrákról lévén szó, a célirányos cselekvések előállításának részleteitől most eltekintünk. A cél-orientált ágens működése a 2.5.ábrán követhető végig. A működés lefolyása formálisan a következő:

(1) Az ágens érzékeli a környezetét.
Érzékelés: $S \rightarrow P$

(2) Az aktuális érzet és a meglévő tudásbázis alapján frissíti tudásbázisát.
TB-Frissítés: $P \times B \rightarrow B$

(3) A frissített tudásbázis alapján kijelöli célok egy halmazát.
Cél-Kitűzés: $B \rightarrow C$

(4) A célok és a tudásbázis alapján előállítja a célok eléréséhez szükséges lépéseket.
Cselekvés-Előállítás: $C \times B \rightarrow \wp(A)$

(5) Megcselekszi az előállított lépések egyikét.
Cselekvés: $\wp(A) \rightarrow A$

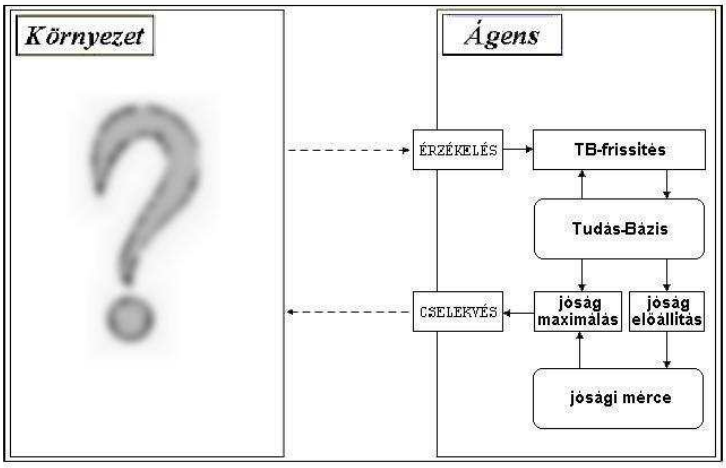
Több újdonság is feltűnik az előzőekhez képest: egyrészt állapot helyett immár tudásbázissal dolgozunk. A tudásbázis tárolhatja a környezet és a cselekvések modelljeit, továbbá az aktuális ismereteinket és múltbéli tapasztalatainkat, vagy bármi egyéb információt. A tudásbázis frissítése az állapotok frissítéséhez hasonlóan zajlik. Amikor a frissített tudásbázis alapján előállítjuk (frissítjük) a célok halmazát, akkor tulajdonképpen „automatikus probléma-felismerést” végzünk. Természetesen az is lehetséges, hogy a tudásbázisban a Tervező által kezdetben elhelyezett célokat „csiszoljuk” csak tovább – lényegében azonban mégis az ágensre bízzuk mind a probléma felismerését, mind a probléma megoldását.

2.1.1.4. Haszon-orientált ágens

Láthattuk, hogy ahhoz, hogy az ágenst meghatározott feladat teljesítésére bírjuk, valamiféle célt kell kitűznünk (vagy kitűzetnünk) elé. Sajnos azonban előfordulhatnak olyan esetek is, amikor a célt nem tudjuk explicite előre megadni, sőt, olyan eset is lehetséges, hogy még értelmezni sem tudjuk az adott környezet (pl. valós környezet) függvényében. Ilyenkor a célt – mint például a számunkra „leghasznosabb” állapotot – implicite kell definiálnunk egy hasznosság-függvény maximuma (vagy minimuma) által. A cél eléréséhez tehát a hasznosság-függvény maximalizálásán (vagy minimalizálásán) keresztül vezet az út.

A cél-orientált ágens esetében láthattuk, hogy a célt magamagának tűzi ki. Hasonlóan cselekszik a haszon-orientált ágens is: valamifajta hasznossági függvényt, *jósági mércét* definiál, aminek alapján „megbecsülheti” cselekvései jóságát. Absztrakt szinten tehát a haszon-orientált ágens működése lényegében megegyezik a cél-orientált ágensével, azzal a különbséggel, hogy amíg

az utóbbi a cél-állapotot explicite, addig az előbbi implicite, egy jósági mérce függvényében adja meg. A haszon-orientált ágens működését összefoglaló diagram a 2.6.ábrán látható.



2.6.ábra. Haszon-orientált ágens

2.1.2. Konkrét ágens-architektúrák

Eddig csak absztrakt szemszögből vizsgáltuk az ágenseket: nem foglalkoztunk avval, hogy az egyes, absztrakt szinten definiált ágens-komponensek miképpen vannak megvalósítva. Konkrét ágens-architektúrák alatt olyan architektúrákat értünk, melyekben definiált az egyes komponensek megvalósítási módja. A következő architektúrákról ejtünk szót:

Logikai-alapú ágensek: A döntéshozatal logikai következtetés útján történik.
Reaktív ágensek: A döntéshozatal egyfajta direkt leképzés helyzetből cselekvésbe.
Hiedelem-Vágy-Szándék alapú ágensek: A döntéshozatal az ágens hiedelmeit, vágyait és szándékait reprezentáló adat-struktúrák módosítása által jön létre.
Rétegelt ágens-architektúrák: A döntéshozatal különböző szoftver rétegek együttműködése révén valósul meg, ahol az egyes rétegek többé-kevésbé explicite, más-más absztrakciós szinten vonnak le következtetéseket a környezet állapotára vonatkozólag.

2.1.2.1. Logikai-alapú architektúrák

A mesterségesen intelligens rendszerek építésére vonatkozó klasszikus (szimbolikus MI) megközelítés szerint intelligens viselkedés érhető el a rendszer részéről pusztán a rendszer környezetének és kívánt viselkedésének *szimbolikus* reprezentációja, illetve annak *szintaktikai* manipulációja által. Esetünkben a szimbolikus reprezentáció *logikai formuláknak*, míg a szintaktikai manipuláció *logikai következtetésnek*, *tétel-bizonyításnak* felel meg.

Legyen adott első-rendű logikai állítások egy L halmaza, továbbá $D = \wp(L)$ az L állítások hatvány-halmaza, azaz összes részhalmazának halmaza. Az ágens aktuális benső állapotát jelölje $\Delta \in D$ tudásbázis. Jelölje $\Delta \vdash_{\rho} \phi$ a ϕ állítás Δ tudásbázisból való bizonyíthatóságát a ρ

következtetési szabályok által. Ekkor a logikai-alapú ágens-architektúra komponensei a következő függvényeket valósítják meg, hasonlóan az állapottal rendelkező ágensek esetéhez: <u>*Érzékelés*</u>: $S \rightarrow P$ az ágens percepciója, amely a környezeti állapotokból érzeteket állít elő. <u>*Állapot-frissítés*</u>: $D \times P \rightarrow D$, amely egy-egy tudásbázis és érzet együtteséből egy-egy új tudásbázist állít elő. <u>*Cselekvés*</u>: $D \rightarrow A$ modellezi az ágens beavatkozó-szerveinek funkcionális működését, ahol a cselekvések előállítása a tudásbázisból kiinduló logikai következtető lépések felhasználásával történik. Ennek pszeudó-kódban írott változata lehet a következő:

```
function Cselekvés(Δ∈ D) returns a∈ A {
      for each a∈ A do {
      if Δ ├ρ Végrehajt(a) then {
            return a
            }
      }
      for each a∈ A do {
      if ¬(Δ ├ρ ¬Végrehajt(a)) then {
            return a
            }
      }
      return null
}
```

Az elképzelés lényege tehát az, hogy a programozó úgy kell, hogy megadja a ρ következtetési szabályokat, és a Δ tudásbázist, hogy, ha Δ-ból ρ által kikövetkeztethető a `Végrehajt(a)` állítás, akkor a legyen az adott helyzetben végrehajtható legjobb cselekvés. Így hát az ágens aktuális cselekvésének kiválasztása úgy történik, hogy rendre végighaladunk az A halmaz elemein, miközben minden elemre megvizsgáljuk, hogy a bizonyítani kívánt `Végrehajt(a)` állítás levezethető-e az aktuális Δ tudásbázisból a ρ következtetési szabályok által. Ha igen, akkor az a cselekvéssel térünk vissza, ha nem, akkor hasonlóan, egy olyan a cselekvést keresünk, amelyre a $\neg$`Végrehajt(a)` állítás nem következik a Δ tudásbázisból, azaz amely egyértelműen nem kizárt, s így konzisztens a Δ tudásbázissal. Ha ilyen cselekvés sincs, akkor egy speciális *null* jelű cselekvéssel térünk vissza.

A logikai-alapú megközelítések nagy előnye a tiszta, egyszerű és elegáns leírásmód *(szemantika)*. Hátránya viszont a nagy számítási komplexitás mellett, amely megkérdőjelezni látszik az ilyen alapú megközelítések valós-idejű felhasználhatóságát, a komplex és dinamikus környezetek reprezentálhatóságának máig megoldatlan problémája.

2.1.2.2. Reaktív architektúrák

A szimbolikus/logikai alapon működő ágenseknél felmerülő problémák arra ösztönöztek kutatókat, hogy használható alternatívát keressenek a „*szimbolikus* MI” paradigmájára. A ’80-as évek második felében intenzív kutatás indult egy alapjaiban új megközelítés irányában. A kutatókat

inkább az addigi meggondolások elutasítása, mintsem közösen elfogadott álláspontjuk fogta össze (Wooldridge, 1999):

- ***A szimbolikus reprezentációk és az azokon végzett szintaktikai manipulációkon alapuló döntés-hozatali eljárások elutasítása.***
- ***Az intelligens, racionális viselkedés nem választható külön a környezet fogalmától: az ágens és a környezete közti interakció eredményeképp áll elő.***
- ***Az intelligens viselkedés számos egyszerűbb viselkedés elegyéből születik.***

Az ilyesfajta alternatívákra legtöbbször, mint *viselkedés-alapú, beágyazott* és *reaktív* megközelítésekre hivatkoznak. *Viselkedés-alapú*, hiszen mindegyikben megjelenik a különböző egyéni viselkedések elegyítésére tett törekvés. *Elhelyezett,* mivel az ágensekre legtöbbször a környezet részeként tekintenek. *Reaktív,* mert az ilyen alapon működő rendszereket gyakran nem tekintik többnek, mint környezetükre mindenfajta következtetés nélkül pusztán csak reagálni képes rendszereket. Ebben a szakaszban a legismertebb reaktív architektúrát, az *alárendeltségi architektúrát* (Brooks, 1991) mutatjuk be.

Az alárendelt architektúrának két fő jellemzője van: Egyrészt az ágens döntés-hozatali mechanizmusa ún. *feladatvégző viselkedések* egy halmaza által jön létre, ahol az egyes viselkedések érzékelésből cselekvésbe képző függvényekként foghatók fel, melyek mind egy-egy feladat végrehajtására törekszenek, gyakorta $helyzet \rightarrow cselekvés$ alakú direkt leképzésekként megadva. (Fontos megjegyezni, hogy a feladatvégző viselkedések *nem* tartalmaznak komplex szimbolikus reprezentációt, és *nem* végeznek semmiféle szimbolikus következtetést sem.) Másrészt a különböző viselkedések akár egyszerre is *tüzelhetnek.* Ezért Brooks a viselkedés-modulok *alárendeltségi hierarchiába* szervezését javasolta, *rétegekbe* rendezve a viselkedéseket. Az alacsonyabb szinteken elhelyezkedő viselkedések *meggátolják* a magasabb szinteken lévő viselkedések érvényesülését.

Az *Érzékelés* komponens által megvalósított függvény változatlanul, a logikai-alapú ágenseknél látott formában van jelen. A *Cselekvés* a viselkedések *gátlás* reláció által rendezett halmazán jön létre. A viselkedés egy *(c,a)* páros, ahol $c \subseteq P$ (érzetek egy halmaza) a viselkedés *feltétele*. Egy *(c,a)* viselkedés *tüzel,* ha a környezet egy olyan $s \in S$ állapotban van, amelyre $Érzékelés(s) \in c$ teljesül, azaz, ha az ágens aktuális érzete teljesíti az adott viselkedés által előírt feltétel-részt. Legyen $Viselkedések = \{(c,a) \mid c \subseteq P, a \in A\}$ az összes lehetséges viselkedést leíró szabály halmaza. Ekkor minden ágens $R \subseteq Viselkedések$ viselkedés-szabályainak halmazához tartozik egy-egy $\prec \subseteq R \times R$ bináris *gátlás reláció,* amely az *R* halmaz teljes rendezésének felel meg (azaz tranzitív, irreflexív és antiszimmetrikus). Minden $b_i, b_j \in R$ viselkedés-párra fennáll, hogy $b_i \prec b_j$, azaz „b_i *gátolja* b_j *–t",* ha $(b_i, b_j) \in \prec$ teljesül. Ez tehát azt jelenti, hogy b_i alacsonyabb szinten helyezkedik el az *alárendeltségi hierarchiában*, mint b_j, s így magasabb a prioritása. Ha tehát az

ágens aktuális észlelése mind b_i, mind pedig b_j feltétel-részét teljesítené, b_i érvényesülne, s így az általa definiált a_i cselekvés hajtódna végre. A *Cselekvés* függvény pszeudo-kódja a következő:

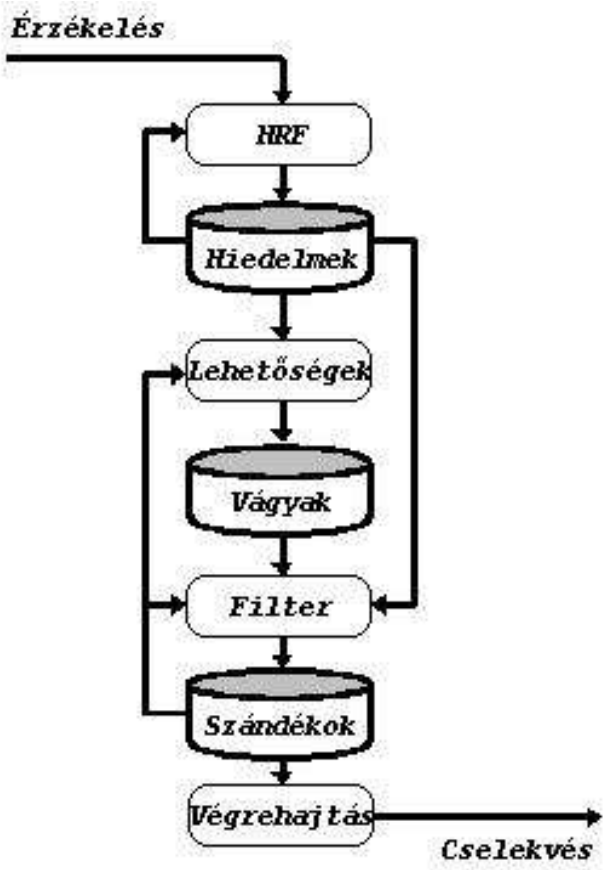
```
function Cselekvés(p ∈ P) returns a∈A {
      var tüzel∈ ℘(R)
      var kiválasztott∈A
      tüzel:= {(c,a) | (c,a)∈ R, p∈ c}
      for each (c,a)∈ tüzel do {
            if ¬(∃(c',a')∈ tüzel ,(c',a') ≺ (c,a)) then {
            return a
            }
      }
      return null
}
```

Kezdetben tehát előállítjuk a `tüzel` halmazt, amely tartalmazza a *p* észlelés hatására tüzelő összes viselkedést. Végighaladva a kapott halmazon, azt a viselkedést választjuk ki, amelynél nincs magasabb prioritású, aktuálisan tüzelő viselkedés az *alárendeltségi hierarchiában.* Ha a `tüzel` halmaz nem üres, akkor ilyennek mindenképp lennie kell. Az így kapott viselkedéshez tartozó *a* cselekvéssel térünk tehát vissza, ez lesz az ágens *p* észlelésre adott válasza. Ha ellenben a `tüzel` halmaz üresnek bizonyulna, azaz az aktuális *p* észleléshez nem definiáltunk egy viselkedést sem, akkor a speciális *null* jelű cselekvéssel térünk vissza. – Az alárendeltségi architektúra alkalmazásának egy igen sikeres, egyszerű példája a Luc Steels-féle Mars-felfedező robotok szoftver-architektúrája (Steels, 1990).

Összefoglalva megállapíthatjuk, hogy a Brooks által javasolt architektúra egyszerű, gazdaságos, csekély számításigényű, robusztus és elegáns megoldást ad a reaktív alapon működő ágensek megvalósítására. Sajnos azonban számos alapvető, máig megoldatlan probléma fémjelzi nem csak az alárendeltségi, hanem a tisztán reaktív architektúrákat is: Ha ágensünk semmiféle modellt sem használ a környezet reprezentációjára, akkor pusztán a *lokális* környezetére alapozva kell meghoznia a legjobb cselekvésre vonatkozó döntését. Mivel azonban a tisztán reaktív ágensek csupán a *lokális* információk alapján hozzák meg döntésük, ezért gyakorlatilag „rövid-távon" gondolkodva, kizárt számukra a *nem-lokális* információk figyelembevétele. Továbbá kérdéses a tisztán reaktív architektúrák tapasztalat útján történő *tanulási* képességének megvalósíthatósága. Komplex viselkedések megalkotásakor nagyon sok „elemi" viselkedés kölcsönhatását kellene számításba venni, aminek jelenleg még nincs kialakult módszertana. Amíg az egyszerűbb ágensek még manuálisan is megtervezhetők, addig a *többrétegű* viselkedéssel rendelkező ágensek tervezése már alapvető nehézségekbe ütközik. Számos megoldási javaslat született e hátrányok kiküszöbölésére, amelyek közül talán az ágensek evolúció útján történő kialakítása a legismertebb.

2.1.2.3. Hiedelem-Vágy-Szándék (Belief-Desire-Intention) architektúrák

Ezek az architektúrák a *gyakorlati következtetés* filozófiáján alapszanak (Bratman, 1987), ami röviden arra próbál választ adni, hogy *hogyan* valósítsuk meg azt, *amit* szeretnénk. Az előbbi folyamat *eszköz-cél* következtetés, az utóbbi pedig *megfontolás* néven vált ismertté. Ezek szerint a döntési folyamat általában a *lehetőségek* felmérésével kezdődik. Miután rendelkezésünkre állnak a *lehetőségek,* ki kell választanunk közülük néhányat, aminek irányában *elkötelezzük* magunkat, amikre *vágyunk* – ezekből lesznek aztán a *szándékok*, amelyek később meghatározzák jövőbeni cselekedeteinket. Ez tehát azt jelenti, hogy, ha szándékaink valósak, akkor a későbbiekben *ki kell tartanunk* mellettük. *Tennünk* kell értük, *épeszű próbálkozást* kell tennünk a megvalósításuk érdekében, s ha ezek esetleg csődöt mondanának, *újra kell próbálkoznunk* mindaddig, amíg vagy nem sikerülnek, vagy lehetetlenné nem válnak, vagy maga a szándék értelmét nem veszti számunkra, azaz megszűnik az *ok*, amiért fenntartottuk. A fenntartott szándékok természetesen meghatározzák későbbi gyakorlati következtetéseinket, hiszen nem választhatunk olyan lehetőségek közül, amelyek ellentmondanak addigi szándékainknak, azaz, amelyek *inkonzisztensek* velük. Szándékaink szorosan összefüggnek a jövőre vonatkozó *vágyainkkal*, hiszen amire szándékszunk, annak a megvalósulásában hinnünk kell, máskülönben értelmetlenné tennénk szándékainkat.



2.7. ábra: A Hiedelem-Vágy-Szándék architektúra

Látható, hogy a gyakorlati következtetés kulcsfontosságú motívuma a szándékok kialakítása. A gyakorlati következtetésen alapuló ágensek megvalósításában tehát kulcsfontosságú

szerepet játszik a szándékok megfelelő *egyensúlyban* tartása. Néha szükséges lehet *újragondolni* a meglévő szándékokat, s azokat, amik felvételük óta lehetetlenné váltak, értelmüket vesztették, vagy megvalósultak, az ágensnek *el kell hagynia*. Gyakorlatilag tehát a *pro-aktív (cél-orientált)* és a *reaktív (esemény-vezérelt)* viselkedések közt kell megtalálni az egyensúlyt. Gyorsan változó környezetben érdemes gyakrabban újraértelmezni a meglévő szándékokat (inkább a reaktív döntési stratégia követendő), míg lassabban változó környezetben kevesebbszer (inkább a cél-orientált, pro-aktív stratégia követendő). Erre a jelenségre mutat tapasztalati bizonyítékokat David Kinny és Michael Georgeff „dMARS" nevezetű ágens-rendszere (d'Inverno és társai, 1997).

A Hiedelem-Vágy-Szándék architektúrán alapuló ágensek gyakorlati következtetésének folyamatát szemlélteti az 2.7. ábra (Wooldridge, 1995). Az ábrán hét főbb komponens látható:

- Adott *Hiedelmek* egy halmaza, amely az ágens környezetről alkotott elképzeléseit foglalja magában – reprezentálja.
- Adott egy *Hiedelem Revíziós Függvény (HRF)*, amely az aktuális érzet és a meglévő hiedelmek alapján frissíti a hiedelmek halmazát.
- Adott egy *Lehetőség Előállító Függvény,* amely a meglévő hiedelmek és szándékok felhasználásával előállítja az ágens lehetőségeit (vágyait).
- Adott *Vágyak* egy halmaza, amely az ágens cselekvéseinek lehetséges lefolyásait reprezentálja.
- Adott egy *Filter Függvény,* amely az ágens *megfontolásait* modellezve – a meglévő hiedelmek, vágyak és szándékok alapján – frissíti a szándékok halmazát.
- Adott *Szándékok* egy halmaza, amely az ágens által megteremteni kívánt viszonyokat reprezentálja, amikkel kapcsolatban az ágens elkötelezte magát.
- Adott egy *Cselekvés Választó Függvény (Végrehajtás),* amely a meglévő szándékok alapján kiválasztja az ágens által végrehajtandó cselekvést.

Legyen *Bel* az összes lehetséges hiedelem halmaza, *Des* az összes lehetséges vágy halmaza, és *Int* az összes lehetséges szándék halmaza. Ekkor az ágens tetszőleges időpillanatban vett állapotát egy-egy *(B,D,I)* hármas határozza meg, ahol $B \subseteq Bel, D \subseteq Des, I \subseteq Int$. Az ágens hiedelem revíziós függvénye egy leképezés, amely az aktuális érzet és a meglévő hiedelmek alapján előállítja az új hiedelmek halmazát. A függvény részletes tárgyalása sajnos kívül esik a dolgozat keretein. A függvény formális definíciója:

$$HRF : \wp(Bel) \times P \rightarrow \wp(Bel)$$

A lehetőségeket előállító függvény hiedelmek és szándékok halmazából képez vágyak egy halmazába. Elsősorban a szándékok megvalósításának mikéntjéért felelős (a *hogyan* felől dönt). Így, ha tegyük fel ágensünk szándékában áll *x,* akkor olyan lehetőségeket kell kínálnia számára, amelyekkel *x elérhető.* Minthogy ezek a lehetőségek *x–nél* alacsonyabb absztrakciós szinten

definiáltak, ezért egy idő után, a lehetőségek periodikus szándékká alakulásával, a lehetőségek egyre konkrétabbá válnak, mígnem végül végrehajtható szándékokká alakulnak. A lehetőségek előállításán, azaz az eszköz-cél következtetésen túl függvényünknek figyelembe kell vennie egyéb megkötéseket is. Először is *konzisztensnek* kell lennie, ami azt jelenti, hogy csak olyan lehetőségeket állíthat elő, amelyek nem mondanak ellent az ágens meglévő hiedelmeinek és szándékainak (ennek vizsgálata az említett halmazok logikai reprezentációja esetén nem jelent gondot). Másodszor pedig *opportunista* módon kell megválasztania a lehetőségeket, amennyiben képes kihasználni a környezet kedvező alakulását. A függvény formális definíciója:

$$\textit{Lehetőségek:}\ \wp(Bel)\times\wp(Int)\rightarrow\wp(Des)$$

A *Filter* függvény valósítja meg az ágens megfontolási mechanizmusát (a *mit* felöl dönt). Gyakorlatilag az ágens szándékainak frissítését végzi az aktuális vágyak, hiedelmek és szándékok alapján. Egyrészt *el kell vetnie* a megvalósíthatatlan és túl költséges szándékokat. Másodsorban *fenn kell tartania* a még megvalósítatlan, ámde vélhetőleg nem túl költséges szándékokat. Harmadrészt *fel kell vennie* olyan új szándékokat, amelyek vagy az eddigiek megvalósítását segítik, vagy új lehetőségek kiaknázását teszik lehetővé. A függvény formális definíciója a következő:

$$\textit{Filter:}\ \wp(Bel)\times\wp(Des)\times\wp(Int)\rightarrow\wp(Int),\ \textit{ahol}$$

$$\forall B\in\wp(Bel),\forall D\in\wp(Des),\forall I\in\wp(Int)\ \textit{esetén teljesül a}$$

$$Filter(B,D,I)\subseteq I\cup D\ \ \textit{megkötés.}$$

Ez tehát azt jelenti, hogy a *Filter* függvény nem teremt *új* szándékokat, hanem a már meglévő szándékok és lehetőségek közül „válogatja ki” az új szándékokat.

A cselekvés kiválasztó függvény a végrehajtható szándékok kiválasztásáért felelős. Formális definíciója:

$$\textit{Végrehajtás:}\ \wp(Int)\rightarrow A$$

Végeredményben tehát az ágens beavatkozó-szervei által megvalósított függvény változatlanul *Cselekvés:* $P\rightarrow A$ formájú, aminek pszeudo-kódja most a következő:

```
function Cselekvés(p∈P) returns a∈A {
      B := HRF(B, p)
      D := Lehetőségek(B, I)
      I := Filter(B, D, I)
      return Végrehajtás(I)
}
```

Tehát a Hiedelem-Vágy-Szándék architektúrájú ágens működése a következő: (1) Az ágens érzékeli a környezetét, majd (2) az előálló érzet és a meglévő hiedelmek alapján frissíti a világról

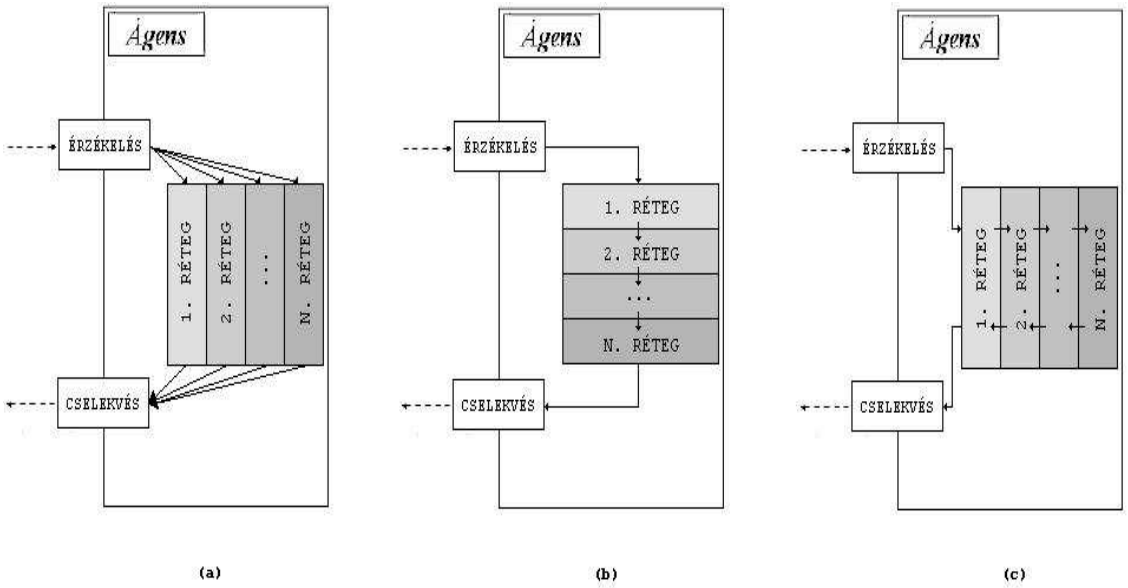
alkotott addigi elképzeléseit. (3) A frissített hiedelmek, és a meglévő szándékok alapján vágyakat tűz ki maga elé, amiket aztán (4) újabb szándékokká alakít, majd (5) lehetőség szerint végre is hajt.

Összefoglalva tehát, a Hiedelem-Vágy-Szándék alapú architektúrák olyan, a gyakorlati következtetés filozófiáján (Bratman, 1987) alapuló architektúrák, melyekben előnyös tulajdonságként jelenik meg a döntés folyamatának intuitív jellege, továbbá az ágens tisztán funkcionális dekompozíciója, ezzel is megkönnyítve az ilyen alapon működő ágensek tervezését. Komoly nehézséget okoz viszont az ágens szándékokhoz való elkötelezettségének szabályozása, egyensúlyban tartása: a pro-aktív és a reaktív viselkedés egyensúlyának megteremtése.

2.1.2.4. Rétegelt architektúrák

A rétegelt architektúrák megoldást jelenthetnek a Hiedelem-Vágy-Szándék architektúrájú ágensek viselkedésének szabályozására. A rétegelt architektúra elve szerint ugyanis az ágens alrendszereit *rétegekbe* kell szervezni ahhoz, hogy az ágens viselkedése több különböző (pl. pro-aktív, reaktív vagy szociális) viselkedés együtteseként jöjjön létre, ahol a különböző rétegek felelősek az ágens különböző viselkedéseiért.

A rétegek szervezésének két fő módja ismeretes: A *horizontálisan rétegelt* architektúrák esetében minden réteg közvetlen kapcsolatban áll mind a szenzorikus bemenettel, mind a cselekvést előállító kimenettel. Ennek következtében az egyes rétegek – melyek így egy-egy ágensnek tekinthetők – külön-külön tesznek javaslatot a végrehajtandó cselekvésre, hasonlóan a neurális hálózatoknál ismeretes „Mixture of Experts” szervezéshez (Jordan és Jacobs, 1994). A *vertikálisan rétegelt* architektúrák esetén legfeljebb csak egy-egy réteg foglalkozik a bemenettel és a kimenettel, hasonlóan a számítógép-hálózatoknál ismeretes rétegelt szervezéshez (Tanenbaum, 1999). Az 2.8. ábra mutatja a rétegelt architektúrák szervezési módjait:



2.8. ábra: Rétegelt architektúrák: (a) Horizontális, (b) Egy-utas vertikális, (c) Két-utas vertikális

A horizontálisan rétegelt architektúrák (2.8.(a) ábra) nagy előnye egyszerűségükben rejlik. Elég pusztán N réteg megvalósítása egy N viselkedéssel rendelkező ágens létrehozásához. Fennáll azonban az a veszély, hogy az egymással „versengő” rétegek az ágens szintjén inkoherens viselkedést eredményeznek. A konzisztens viselkedésért általában egy *döntéshozó mechanizmus*, egy ún. *mediátor* felel, amely arról dönt, hogy a különböző rétegek közül aktuálisan melyik érvényesüljön. A mediátor megtervezése sajnos korántsem triviális feladat; bevezetésével „lekorlátozzuk” az ágens viselkedését.

A vertikálisan rétegelt architektúrák részben megoldást jelentenek ezekre a nehézségekre. Az egy-utas változatban (2.8.(b) ábra) a vezérlés rétegről rétegre halad mindaddig, amíg a kimeneti – a cselekvés előállításáért felelős – réteghez nem ér. A két-utas változatban (2.8.(c) ábra) a vezérlés először rendre végighalad a rétegeken, majd visszafelé is megteszi ugyanezt az *utat,* mígnem visszaér a bemeneti (s egyben kimeneti, cselekvést előállító) réteghez.

A döntéshozatal leegyszerűsödése azonban újabb problémákkal jár. A vertikálisan rétegelt architektúrák kevésbé *flexibilisek*, mint horizontális vetélytársaik. Vertikális esetben a vezérlésnek végig kell haladnia minden egyes rétegen ahhoz, hogy megszülessen a cselekvésre vonatkozó döntés. Egy-egy réteg meghibásodása az egész folyamat meghibásodását okozhatja.

Mindennek ellenére a rétegelt architektúrák tekinthetők a manapság legelterjedtebb, általános ágens-architektúra osztálynak. A rétegek által megvalósított funkcionális dekompozíció természetessé és átláthatóvá teszi az ágensek tervezését. A reaktív, pro-aktív és szociális viselkedés pusztán csak a megfelelő rétegek beiktatásával elérhető. A rétegelt architektúrák nagy hátránya viszont az, hogy a nem rétegelt architektúrákkal szemben nincsen általánosan megfogalmazott, jól definiált, egységes szemantikájuk és működési koncepciójuk. Másrészt az egyes rétegek közötti együttműködés sincs általánosan megoldva. Horizontálisan rétegelt architektúrára mutat jó példát Ferguson „TOURINGMACHINES” koncepciója (Ferguson, 1992), vertikálisra pedig Müller „INTERRAP” elnevezésű architektúrája (Müller, 1993).

2.2. TERVKÉSZÍTÉS

Az előzőekben bemutatott ágens-architektúrák sejteni engedték, hogy az ágensnek nem mindig lesz elegendő pusztán csak az aktuális megfigyelést, vagy benső állapotot figyelembe vennie ahhoz, hogy a megfelelő döntést meghozhassa (lásd. Reaktív ágens). A megfelelő cselekvés kiválasztása legtöbbször bizonyos szintű „előrelátást” kíván, aminek során az ágens különböző cselekvés-sorozatokat mérlegel annak eldöntése érdekében, hogy kiválaszthassa közülük a céljainak legmegfelelőbbet. Ezt a folyamatot nevezzük *tervkészítésnek*.

Ebben a szakaszban a tervkészítés elméletével és az ágensek tervezési képességeivel foglalkozunk. Kezdetben bevezetjük a probléma és a terv fogalmát. Ezt követi a terv-reprezentációk és a tervkészítési algoritmusok leírása. Végül összefoglaljuk a tervkészítő ágensek működését. A

szakasz megírásához Stuart J. Russell MI könyvének e tárgyban íródott fejezeteit vettem alapul (Russell és Norvig, 1995), mivel alapos tárgyalásmódjuk alátámasztja elképzeléseimet.

2.2.1. Probléma és Terv

Problémának nevezzük az ágens-környezet *kiinduló állapotának*, *cél-állapotainak* és lehetséges *cselekvéseinek* együttesét. Ekkor a *probléma megoldása* a cselekvések egy olyan sorozata, melyek végrehajtása a kiinduló állapotból a cél-állapotok valamelyikébe vezet. A cél-állapot megadása lehet explicit, vagy implicit: Explicit akkor, ha a cél-állapotok halmaza egyértelműen adott, és implicit akkor, ha nem. Az utóbbi esetben többnyire egy-egy *cél-függvény* segítségével dönthetjük el, hogy az adott állapot része-e a cél-állapotok halmazának. Cél-állapotnak tekinthetjük mindazon állapotokat, melyekhez a cél-függvény – a hozzájuk vezető cselekvés-sorozat ismeretében – egy-egy adott értéket rendel (lehet ez akár a függvény maximuma, minimuma, vagy bármely más értéke). A problémákat a következőképp osztályozhatjuk:

- *Egyállapotú problémák* azok, amelyek olyan hozzáférhető (kvázi determinisztikus) környezetet írnak le, ahol az egyes cselekvések kimenetele az ágens számára teljes egészében ismert.
- *Többállapotú problémák* azok, amelyek olyan, nem hozzáférhető (kvázi nem-determinisztikus) környezetet írnak le, ahol az egyes cselekvések kimenetele az ágens számára teljes egészében ismert.
- *Eshetőségi problémák* azok, amelyek olyan, nem hozzáférhető (kvázi nem-determinisztikus) környezetet írnak le, ahol az egyes cselekvések kimenetele az ágens számára csak részben ismert.
- *Felderíthetőségi problémák* azok, amelyek olyan, nem hozzáférhető (kvázi nem-determinisztikus) környezetet írnak le, ahol az egyes cselekvések kimenetele az ágens számára (kezdetben) egyáltalán nem ismert.

Látható, hogy a környezetek definíciójához hasonlóan a problémák definíciója is az ágens szemszögéből történt. Ez nem jelent megkötést a problémák körére vonatkozólag, mivel az ágens, illetve a környezet kellő általánosítása esetén az ágens-környezet által reprezentált probléma ekvivalens lehet bármely általános értelemben vett problémával. A problémák megoldására szolgálnak a tervek:

Tervnek nevezzük *lépések* egy halmazát és a rajtuk értelmezett *kényszerek* és *relációk* összességét. A tervnek ez a – már-már megfoghatatlanul általános – definíciója nem véletlen, ugyanis a különböző tervkészítő módszerek más-más módon definiálják és reprezentálják mind a lépéseket, mind a kényszereket, mind a relációkat, miközben gyakorlatilag mind a fenti, általános definíció speciális esetei. A tervekkel kapcsolatos fogalmak részletesebb leírását a következő szakasz szolgáltatja.

2.2.2. Reprezentációk és Algoritmusok

A tervkészítési módszereket a problémamegoldó módszerektől általában a cselekvések, állapotok, célok és tervek reprezentációja különbözteti meg. Amíg a problémamegoldó módszerek *keresést* végeznek az állapotok vagy tervek terében, addig a tervkészítő módszerek esetében a terv megtalálására más-más, *specifikus módszerek* szolgálnak. A tervkészítés három fő alapgondolata:

- *Az állapotok, célok és cselekvések hozzáférhetővé tétele.* Ez azt jelenti, hogy a problémamegoldó módszerekkel ellentétben az ágens valamifajta formális leírást használ az állapotok, célok és cselekvések reprezentációjára. Az előbbi kettőt általában logikai mondatok formájában tárolja, míg a cselekvésekhez logikai elő és utófeltételeket rendel. Ez lehetővé teszi az állapotok és a célok közti közvetlen kapcsolatot, amennyiben az elő és utófeltételek környezeti állapotokra hivatkozó logikai állítások.
- *A tervkészítés függetlenítése a cselekvések végrehajtási sorrendjétől.* Ez azt jelenti, hogy a terv kialakítása során nem kell a cselekvések majdani sorrendjét követnünk a cselekvések tervbevételekor. A problémamegoldó módszerek stratégiája ellenben – a keresés következtében – mindig csak a majdani végrehajtás sorrendjében „fűzheti hozzá" a tervezett cselekvéseket a tervhez. A tervkészítés e szabadsága sokkal hatékonyabbá teheti a tervezést.
- *A világ független alkotóelemeinek felismerése.* Ez pedig azt jelenti, hogy a célok többnyire független részcélokra oszthatók, melyekre külön-külön részterv készíthető, s így a terv (megoldás) a résztervek (részmegoldások) összefűzésével állítható elő. A felbontást a reprezentáció formális jellege teszi lehetővé, nagyban megnövelve ezáltal a tervkészítés problémamegoldó módszerekhez viszonyított hatékonyságát.

A következőkben elvi sémákat mutatunk be különböző tervkészítő módszerek megvalósítására. Elsőként a részben rendezett tervkészítés (RRT), majd a módszer egy hatékony kiterjesztésének (RRT-DUNF) ismertetése kerül sorra, amit a hierarchikus dekompozíció (HD-RRT) bemutatása után a feltételes tervkészítés (FRRT) elvének leírása zár. Mielőtt azonban még rátérnénk az algoritmusok tárgyalására, röviden ejtsünk szót a tervekkel kapcsolatos főbb alapfogalmakról.

A terv egyik legfontosabb alkotóeleme az operátor. Az *operátor* egy hármas adatstruktúraként fogható fel, melynek építőelemei a következők:

- *Cselekvés leírás:* az ágens-környezet számára szolgáló cselekvés-megnevezés.
- *Előfeltételek:* a cselekvés végrehajtásához szükséges környezetre vonatkozó állítások.
- *Következmények:* a cselekvés végrehajtása utáni környezetre vonatkozó állítások.

Legyen *o* egy operátor. Jelölje *o* előfeltételeit *elő(o),* következményeit pedig *köv(o).* Kezdetben csak olyan operátorokkal foglalkozunk, melyek előfeltételei ponált, míg következményei ponált és negált elsőrendű logikai kifejezések konjunkciójaként adottak, ahol a cselekvésleírás maga is egy logikai kifejezés (pozitív literál). Ekkor egy *o* operátor általános felírása a következő:

$Op($CSELEKVÉSLEÍRÁS: *cselekvés_név*(A_1, ..., A_n),
ELŐFELTÉTEL: $elő_1\left(E_{1,1},\ldots,E_{1,k_1}\right)\wedge\ldots\wedge elő_p\left(E_{p,1},\ldots,E_{p,k_p}\right)$,
KÖVETKEZMÉNY: $köv_1\left(U_{1,1},\ldots,U_{1,l_1}\right)\wedge\ldots\wedge köv_q\left(U_{q,1},\ldots,U_{q,l_q}\right)$).

Látható, hogy *o*-ban *p* előfeltétel és *q* következmény szerepel, ahol a cselekvés egy *n*–argumentumú, az *i.* előfeltétel egy k_i–argumentumú, a *j.* következmény pedig egy l_j–argumentumú kifejezés. Az következmény-részben megállapodásunkhoz híven mind ponált, mind negált kifejezések szerepelhetnek.

Operátor-sémának nevezzük azt az operátort, amelyben nem minden kifejezés minden argumentuma *behelyettesített*, hanem létezik olyan kifejezés is, melynek valamely argumentuma *változó*. Az *o* operátort *alkalmazhatónak* nevezzük az ágens-környezet egy *s* állapotában, ha *o* változóinak létezik egy olyan lehetséges behelyettesítése, hogy a behelyettesített operátor előfeltételeire teljesül $elő(o)\subseteq s$. Az operátor végrehajtása után *köv(o)* összes ponált állítása igaz lesz *s* összes állításával egyetemben, kivéve azokat az *s*-beli (ponált) állításokat, amelyek *köv(o)*-ban negáltan szerepeltek. Az *s*-beli negált állításokat tehát mindaddig igaznak fogjuk tekinteni (implicite), amíg *s*-ben nem szerepel (explicite) az állítás ponált változata. (Ez egy igen „negatív", ámde hatékony szemlélet).

Immár minden információ a rendelkezésünkre áll ahhoz, hogy definiálhassuk a terv fogalmát. A *terv* egy négyelemű adatstruktúra, melynek komponensei (Russell és Norvig, 1995):

- *A terv lépéseinek* halmaza, ahol a lépések a problémához kapcsolódó operátorok közül kerülnek kiválasztásra.
- *Sorrendi kényszerek* halmaza, ahol a kényszerek a terv lépéseinek egymásutániságát szabják meg.
- *Értékkényszerek* halmaza, ahol a kényszerek a terv lépéseiben (operátoraiban) szereplő változók értékeire tesznek megkötést.
- *Okozati kapcsolatok* halmaza, ahol a kapcsolatok a terv lépései közt fennálló ok-okozati kapcsolatokat írják le.

Legyen a terv lépéseinek halmaza *S*. Ekkor a terv valamely két $s_i,s_j\in S$ lépésre vonatkozó $\prec$ sorrendi reláció azt írja elő, hogy a terv majdani végrehajtásakor s_i meg kell, hogy előzze (nem feltétlen közvetlenül) s_j-t. Jelölje ezt a sorrendi kényszert: $s_i\prec s_j$.
Legyen V_t a *t* terv lépéseit leíró operátorokban szereplő változók halmaza. Az értékkényszerek ekkor valamely $V_t^i\in V_t$ változóra tehetnek megkötést oly módon, hogy vagy egy másik $V_t^j\in V_t$ változóval, vagy pedig egy konstanssal *egyesítik*.

Az okozati kapcsolatok a terv lépései közti ok-okozati összefüggéseket írják le. Ha $s_i,s_j\in S$ a *t* terv két olyan lépése, melyekre teljesül, hogy $\exists\, c\subseteq köv(s_i)$, melyre fennáll, hogy $c\subseteq elő(s_j)$, azaz

az s_i lépés következményeinek valamely c részhalmaza szükséges előfeltétele s_j-nek, akkor azt mondjuk, hogy „s_i okozati kapcsolatban áll s_j -vel. Jelölje ezt az okozati kapcsolatot: $s_i \xrightarrow{c} s_j$.

A terveket tervkészítő módszerek segítségével állítjuk elő. A tervkészítés történhet egyrészt a lehetséges szituációk terében. Ekkor a tervet úgy állítjuk elő, hogy keresünk egy a kezdeti és a cél-állapot (szituáció) közt vezető cselekvés-sorozatot. Ha a keresés a kezdeti állapot felöl indul, akkor *progresszív tervkészítésről,* ha a cél-állapot felöl, akkor pedig *regresszív tervkészítésről* beszélünk. Másrészt a tervkészítés történhet a tervek terében is. Ekkor egy *részleges tervvel* indulunk, amit addig „bővítünk", amíg egy kész tervet nem kapunk belőle. A részleges tervek módosítása *finomító* és *módosító operátorok* segítségével történhet. A finomító operátorokkal a terv kényszereit, míg a módosító operátorokkal a terv bármely más aspektusát módosíthatjuk. *Részben rendezett* az a terv, amelyben bizonyos lépések sorrendje nem kötött egymáshoz képest. A részben rendezett tervek előállítására alkalmas tervkészítő módszereket *részben rendezett tervkészítőknek* nevezzük, ellenben a *teljesen rendezett tervkészítőkkel,* amelyek csupán olyan tervek előállítására képesek, melyekben a lépések sorrendje egyértelműen rögzített. Ezeket a terveket *teljesen rendezett* terveknek nevezzük. A részben rendezett tervek lépéseinek teljes sorrendezését a terv *sorbarendezésének* nevezzük. Hasonlóan az olyan tervet, amelyben csak behelyettesített változók fordulnak elő, *teljesen specifikált tervnek* nevezzük. Végül pedig, azt a tervet, amely garantálja egy *tervkészítő ágens* számára a cél elérését, az ágens-környezet által reprezentált probléma *megoldásának* nevezzük.

A megoldás lehet teljesen, vagy akár csak részben rendezett. Az utóbbi esetben lehetőségünk van bizonyos lépések párhuzamosítására, illetve a végrehajtási sorrend megválasztására. A részben rendezett tervek azonban nem mindig felelnek meg elvárásainknak. A következő két megkötésnek kell teljesülnie rájuk (Russell és Norvig, 1995):

(M1) $\forall s_j \in S$ *lépés* $\forall c \in$ *elő*(s_j) *előzményéhez* $\exists s_i \in S$ *úgy, hogy* $c \in$ *köv*(s_i) *és* $\neg\exists s_k$ *amelyre* $(\neg c) \in$ *köv*(s_k) *és* $s_i \prec s_k \prec s_j$ *teljesülhet.*

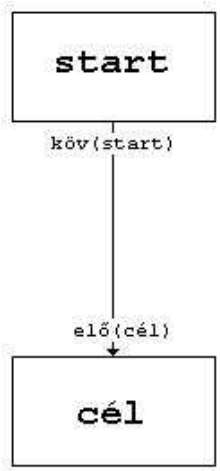
(M2) $\neg\exists s_{i_1}, s_{i_2}, \ldots, s_{i_N}$ *úgy, hogy* $s_{i_1} \prec s_{i_2} \prec \cdots \prec s_{i_{N-1}} \prec s_{i_N} \prec s_{i_1}$ *teljesül valamely* $N \in Z^+$ *esetén, továbbá a tervben szereplő semelyik* V *változóhoz sem tartozik egynél több különböző konstans értékkényszer.*

Az első feltétel (M1) a *teljességre* vonatkozik. E szerint minden lépés minden előfeltételéhez kell, hogy létezzen egy másik olyan lépés a tervben, amelynek az adott előfeltétel következménye. Továbbá nem létezhet olyan lépés, amely a végrehajtás során az előbbi két lépés közé „ékelődve" megszüntetheti az utóbbi megfelelő előfeltételét. A második feltétel (M2) a *konzisztenciára* vonatkozik. Eszerint se a terv sorrendi kényszereinek halmazában, se a változókra vonatkozó értékkényszerek (kötések) halmazában nem lehet ellentmondás.

Ha a fenti két feltétel teljesül egy tervre, akkor a terv megoldásnak tekinthető. A feltételek betartása pedig már a tervkészítő dolga.

2.2.2.1. RRT

Ebben a szakaszban a részben rendezett tervkészítés elvét vázoljuk röviden. A tervkészítés feladata, hogy a tervek terében egy teljes és konzisztens tervet, azaz egy megoldást találjon a problémára, amit kezdetben csupán csak egy *start* és egy *cél* lépés reprezentál. A *start* lépésnek nincsenek előfeltételei, csak következményei, amik a kiindulási állapotot állítják elő. A *cél* lépésnek pedig csak előfeltételei vannak, amik a kívánt cél-állapotot reprezentálják. Ezt nevezzük minimális részleges tervnek. Egy ilyen terv általános sémáját szemlélteti a 2.9. ábra.

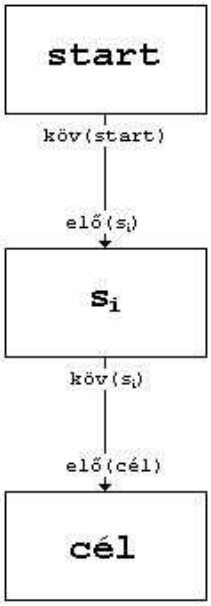


2.9.ábra. Minimális részleges terv általános alakja

A tervkészítés során addig-addig finomítjuk a tervet, addig-addig adunk hozzá újabb és újabb lépéseket, amíg a *start* következményeiből a lépések segítségével a *cél* összes előfeltételét elő nem állítjuk, vagy a zsákutcába nem jutunk. Ekkor visszalépve másként próbáljuk meg kiegészíteni a tervet. Ha minden módon zsákutcába jutunk, akkor feladjuk.

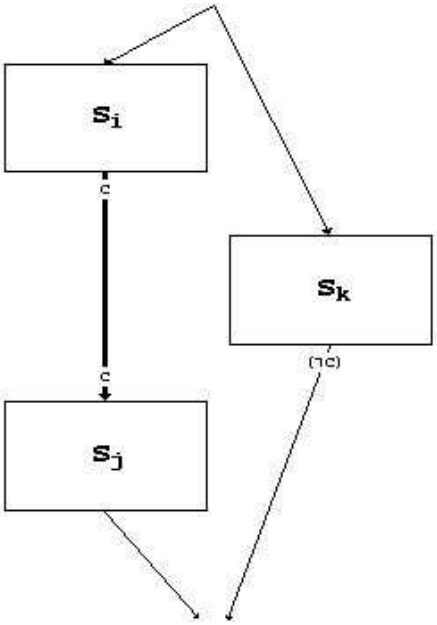
Most pedig lássuk egy regresszív alapon működő részben rendezett tervkészítő vázlatos működését! Legyen adott – a 2.9. ábrának megfelelően – egy probléma. A regresszív RRT algoritmus (röviden RRT) ezt tekinti kiindulási tervnek. Első lépésben megvizsgálja, hogy a terv megoldás-e. Ha igen, akkor készen van, ha nem, akkor kiválaszt egy olyan s_j lépést a tervből, amelynek valamely $c \in elő(s_j)$ előfeltétele nem teljesül. Ehhez az előfeltételhez keres egy olyan s_i lépést (akár a terv lépései közül, akár az operátorsémák halmazából), amelynek következményei közt szerepel a c előfeltétel, azaz amelyre $c \in köv(s_i)$. Ha s_i a terv lépései közül való, akkor a terv okozati kapcsolatainak halmazához hozzáadja az $s_i \xrightarrow{c} s_j$ kapcsolatot, a sorrendi kényszerek halmazát pedig kiegészíti egy $s_i \prec s_j$ kényszerrel. Ha s_i még nem volt része a terv lépéseinek, akkor hozzáveszi, és a sorrendi kényszerek halmazát egy $start \prec s_i \prec cél$ kényszerrel egészíti ki. A kezdeti tervből kiindulva a fenti lépések végrehajtásával a 2.10. ábrán látható állapothoz jutunk.

A 2.10. ábrán látható tervben még csak sorrendi kényszerek láthatók (ezeket jelölik a vékony nyilak). A későbbiek során, újabb lépések behozatalát követően a tervben már okozati kapcsolatok (vastag nyilak) is megjelenhetek. Egy ilyen helyzetet mutat be a 2.11. ábra.



2.10.ábra. A kezdeti terv egy lépéssel kiegészítve

Az ábrán egy olyan terv részlete látható, amelyben sérül az (M1) feltétel, hiszen s_k nem függ se s_i, se s_j lépésektől se sorrendileg, miközben a következményei közt szerepel egy olyan $(\neg c) \in$ *köv*(s_k) részállítás, amely *fenyegeti* az $s_i \xrightarrow{c} s_j$ okozati kapcsolatot. Ez azt jelenti, hogy s_k a végrehajtás során s_i és s_j lépések közé „ékelődve" megszüntetheti a $c \in$ *elő*(s_j) előfeltételt, amely szükséges feltétele az s_j lépés, s így az egész terv végrehajtásának. Az s_i és s_j közt fennálló okozati kapcsolatot ezúttal *védett kapcsolatnak* fogjuk nevezni. Az okozati kapcsolatok védelmét az RRT algoritmus az új lépések tervbevételét követően látja el. Teszi ezt úgy, hogy megvizsgálja az összes okozati kapcsolatot fenyegető lépés *előre*, vagy *hátramozdításának* lehetőségét.



2.11.ábra. Terv-részlet egy fenyegetett okozati kapcsolattal

Előremozdításnak nevezzük azt, ha a terv sorrendi kényszereihez az $s_j \prec s_k$ kényszert, hátramozdításnak pedig azt, ha az $s_k \prec s_i$ kényszert adjuk. Az új sorrendi kényszer csak akkor válik véglegessé, ha hozzáadása nem sérti az (M2) feltételt, azaz nem okoz inkonzisztenciát a sorrendi kényszerek közt. *Lehetséges fenyegetésnek* számít továbbá s_k az $s_i \xrightarrow{c} s_j$ védett kapcsolatra nézve, ha s_k változóinak halmaza nem teljesen behelyettesített, s így létezik olyan (akár részleges) behelyettesítésük, hogy $(\neg c) \in$ *köv*(s_k) teljesülhet. Ennek a veszélynek a kezelésére több módszer is ismeretes, melyekre most külön nem térünk ki. Az RRT algoritmus működését szemléltető pszeudokód a következő:

```
function RRT(start, cél∈ S, O) returns t∈ T {
      t := Kezdeti_terv(start, cél)
      while ¬Megoldás(t) do {
            [sk, c] := Kielégítetlen_részcél(t)
            t := Terv_kiegészítés(t,O,sk,c)
            t := Fenyegetés_feloldás(t)
      }
      return t
}
```

Tehát a kezdeti tervből kiindulva, az operátorok O halmazának felhasználásával addig módosítjuk a tervet, amíg vagy egy megoldáshoz, vagy pedig zsákutcába nem jutunk. Először tehát keresünk egy még kielégítetlen részcélt, azaz a terv egy olyan s_k lépését, amelynek egy c előfeltételét még nem teljesítettük. Ilyen mindenképp kell, hogy legyen, hiszen t-ről megállapítottuk, hogy még nem megoldás. A következőkben tehát ennek a c előfeltételnek a kielégítése lesz a célunk, amihez a terv kiegészítéséhez kell folyamodnunk. Ha a kiegészítés minden módon meghiúsul, akkor zsákutcába jutottunk. Ha sikerül, akkor továbblépve megpróbáljuk feloldani a lehetséges (keletkezett) fenyegetéseket. Ha a fenyegetések feloldása nem oldható meg az (M1) és (M2) feltételek betartásával, akkor visszalépve másként próbáljuk meg újra kiegészíteni a tervet. Ha nincs több lehetséges kiegészítés, vagy ha semelyik kiegészítés esetén sem tarthatók be az (M1) és (M2) feltételek, akkor zsákutcába jutottunk. Ha mégis sikerülne túljutnunk a fenyegetések feloldásán, akkor újra megvizsgáljuk, hogy a terv megoldás-e. Ha igen, akkor az a talált megoldással térünk vissza, ha nem, akkor újra kezdjük a kielégítetlen részcélok keresését.

Összefoglalva tehát azt mondhatjuk, hogy egy-állapotú problémák esetén az RRT megbízható, hatékony, helyes és teljes algoritmus. Sajnos azonban több-állapotú problémák esetén – az operátorok reprezentációjának „determinisztikus” jellege miatt – egy-egy tervbeli lépés többféle kimenetelének, vagy a lépések előfeltételtől függő különböző következményeinek a kezelése már nem megoldható. Az eshetőségi és a felderíthetőségi problémák általános kezelhetősége is kizárt, hiszen amellett, hogy esetükben is megjelenik a „nem-determinizmus”, még a környezet és a cselekvések kimenetelének és reprezentációjának a bizonytalansága is közrejátszik.

2.2.2.2. RRT-DUNF

Ezeket a hiányosságokat részben pótolja az RRT algoritmus RRT-DUNF (R*észben* R*endezett* T*ervkészítés* D*iszjunkcióval*, U*niverzális kvantorokkal*, N*egálással és* F*eltételes következményekkel*) elnevezésű kiterjesztése, amely az operátor-leírás kifejezőbbé tételével próbálja meg rugalmasabbá tenni az eljárást. Ezek szerint az operátorok előfeltételei és következményei tetszőleges, konjunkciót, diszjunkciót és negációt tartalmazó univerzálisan kvantifikált logikai kifejezéseket lehetnek, ahol a következmények közt feltételes kifejezések is szerepelhetnek. A feltételes kifejezések *„Q* **when** *P”* alakú logikai állítások, melyekben *Q* akkor tekinthető az adott operátor egy következményének, ha a végrehajtás során a(z) – előfeltételekre (változóiban) hivatkozó – *P* állítás kiértékelése igaznak bizonyul.

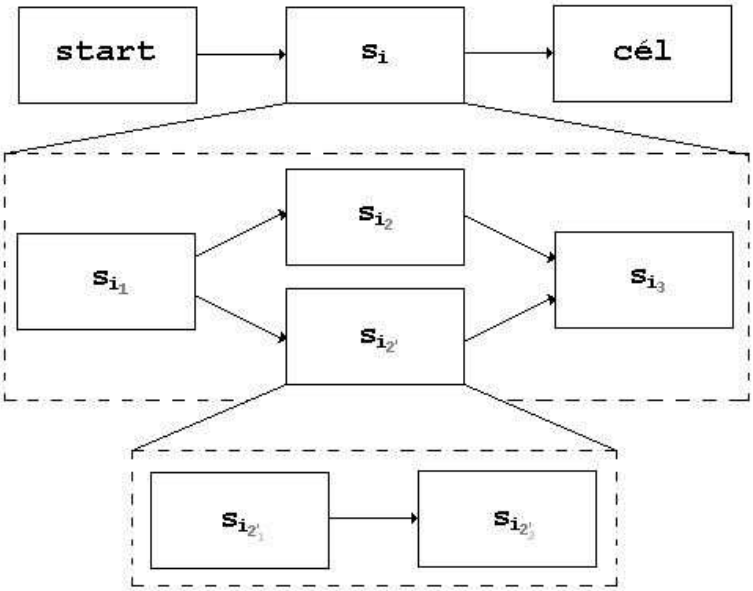
Az RRT-DUNF algoritmus a részletektől eltekintve gyakorlatilag megegyezik az RRT-vel. A különbség az algoritmus egyes lépéseinek megvalósításában, azaz a még kielégítetlen részcélok kiválasztásában, a terv kiegészítésének és a fenyegetések feloldásásának módjában rejlik. A kielégítetlen részcélok kiválasztásakor ügyelnünk kell arra, hogy a kiválasztott részcélhoz tartozó tervbeli lépés következményei közt szerepel-e *„k* **when** *c”* alakú feltételes állítás, ahol *c* a lépés kielégítetlen előfeltétele. A részcélt akkor érdemes kiválasztanunk, ha *k* egy már meglévő okozati kapcsolat védett feltétele, hiszen ekkor *c* teljesülése (közvetve) szükséges feltétele *k* teljesülésének, s így a terv végrehajthatóságának. Ha a lépés következményei közt szerepel egy *„* $\neg k$ **when** *c”* alakú feltételes rész-következmény, akkor *c* teljesülése lehetséges fenyegetést jelent minden $s_i \xrightarrow{k} s_j$ okozati kapcsolatra nézve. A fenyegetés úgy oldható fel, ha biztosítjuk, hogy *c* nem teljesül. Ez úgy oldható meg, ha a terv kiegészítésekor csak olyan lépésések közül választunk, melyek következményei közt szerepel a $\neg c$ állítás. Ezt nevezik *konfrontációnak.* A *diszjunktív előfeltételek* sem okoznak gondot, csupán csak több lehetőséget kínálnak a kielégítetlen részcélok kiválasztására, hiszen a diszjunktív kifejezés bármely elemének teljesítése már elég a részcél teljesüléséhez. A *diszjunktív következmények* kezelése ellenben már sokkal bonyolultabb, hiszen azok az adott lépés több lehetséges kimenetelét írják le. Legtöbbször minden kimenetelre külön-külön tervet kell készítenünk, ami igencsak megnöveli a tervkészítés számítási igényét, hiszen a lehetséges tervek száma a tervek hosszával exponenciális arányban nő. További költséget jelent az univerzális kvantifikáció bevezetése. Egy adott kifejezés változóira vonatkozó univerzális kvantifikáció megfelel a kifejezés összes behelyettesítése konjunkciójának. Ez – a „kifejtés” – mind az előfeltételekben, mind a következményekben rendkívül terebélyes logikai kifejezéseket eredményezhet. Továbbá megkötést kell tennünk a behelyettesítendő változók *típusára (típusolt argumentumok)* is, hiszen, ha a lehetséges behelyettesítések száma végtelen, akkor a konjunkció behelyettesített rész-állításainak az elemszáma sem lesz véges. Ehhez sajnos *statikus* környezetre van szükség, ahol a típusok véges száma már a kezdetek kezdetén adott s időben nem változik.

Összefoglalva elmondhatjuk, hogy bár az RRT-DUNF algoritmus a kifejezőbb leírásmódnak köszönhetően alkalmas lehet több-állapotú problémák megoldására, hatékonyságára

és általánosságára nézve nincs garancia. Komplex, dinamikus környezetek esetén nemigen használható. Eshetőségi és felderíthetőségi problémák megoldása szóba sem jöhet általa. A leírásmód kellő megválasztása azonban nagyságrendi javulást eredményezhet az RRT-hez képest.

2.2.2.3. HD-RRT

A hatékonyság növelésének egy másik módja, ha nem a lépések (operátorok) leírását, hanem a tervben betöltött szerepüket értelmezzük át. A HD-RRT (H*ierarchikus* D*ekompozíciós* R*észben* R*endezett* T*ervkészítő*) alapötlete szerint hatékonyabb a tervet rész-tervekre bontva megoldani. Ehhez bevezeti az *absztrakt* és *elemi* operátorok fogalmát. Elemi operátoroknak nevezzük az operátorok O halmazának azon $E \subseteq O$ részhalmazát, melyben $\forall o \in E$ operátorra teljesül, hogy a terv végrehajtásakor közvetlenül végrehajtható cselekvést reprezentál. Az operátorok maradék $O \backslash E$ halmazának elemeit absztrakt (vagy összetett) operátornak nevezzük. A fogalmat az indokolja, hogy az absztrakt operátorok – mivel önmagukban még nem végrehajthatók – tekinthetők az őket tartalmazó terv rész-terveinek. Ahhoz tehát, hogy a terv végrehajtható legyen, az absztrakt operátorokat valahogy végrehajtható lépésekké (azaz résztervekké) kell alakítanunk. A *hierarchikus dekompozíció* során erre a célra *felbontási módszerek (sémák)* szolgálnak, melyek az absztrakt lépésekhez egy-egy résztervet rendelnek. Természetesen a felbontás nyomán előálló résztervekben is szerepelhetnek absztrakt lépések is, melyeket később hasonlóan helyettesíthetünk az őket megvalósító résztervekkel. A 2.12. ábra egy ilyen esetre mutat példát:



2.12.ábra. Absztrakt lépések felbontása

Az ábrán egy – a tervben szereplő – összetett s_i lépés többszintű felbontása látható. A terv összetett lépéseit addig-addig bontjuk tovább a tervkészítés során, mígnem a kezdeti terv elemi lépések útján történő megvalósításához (azaz megoldáshoz) nem jutunk. A bemutatott gondolatmenetet a következő pszeudo-kód szemlélteti:

```
function HD-RRT(t, O, M) returns t∈T {
      while ¬Megoldás(t) do {
            [sk, c] := Kielégítetlen_részcél(t)
            t := Terv_kiegészítés(t,O,sk,c)
            si := Összetett_lépés_választás(t)
            t := Összetett_lépés_felbontás(t, M, si)
            t := Fenyegetés_feloldás(t)
      }
      return t
}
```

Több változtatás is történt az eddigi RRT algoritmushoz képest. Egyrészt megoldásnak immár csak az elemi lépésekből álló, (M1) és (M2) feltételeket kielégítő terveket tekintjük. Másrészt algoritmusunk most már nem csak egy *start-cél* lépésekből álló kezdeti tervet kaphat kiindulásként, hanem tetszőleges (akár kész) tervet is. Ennek előnye, hogy jobban „körvonalazható" a kívánt megoldás, több beleszólásunk van a végleges tervbe: részletesebben adhatjuk meg a kiindulást. Ezen felül két új művelettel is kiegészült az eljárás. A teljes algoritmus a következő:

Az eljárás három bemenetet kap: (1) a kezdeti t tervet, (2) az operátorok O halmazát és (3) a módszerek M halmazát, ahol a módszerek *felbontás(o,t')* alakú logikai állítások, melyek egy $o \in O$ absztrakt operátorhoz egy-egy olyan t' tervet rendelnek, amely az adott operátort valósítja meg.

Kezdetben megvizsgáljuk, hogy a kiindulási t terv megoldás-e. Ha még nem, akkor kell benne lennie egy még kielégítetlen c előfeltételt tartalmazó s_k lépésnek. Miután kielégítettük s_k előfeltételeit, megnézzük, hogy van-e még összetett s_i lépés a tervben. Ha van, akkor egy alkalmas (azaz s_i-re illeszkedő) felbontási módszer segítségével rész-lépésekre bontjuk, majd a t tervben ezekkel helyettesítjük. A terv meglévő értékkényszereihez hozzáadjuk a választott módszer értékkényszereit (változó-kötéseit). Minden t-beli $s_a \prec s_i$ sorrendi kényszert $s_a \prec s_m$ alakú sorrendi kényszerekkel helyettesítünk, ahol s_m a felbontás által adott részterv olyan lépése, melyhez nem létezik olyan s_j lépés a résztervben, hogy $s_m \prec s_j$ szerepelne a részterv sorrendi kényszerei közt. Magyarán s_a lépést a módszer „utolsó" lépései elé tesszük. Hasonlóan minden t-beli $s_i \prec s_z$ sorrendi kényszert $s_n \prec s_z$ alakú sorrendi kényszerekre cserélünk, ahol s_n a részterv olyan lépése, amely előtt más lépés már nem állhat, azaz amelyre a részterv sorrendi kényszerei közt nem létezik $s_{j'} \prec s_n$ alakú sorrendi kényszer. Ezen felül a t terv minden $s_a \xrightarrow{c} s_i$ okozati kapcsolatát cseréljük le olyan $s_a \xrightarrow{c} s_m$ alakú kapcsolatokra, ahol s_m a módszer által adott részterv olyan lépése, amely előtt nincs más olyan lépés a résztervben, melynek c előfeltétele. Hasonlóan minden $s_i \xrightarrow{c} s_z$ alakú t-beli okozati kapcsolatot helyettesítsünk $s_n \xrightarrow{c} s_z$ okozati kapcsolatok egy olyan halmazával, melyekben s_n a részterv olyan lépése, amely után már nincs olyan lépése a résztervnek, melynek c következménye. Ha az új értékkényszerek, sorrendi kényszerek vagy okozati kapcsolatok ellentmondást okoznak, akkor visszalépünk és másként próbáljuk meg felbontani s_i-t. Ha ez nem oldható meg, akkor zsákutcába jutott a tervkészítés. Ha sikerül, akkor

megpróbáljuk feloldani a kapott tervben esetlegesen kialakult fenyegetéseket. Ha ez nem lehetséges, akkor visszalépünk és egy másik összetett lépést próbálunk meg ellentmondásmentesen felbontani. Ha nem találunk ilyet, akkor a tervkészítés végérvényesen zsákutcába jutott. Ha találunk, akkor újra megvizsgáljuk, hogy tervünk megoldás-e. Ha még mindig nem, akkor újra kezdjük a kielégítetlen részcélok kiválasztásánál, ellenben a kapott tervvel térünk vissza.

A hierarchikus dekompozíció elvét szinte minden ma ismert tervkészítő-rendszer alkalmazza kisebb-nagyobb változtatásokkal. Az alapelv egy analitikus, bizonyítottan helyes és teljes eljárást ecsetel. Alapesetben a HD-RRT képességei az RRT képességeivel azonosak, azaz az operátorok leírónyelvének kiterjesztése nélkül csak egyállapotú problémák megoldására alkalmas. Bármily meglepően is hangzik, általános esetben hatékonyság tekintetében sem jobb az RRT-nél. Ellenben – heurisztikák alkalmazásával – nagyságrendekkel hatékonyabbá tehető. Sajnos azonban ekkor már nem sokat mondhatunk a módszer képességeiről általánosságban. A heurisztikák közül talán a legismertebb a *lefelé* és a *felfelé megoldhatósági* tulajdonság. Az előbbi azt írja elő, hogy minden absztrakt t megoldáshoz kell, hogy létezzen olyan elemi megoldás, melynek t egy absztrakciója. Az utóbbi arról beszél, hogy elemi megoldásoknak nem létezik inkonzisztens absztrakciója. A felfelé megoldhatóság elégséges feltétele lehet az úgynevezett *egyedi elsődleges részcselekvés* betartása, amely szerint az absztrakt lépés felbontásában kell, hogy legyen legalább egy olyan lépés, melynek előfeltételei között szerepel az absztrakt lépés minden előfeltétele és következményei közt az összes következménye. Ezek szerint, ha egy terv egy adott felbontása inkonzisztens, akkor abból már nem lehet további felbontások útján megoldást produkálni, illetve, ha konzisztens, akkor biztosan létezik egy – további felbontás útján előálló – megoldás. Sajnos mindkét feltétel teljesülésére létezik ellenpélda, így általános esetben sosem lehetünk biztosak abban, hogy egy inkonzisztens absztrakt terv elvetése egy konzisztens tervvel szemben jó-e, vagy sem, hacsak nem ismerjük kellő bizonyossággal cselekvéseink működését. Bonyolultabb (pl. eshetőségi, vagy felderíthetőségi) környezetben azonban erre nincs lehetőségünk, s így ekkor e heurisztikák alkalmazása sem jöhet szóba. Léteznek természetesen más egyéb heurisztikák is, amelyekkel szemben többnyire hasonló ellenvetéseink lehetnek. Komplex környezetek esetén az eljárás rugalmasságát az a tény is nehezítheti, hogy a tervkészítés módszereit (sémáit) készen kapja. Ezek ugyebár nem feltétlen felelnek meg a környezet elvárásainak, s így az általuk létrehozható terv sem feltétlen megfelelő.

2.2.2.4. FRRT

Ha tehát a környezet nem statikus és hozzáférhető, akkor az előzőleg bemutatott tervkészítési módszerek egytől-egyig csődöt mondanak. A probléma abban áll, hogy eshetőségi problémák esetén a cselekvések kimenetele, illetve a környező világ állapota nem jósolható meg. Eddig például megállapodás szerint úgy tekintettük, hogy azon logikai állítások, melyek nem szerepelnek egy-egy tervbeli lépés előfeltételei közt, mindenképp hamisak. Ez nem feltétlen igaz

akkor, ha a környezet állapotára vonatkozó ismereteink hiányosak, s így az adott állításról nem áll módunkban nyilatkozni, avagy az állítás igazságtartalmára – cselekvéseink következményei alapján – nem tudunk biztosan következtetni. Elképzelhető tehát az is (jellemzően valós környezetek esetén), hogy nincs is olyan cselekvésünk, amely az adott – környezeti állapotra vonatkozó – állítást (mint előfeltételt) képes volna bizton előállítani.

Ennek megoldására született meg az FRRT (F*eltételes* R*észben* R*endezett* T*ervkészítés*) elve, mely szerint a terv végrehajtása során előforduló eshetőségekre előre felkészülhetünk, ha a tervkészítés során a tervbe feltételes elágazásokat, ún. *érzékelő cselekvéseket* iktatunk. Az érzékelő cselekvések döntik el, hogy a terv végrehajtásakor a lépések milyen *kontextusa*, azaz a környezet milyen állapota érvényes. Így, ha – a cselekvésekre és a környezetre vonatkozó megfelelő modellek birtokában – minden eshetőséget számításba tudunk venni, minden lehetséges kontextushoz külön-külön tervet készíthetünk, imígyen oldva meg az eshetőségi problémát. A tervkészítés ezen megközelítését szokás *feltételes*, avagy *eshetőségi tervkészítésnek* is nevezni.

A feltételes lépések IF-THEN-ELSE alakú adatstruktúrák, ahol a feltétel-rész egy érzékelő cselekvést valósít meg, a feltételes következmények pedig akár egész feltételes tervek is lehetnek. Ha a feltétel-rész által megvalósított érzékelő cselekvésnek előfeltételei is vannak, akkor azok előállítására külön (akár feltételes) tervet kell készítenünk. Ebben az esetben e terv képezi majd a feltételes lépés feltétel-részét.

Az FRRT algoritmus váza nagyjából megegyezik az RRT-ével. A főbb különbségek: (1) az értékkényszereken, sorrendi kényszereken és okozati kapcsolatokon túl immár a *feltételes kapcsolatok* is részei a tervnek. A feltételes kapcsolatok érzékelő lépéseket kapcsolnak össze a terv azon lépéseivel, melyek kontextusának egy része az adott érzékelésből származik. (2) az egyes tervbeli lépések kontextusa átöröklődik a későbbi lépésekre. (3) így a tervnek nem csak egy cél lépése lehet, hanem annyi, ahány különböző kontextus előállhat a tervben szereplő érzékelések nyomán: minden eshetőségre külön tervet dolgozunk ki. (4) egy s_f lépés csak akkor jelent fenyegetést egy $s_i \xrightarrow{c} s_j$ védett kapcsolatra nézve, ha s_f és s_j kontextusai azonosak. Ezért a tervkészítés során a fenyegetések feloldását nem csak s_f előre, vagy hátramozdításával érhetjük el, hanem úgy is, hogy az s_j lépéssel feltételes kapcsolatban álló egyik s_{felt} megfigyelés ellenkező kimenetelét rendeljük hozzá s_f-hez feltételként, így különböztetve meg a két lépés kontextusát. Ezt nevezik *kondicionálásnak.* Az FRRT pszeudo-kódja a következő:

```
function FRRT(a0, C, O) returns t∈ T {
      t := Kezdeti_terv(a0, C)
      while ¬Befejezés(t) do {
            C := Alternatív_célkörnyezet(t, C)
            [sk, c] := Kielégítetlen_részcél(t)
            t := Terv_kiegészítés(t,O,sk,c)
            t := Fenyegetés_feloldás(t)
      }
      return t}
```

A vázolt algoritmus a hasonlóságok ellenére lényegileg különbözik az RRT-től. Az eljárás bemenetére három paraméter érkezik. O az operátorok halmaza, C a kezdeti cél-konfigurációk (különböző kontextusú, ámde azonos előfeltételű cél-lépések) legalább egyelemű halmaza, míg a_0 a környezet érzékeléséből származó kezdeti állapot (logikai) leírása. A kezdeti állapot és a célok felhasználásával előállítjuk a kezdeti t tervet. Ezt befejezetnek tekintjük, ha a terv megoldás, azaz nincsenek benne kielégítetlen előfeltételek, teljes és konzisztens, továbbá a benne szereplő érzékelések minden kimeneteléhez létezik megfelelő kontextusú cél-lépés, azaz minden lehetséges cél-környezetét számításba veszünk. Ha t még nem befejezett, akkor előállítunk t-ben egy újabb cél-lépést, melynek kontextusa a meglévő cél-lépések kontextusai diszjunkciójának negáltja, azaz, amelynek kontextusa minden eddigi cél-kontextustól különböző. Ezt követően – az RRT-nél látottaknak megfelelően – keresünk egy még kielégítetlen előfeltételű tervbeli lépést, kielégítjük a terv kiegészítése által, majd végül feloldjuk a keletkezett fenyegetéseket. A fenyegetések feloldásánál immár – az előre és hátramozdításon túl – a kondicionálást is alkalmazzuk. A kondicionált lépések kontextusát tovább-terjesztjük a tervben egészen a nekik megfelelő célokig. Ha bárhol inkonzisztenciát tapasztalunk, akkor – az előző algoritmusok leírásánál látottaknak megfelelően – visszalépünk, és addig próbálkozunk, mígnem vagy kifutunk az összes lehetőségből, vagy egy befejezett tervhez nem jutunk. Ebben az esetben a kapott tervvel térünk vissza.

Összefoglalva, a feltételes tervkészítés elve alkalmas lehet eshetőségi problémák megoldására. A fentebb vázolt elvnek természetesen még több – hatékonyságot és rugalmasságot növelő – kiegészítése létezik. Ezek közül említésre szorul a *paraméterezett terv* fogalma, amelyben olyan megfigyelések is szerepelhetnek, melyek értéke egészen a terv végrehajtásáig ismeretlen marad. Ezen paramétereket *futásidejű változók* formájában követjük nyomon, amik tehát csak a terv végrehajtásakor nyernek behelyettesítést. A ciklusokat is tartalmazó feltételes cselekvések tekinthetők programoknak, melyek előállítására az *automatikus programozás* módszertana szolgál. Sajnos azonban – ha nem zárjuk ki a rekurziók keletkezését – az algoritmus befejeződésére nem tudunk általános esetben garanciát adni. Látható, hogy – ha megengedjük az egyes érzékelő lépések többszöri felhasználását is a tervben – a feltételes cselekvések feltétel részében könnyen előállhat az a helyzet, hogy a feltétel-részt megvalósító tervben ugyanaz a feltétel önmagára hivatkozik (ördögi kör). A rekurziók kiküszöbölése, továbbá az összes eshetőség (cél-kontextus) figyelembevétele a tervkészítés során igencsak megnövelheti az eljárás erőforrásigényeit. A felderíthetőségi problémák – legalább közelítő – kezelése nem megoldott. A cselekvések, környezeti állapotok és érzékelések modelljeit sémák formájában továbbra is meg kell adnunk a tervkészítő számára.

2.2.3. Tervkészítő Ágensek

A tervkészítés és az ágensek elveinek egyesítésével olyan konkrét ágens-architektúrák hozhatók létre, melyek döntési mechanizmusa a tervkészítésen alapszik. Ebben a szakaszban az előző két szakaszra támaszkodva röviden bemutatjuk az így keletkezett, ún. *tervkészítő ágenseket.* A

tervkészítő ágensek a cél-orientált absztrakt ágens-architektúrák osztályába sorolhatók, ahol a *Cselekvés-Előállítás* függvény (lásd. 2.1.1.3) helyére most konkrétan egy-egy tervkészítő algoritmus kerül. A következőkben szót ejtünk majd az egyszerű, feltételes, újratervező és beágyazott tervkészítő ágensek főbb sajátosságairól.

2.2.3.1. Egyszerű Tervkészítő Ágens

Az egyszerű tervkészítő ágens a cél-orientált ágensek tervkészítő változatának a legegyszerűbb példánya. A tervkészítéshez egy olyan ideális tervkészítő algoritmust (pl. RRT – lásd. 2.2.2.1) használ, mely a cselekvések, célok és környezeti állapotok teljes ismeretében képes számára előállítani egy-egy – a cél garantált elérésére – alkalmas tervet. Az egyszerű tervkészítő ágens, miután érzékelte környezetét, a kapott érzet és a tudásbázisban tárolt információk alapján kijelöl egy célt. Ha a cél már a kezdeti állapotban teljesül, akkor egy üres tervet generál. Ha a cél elérhetetlen, akkor elveti, s új célt keres. A választott cél elérésére előállít egy „ideális" tervet, melyet aztán lépésről-lépésre – mondhatni „csukott szemmel" – hajt végre. Sajnos ez a fajta megközelítés csak egyállapotú környezetekben lehet eredményes, ahol a cselekvések kimenetele és a környezet állapota teljes egészében ismert és megjósolható. Ekkor az ágens számára elég csupán csak egy-egy terv végrehajtását követően megvizsgálni a környezetét, új célt és új tervet keresnie.

2.2.3.2. Feltételes Tervkészítő Ágens

A feltételes tervkészítő ágens működése csak annyiban különbözik az egyszerű tervkészítő ágensétől, hogy terveiben már feltétel-vizsgálatok is szerepelhetnek (lásd. FRRT – 2.2.2.4), melyek lehetővé teszik számára, hogy az egyszerű tervkészítő ágens által elhanyagolt tényezők figyelembevételével hatékonyabbá tegye döntéseit. Bizonyos célok eléréséhez egyenesen szükséges lehet az érzékelő cselekvések tervbe iktatása, hiszen általuk figyelembe vehetők a tervkészítés során nem hozzáférhető környezeti jelenségek hatásai: az eshetőségek. A feltételes tervkészítő ágens tehát alkalmas lehet eshetőségi környezetekben való boldogulásra.

2.2.3.3. Egyszerű Újratervező Ágens

A feltételes tervkészítő ágens hiányossága, hogy – az egyszerű tervkészítő ágenshez hasonlóan – az elkészített terveket – az elejétől a végéig – egyben szeretné végrehajtani. Ámde a terv végrehajtásakor előfordulhatnak olyan eshetőségek is, melyekre a terv készítésekor nem tudtunk felkészülni, melyeket a tervben nem vettünk figyelembe. Ha a lehetséges eshetőségeket, azaz a cselekvések váratlan következményeit az ágens fel tudja sorolni, akkor *korlátos határozatlanságról,* ha nem, akkor *korlátlan határozatlanságról* beszélünk. Az utóbbi esetben a tervek végrehajtása menet közben meghiúsulhat, s így új tervre lehet szükség.

Az *egyszerű újratervező ágens* erre a problémára kínál megoldást azáltal, hogy a terv végrehajtása során – lévén folyamatos kölcsönhatásban van a környezetével – képes – a terv

megfelelő végrehajtását felügyelve – szükség esetén módosítani. A felügyelet két módját különböztetjük meg: *végrehajtás felügyeletnek* nevezzük a jelen, vagy előző végrehajtott tervbeli lépésekből a jelen, vagy elkövetkező tervbeli lépésekbe vezető okozati kapcsolatok védett feltételeinek felügyeletét. *Cselevés felügyeletnek* nevezzük a pillanatnyilag végrehajtott lépés előfeltételeinek felügyeletét. Ha a megfigyelt környezeti állapot sérti valamely felügyelt feltételt, akkor szükség van az újratervezésre. Az újratervezés elve igen egyszerű: az ágens a végrehajtás során folyamatosan figyelemmel kíséri mind a teljes *t* tervet, mind pedig a hátralévő *h* résztervet. Ha a felügyelet jelzi, hogy *h* feltételei sérülnek, akkor kiválasztja a *t* terv valamely pontjából kiinduló azon h' résztervet, amely az aktuális állapotból a leghamarabb állítja elő a *t* terv cél-állapotát. Ezt követően hozzáfűz h' elejéhez egy – h' szükséges feltételeinek előállítására szolgáló – résztervet, s az így kapott tervet tekinti a hátramaradó *h*-nak.

Látható, hogy – a feltételes tervkészítő ágenssel ellenben – az egyszerű újratervező ágens tervei nem minden eshetőséget vesznek figyelembe, hanem csak azokat, amik valóban bekövetkeznek a végrehajtás során. Ez legtöbbször gazdaságosabb, a környezet változásaihoz jobban igazodó, ámde kevésbé robusztus terveket eredményez.

2.2.3.4. Beágyazott Tervkészítő Ágens

Az előzőekben ismertetett ágensek mindegyikénél élesen különvált egymástól a tervkészítés és a terv végrehajtásának folyamata. A *beágyazott tervkészítő ágensben* ellenben már egy egységet képez. „Az ágensre úgy tekinthetünk, mintha minden pillanatban egy terv (életének nagy terve) végrehajtásán dolgozna”. (Russell és Norvig, 1995) A szokásos – tervekkel kapcsolatos – tényezőkön túl ezért a beágyazott tervkészítő ágensnek folyamatosan figyelnie kell a környezetét, és megfigyelései alapján (1) finomítania a tervet, ha újabb információk birtokába jut; (2) kijavítania, ha váratlan környezeti hatások érik, és (3) kihagynia a menet közben feleslegessé vált lépéseket.

Kezdjük a másodikkal! Az aktuálisan érzékelt környezet alapján frissítjük tervünk *start* lépésének következményeit. Mivel az ágens működése során mindvégig úgy tekinti, mintha a *start* lépés következményei által leírt környezeti állapotban volna, ezért a tervek végrehajtása során folyamatosan frissíti őket. Ha tehát (2) a környezeti állapot váratlanul úgy módosul, hogy létezik olyan $start \xrightarrow{c} s$ okozati kapcsolat, melynek *c* feltétele már nem teljesül a *start* lépés következményei közt, töröljük a kapcsolatot a tervből. Ha (1) létezik olyan $s_j \xrightarrow{c} s_k$ okozati kapcsolat a tervben, melynek *c* feltételét egy korábbi $s_i \prec s_j$ lépés is előállítja, akkor cseréljük le ezt egy olyan $s_i \xrightarrow{c} s_k$ kapcsolatra, melyre nem vonatkozik semmiféle fenyegetés sem. Ezt nevezzük a kapcsolat *kiterjesztésének.* Végül pedig (3) töröljük a terv azon s_j lépéseit, melyek nem támasztanak alá egy okozati kapcsolatot sem. Ezeket a lépéseket *redundáns lépéseknek* nevezzük. Az így kapott *t* terv azon *s* lépését hajtjuk végre, melynek minden előfeltételét kielégíti a *start* lépés, kettejük közt nincsen más lépés és az *s* lépés nem fenyegeti a *t* terv semelyik okozati kapcsolatát

sem. A lépést a végrehajtás után minden összefüggésével együtt töröljük a tervből feltéve, hogy végrehajtása a következő észlelés beérkezte előtt befejeződik.

Összefoglalva tehát a tervkészítés és az ágensek egyesítése által végeredményben olyan tervkészítő-ágenshez jutottunk, mely képes lehet eshetőségi, vagy akár felderíthetőségi problémák megoldására is abban az esetben, ha képes a cselekvéseinek és a környezetének a modelljét tanulás útján, helyesen elsajátítani. Sajnos azonban az előálló tervek (megoldások, lásd. 2.2.2) „optimalitására" (lásd. 3.1) már semmilyen garancia sincsen.

2.3. EVOLÚCIÓS SZÁMÍTÁSI MODELLEK

Az optimális megoldás keresésére számos módszer ismeretes, melyek közt a természetes evolúció (Darwin, 1859) számítógépes modellje rangos helyet foglal el. Az evolúciós számítási modelleknek három fajtáját szokták megkülönböztetni aszerint, hogy milyen absztrakciós szinten interpretálják a darwini posztulátumot, miszerint a közös erőforrásokért versengő populáció azon egyedei kerülnek előnyösebb helyzetbe, melyek a versenyben előnyt jelentő tulajdonságokat hordoznak. A legabsztraktabb megközelítés *evolúciós programozás* (Fogel, Owens és Walsh, 1966; Fogel, 1995) néven vált ismertté, és a *fajok* közti versengést modellezi. Köztes absztrakciós szinten mozognak az *evolúciós stratégiák* (Schwefel, 1995), melyek az *egyed* szintjén vizsgálják a természetes kiválasztódás darwini folyamatát. A legelemibb megközelítésnek a *genetikus algoritmusok* (Goldberg, 1989) családjába tartozó módszerek tekinthetők, melyek a *gének* közti versengés szintjén követik nyomon az evolúció folyamatát. Ebben a szakaszban ez utóbbi módszerek főbb változatai közül említünk meg néhányat, a teljesség igénye nélkül.

2.3.1. Genetikus Algoritmusok (GA)

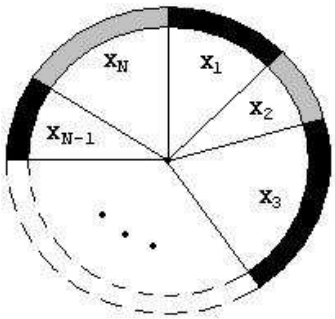
A genetikus algoritmusok – és változatai – *globálisan konvergens*, sztochasztikus keresőeljárások (Holland, 1975). Ez azt jelenti, hogy – a véletlenszerűség felhasználásával – bizonyítottan tartanak a keresési tér – jósági mérce által definiált – *globális optimumához.*

A genetikus algoritmusok esetében az egyedeket (alapesetben) bináris bit-füzérek (kromoszómák) reprezentálják, melyek pozíciói (alléljai) felelnek meg a kromoszóma génjeinek. Az algoritmus lényege, hogy egyedek egy kezdeti, véletlenszerűen előállított populációját – úgynevezett *genetikus operátorok* felhasználásával – addig-addig módosítja, mígnem a populáció megfelelő jellemzői (pl. legjobb egyed, egyedek átlagos jósága) eleget nem tesznek egy választott leállási kritériumnak. Az egyedek jóságát megbecslő jósági mércét szokás *fitness függvénynek* is nevezni. A fitness függvény a populáció egyedeihez rendel egy-egy valós számértéket, mely az adott egyed globális (a populáció viszonylatában vett) jóságát szimbolizálja. Alapvetően a következő – a gének manipulációjára hivatott – genetikus operátorokat különböztetjük meg: a *mutációs operátor* egy-egy választott bináris bitfüzér (kromoszóma) egy-egy véletlenszerűen választott bitpozíciójának (alléljának) megfelelő bit (gén) értékét módosítja (bináris esetben

invertálja). A *keresztezés* két bitfüzér valamely véletlenszerűen választott pozíciója után következő részeinek megcserélése. A *szelekció* a populáció egyedeinek jóság szerinti (véletlenszerű) kiválasztása. Mivel mindegyik operátor a véletlenhez folyamodik, ezért használatukhoz is egy-egy valószínűség tartozik (pl. mutációs valószínűség), amely azt szabja meg, hogy az adott operátort milyen valószínűséggel hajthatjuk végre egy-egy kiválasztott egyeden.

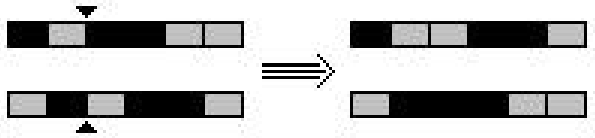
Az algoritmus menete ekkor a következő: Kezdetben előállítjuk egyedek egy véletlenszerű *populációjat*. Az egyedek hossza, a populáció mérete, és az operátorok valószínűsége adott. Ezt követően minden egyedhez rendelünk egy-egy jósági értéket, melynek alapján kiválasztunk közülük egy – a szelekció típusától függő – mennyiséget. A kiválasztott egyedeket párokba rendezzük. A párokat (szülőket) keresztezve és mutálva kapjuk a következő *generáció* egyedeit (gyerekeket). Így haladunk generációról generációra, mígnem a leállási kritérium nem teljesül.

A genetikus operátorok megvalósítására több lehetőségünk is van. Ezek közül a fontosabbak: A szelekció *rulett kerék* elven működik, ha az egyedeket jósági értékükkel arányos valószínűséggel választjuk ki. Legyen a P populáció $x_i \in P$ egyedeinek jósága $f(x_i)$. Ha a populáció N számú egyedből áll, akkor osszunk fel egy egységnyi kerületű kört N darab $\frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$ ívhosszú körcikkre, majd „perdítsük meg a kereket”, azaz állítsunk elő egy *[0,1)* tartományba eső, valós véletlen-számot. Ahol a „golyó megáll”, azaz amely egyednek megfelelő körcikk íve által meghatározott *[0,1)*-beli zárt rész-intervallumba esik az előállított véletlen-szám, azon egyed kerül kiválasztásra. A 2.13. ábra szemlélteti a rulett-kerék elvet.



2.13.ábra. Szelekció rulett-kerék elve

A *mérkőzés* alapú szelekció is rulett-kerék elven működik: K-szor „megpörgetjük a rulett-kereket”, majd az így kiválasztott K egyed legjobbjával térünk vissza. Az új generáció előállításához ezt ismételjük N-szer. A *felső százalék* elve szerint a populáció egyedeinek legjobb N százalékából választjuk ki véletlenszerűen a következő generáció egyedeit. A *valahány legjobb* kiválasztás lényege, hogy az egyedeket jóság szerint sorba rendezi, és az első valahány legjobbat választja ki közülük. – A keresztezésnek is több módja lehetséges. Létezik egy-pontú és több-pontú kereszteződés. Az egy-pontú kereszteződésről már szót ejtettünk az előzőekben (lásd. GA leírása). A több-pontú kereszteződés az egy-pontú kereszteződés értelemszerű kiterjesztése. Egy-pontú kereszteződésre mutat példát a 2.14. ábra.



2.14.ábra. Egy-pontú kereszteződés

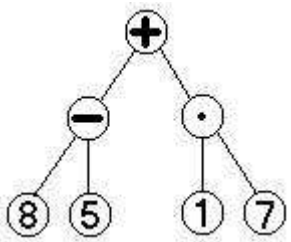
A mutáció az egyed valamely – véletlenszerűen választott – alléljának génjét változtatja meg. Bináris esetben ez az adott pozícióban lévő bit inverziójának felel meg.

Összefoglalva, a genetikus algoritmusok leírása igen természetszerű, jól kezelhető módszert ad a természetes evolúció gén-szintű, számítógépes megvalósítására. A fentebb vázolt egyszerű genetikus algoritmus robusztus, globálisan konvergens, optimális heurisztikus kereső-eljárás. Az algoritmusnak több változtatása is létezik, melyek többnyire a gén-leírás módját, az operátorok megvalósítását és az eljárás adaptivitását finomítják, bővítik tovább.

2.3.2. Genetikus Programozás (GP)

Míg a genetikus algoritmusok esetében az egyedeket bit-füzérek reprezentálják, addig a genetikus programozás (Cramer, 1985; Fujiko és Dickson, 1987; Koza, 1992) esetében ezek már „programok”. Ilyen szempontból tehát a genetikus programozást a genetikus algoritmusok speciális esetének lehet tekinteni, ahol az egyedek leírása a „programok halmazára” szűkül.

Ennek a változtatásnak igen jelentős következményei vannak: egyrészt át kell értékelnünk a jósági függvény fogalmát, másrészt újra kell definiálnunk a genetikus operátorokat, magyarán újra kell fogalmaznunk szinte az egész eljárást. Először is lássuk, hogy mit értünk egyedek alatt: a genetikus programozás egyedei programok, melyeket *program-fák* reprezentálnak. A program-fák benső csúcsai *funkciók*, míg levelei *terminálisok*. Jelölje a funkciók halmazát *F,* míg a terminálisokét *T.* A *lezárási elv* szerint minden funkció terminálisból terminálisba képez, azaz $\forall f \in F$ -re $f : T \rightarrow T$. Ez a programozási szempontból „szorosnak” tekinthető megkötés evolúciós szempontból függetleníti a genetikus operátorok használatát az adott egyed sajátosságaitól, így hát előnyös. Példának okáért legyen a funkciók *F* halmaza *F={+, -, *},* míg a terminálisok *T* halmaza az egész számok halmaza. Ekkor egy lehetséges program-fa lehet a következő:

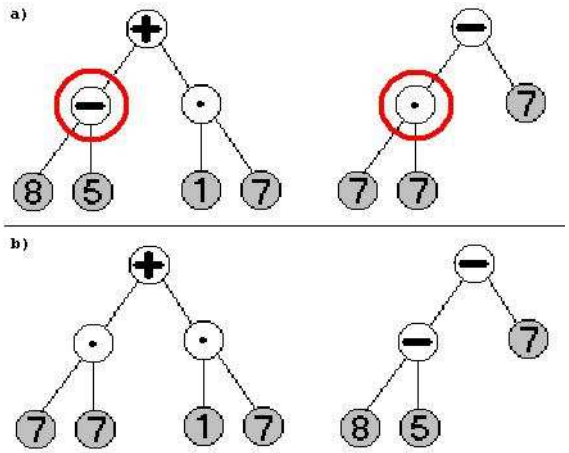


2.15.ábra. Példa a genetikus programozás egyedeire

A jósági függvénynek immár a 2.15.ábrán látottakhoz hasonló programokat kell kiértékelnie ahhoz, hogy megbecsülje jóságukat. A program kiértékelése a program futtatása által

történik. A program által reprezentált egyed jóságát ekkor a futás eredménye, a futás és a program-fa jellemzői befolyásolják. A 2.15.ábrán bemutatott egyed program-kódja prefix-jelöléssel *+(-(8,5),*(1,7),* míg infix-jelöléssel *((8-5)+(1*7)),* aminek kiértékelése értelemszerűen *10*-et ad eredményül. Látható, hogy a fák kiértékelése *„lentről-felfelé”* haladva történik.

Az algoritmus alapelve tehát a következő: Hasonlóan a genetikus algoritmusokhoz, generálunk egy véletlenszerű kiindulási populációt. Az egyedek száma, a populáció mérete, a leállási feltétel, a genetikus operációk valószínűségei, a funkciók és a terminálisok halmaza, továbbá a programok *maximális mélysége* adott. A szelekció – a jósági függvény fentebb leírt működésétől eltekintve – a genetikus algoritmusoknál látottaknak megfelelően működik. Hasonlóképp a kereszteződés, eltekintve attól, hogy most az egyeden belüli „véletlenszerűen kiválasztott pont” a program-fa valamely csúcsa, továbbá a két keresztezett egyeden belül a pont helye akár különböző is lehet, hiszen a fák is más alakúak. A 2.16.ábra a kereszteződésre mutat egy példát,



2.16.ábra. Példa a genetikus programozás egyedeinek keresztezésére

Az ábra a) részén látható a két kiválasztott program, illetve azon belül pirossal megjelölve a kereszteződési pontoknak megfelelő csúcsok. Ekkor a keresztezés során kicseréljük a kiválasztott csúcsokon függő részfákat. Az így kapott egyedeket az ábra b) része szemlélteti. Hasonlóan működik a mutáció is: ekkor a kiválasztott egyed valamely véletlenszerűen kijelölt csúcsán függő részfát módosítjuk. A továbbiakban az algoritmus működése megegyezik a genetikus algoritmussal.

A genetikus programozás, minthogy programokat evolvál, legtöbbször nagyságrendekkel hatékonyabbnak bizonyul a klasszikus genetikus algoritmusoknál. Vannak azonban olyan esetek is, amikor a genetikus programozás marad alul a megoldás jóságát, illetve megtalálásának idejét tekintve. Ennek okai főként a programokon végrehajtott genetikus operátorok („természet-ellenes”) sajátosságaira, illetve a programokra vonatkozó megkötésekre vezethetők vissza.

2.3.3. Génkifejező Programozás (GEP)

Az előbbi két módszer előnyeit ötvözi a *génkifejező programozás* (Ferreira, 2001) korszerű elve. Az elgondolás több fontos szempontból is „természetesebb” az előzőeknél. E szempontok megértéséhez először is be kell vezetnünk a *genotípus* és a *fenotípus* – biológiai eredetű – fogalmát. A genotípus az egyed génállománya. A fenotípus pedig a genállomány által „kódolt” tulajdonságok megnyilvánulása a környezet felé. A genotípus és a fenotípus közti – többnyire nem-lineáris jellegű – „leképzést” *episztázisnak* nevezzük. A genetikus algoritmus és a genetikus programozás nem tesz különbséget genotípus és fenotípus közt. Ellenben a génkifejező programozásnál már megjelenik az episztázis – természetszerű – jelensége a „gén-kifejezés” formájában.

Az egyedeket kromoszómák reprezentálják, melyek gének láncolataként jönnek létre, ahol a gének – a genetikus programozásnál megismert terminológia szerint – funkciókból és terminálisokból álló *füzérek*. Egy-egy gén igen egyszerűen egy-egy – genetikus programozásban megismert – program-fát „kódol” (reprezentál), ahol a kódolás során csupán csak a program-fa szintjeit kell rendre egymás után fűznünk ahhoz, hogy a megfelelő gént megkapjuk. Például a 2.15.ábrán látott program-fának megfelelő gén a következő:

+	−	·	8	5	1	7

2.17.ábra. A 2.15. ábra program-fájának megfelelő gén

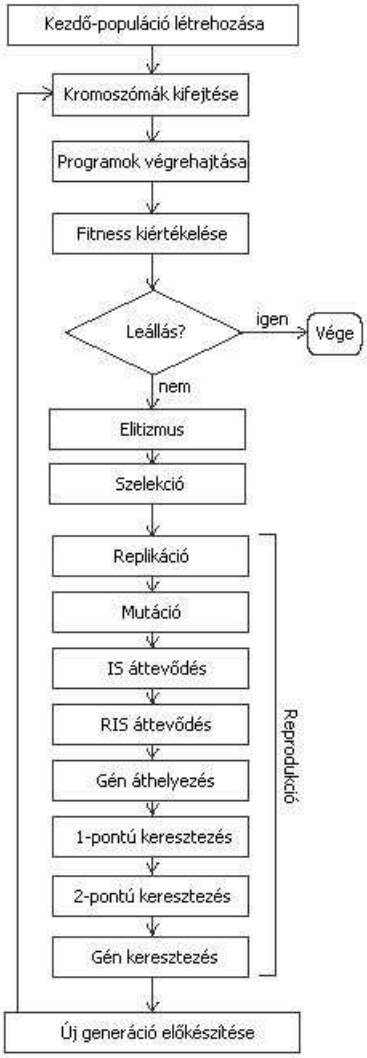
A 2.17.ábrán feltüntetett gén dekódolása megtehető a benne szereplő funkciók argumentum-számának (aritásának) ismeretében. Vesszük a gén első elemét (amely mindig funkció kell, hogy legyen), megtesszük a program-fa gyökerének és megnézzük az aritását. A gén hátramaradó részéből „lecsípünk” az aritásnak megfelelő számú elemet és a fában a funkciónak megfelelő csúcs gyermekeivé tesszük őket. A most „lecsípett” elemeknek megfelelő program-fa-beli csúcsok így a fa egy teljes új szintjét alkotják. Ezen a szinten „balról-jobbra” haladva, mindig „le-lecsípünk” a gén hátramaradó részéből valahány – az épp aktuális csúcsnak megfelelő funkció aritásával megegyező számú – elemet, amiket nyomban a megfelelő csúcs gyermekeivé teszünk. Ha terminálisnak megfelelő csúcshoz érünk, akkor ahhoz már nem fűzünk gyermeket s így nem is csípünk le a gén hátramaradó részéből se semmit. Így haladunk – végig a szinteken (balról-jobbra) – amíg van elem. Ha az adott szinten elfogynak az elemek, akkor egy szinttel lejjebb lépünk, s ott folytatjuk a csúcsok „kifejtését”. Tesszük ezt mindaddig, amíg a fa legalsó leveleihez, azaz csupa terminálisból álló szintre nem érünk. Ekkor a dekódolás végetér, s az előállított fával tér vissza.

A gének és az általuk reprezentált program-fák közti egy-egyértelmű kapcsolat ismeretében immár rátérhetünk a teljes algoritmus működésének a leírására. A génkifejező programozás működésének folyamatát a 2.18. ábra mutatja (Ferreira, 2001).

Kezdetben véletlenszerűen létrehozzuk kromoszómák egy populációját. A kromoszómák génjeinek a száma és a gének hossza adott. Az utóbbira egyrészt azért van szükség, hogy korlátozhassuk az előálló program-fák mélységét, másrészt meg azért, hogy egyáltalán

létrehozhassuk őket. Megállapodás szerint ugyanis a géneket két részre osztjuk: A gén elejét, ahol mind terminálisok, mind pedig funkciók előfordulhatnak, a *gén fejének,* a gén további részét pedig, ahol már csak terminálisok lehetnek, a *gén farkának* nevezzük. Ahhoz, hogy egy gént dekódolni tudjunk, meg kell adnunk a fejhosszát. Erre azért van szükség, hogy a fejrészben előforduló funkcióknak elég terminálist tudjunk „biztosítani" a gén farkában. Ez azt jelenti, hogy a gén program-fává konvertálásakor nem fordulhat elő olyan helyzet, hogy a „hátramaradt" részből (lásd. dekódolás fentebb) egy beolvasott funkció számára már ne legyen mit „lecsípni". Azaz, elég terminálisnak kell lennie a gén farkában a funkciók argumentumainak kielégítéséhez. Ha n az F halmaz funkcióinak maximális aritása, és h a gén fej-hossza, akkor a farok hossza legfeljebb $h(n-1)+1$ kell, hogy legyen. Magyarán a gén fej-hossza határozza meg a gén farok-hosszát, s így az egész gén hosszát, amely e két hossz összege.

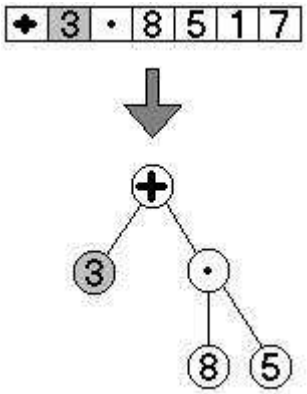
A következő lépés a rendelkezésünkre álló kromoszómák kifejtése, azaz programokká alakítása. Egy-egy kromoszómának megfelelő program a géneknek megfelelő programok valamifajta kompozíciója. Például a gének által kódolt program egymás után történő végrehajtása, stb. A kész programokat rendre végrehajtjuk, s rögzítjük a futás eredményeképp előálló jósági (fitness) értéküket. Megvizsgáljuk, hogy teljesül-e a leállási feltétel (azaz, hogy elértük-e már a maximális megengedett generációk számát, vagy, hogy van-e már adott fitness értékű egyed a populációban, stb.), s ha igen, akkor megállunk, s az aktuális populációt adjuk eredményül. Ha a leállási feltétel még nem teljesül, akkor egyrészt vesszük a populáció N legjobb egyedét és változtatás nélkül átörökítjük őket az új generációba. Ezt nevezzük *N-elitizmusnak.*



2.18.ábra. GEP algoritmus

Másrészt – a genetikus algoritmusoknál látott módon – feltöltjük a maradék helyet szelekció útján kiválasztott egyedekkel, ámde ezek még csak egy „átmeneti" generációba kerülnek, amely átmenetet képez a régi és az új között. Ezt követi a *reprodukció* fázisa, amit az átmeneti generációk egyedein hajtunk végre, előállítva az új generáció végleges formáját. A reprodukció első lépése a *replikáció,* melynek során a replikációs valószínűségnek megfelelően kiválasztjuk az aktuális átmeneti generáció egyedeit, és változtatás nélkül átmásoljuk őket az új generációba. A maradék egyedeket a mutációs valószínűségnek megfelelően – a genetikus algoritmusoknál látott módon – mutáljuk. Természetesen itt ügyelnünk kell arra, hogy a kromoszómán belül kiválasztott (mutálandó) pozíció egy gén fejébe, vagy farkába esik-e. Az utóbbi esetben csak terminálisra szabad változtatnunk, míg az előbbi esetben akár funkcióvá is mutálhatjuk az adott pozíció elemét.

Példaként lássuk a 2.17. ábrán szereplő gén egy mutációját, és a mutált génnek megfelelő program-fát (2.19.ábra).



2.19.ábra. A 2.17. ábrán szereplő GEP-gén egy mutációja és a mutált génnek megfelelő program-fa

Látható, hogy a mutált génből előálló program-fa már nem tartalmaz minden génbeli információt, s így lehetséges a *semleges mutációk* megjelenése. Ez a második fő szempont, amiért a GEP-et „természetesebbnek" tartjuk elődeinél. A harmadik fő szempont pedig az éppen tárgyalt reprodukcióban keresendő, ugyanis, amíg a klasszikus GP esetén egy-egy kiválasztott programot csak egy-egy genetikus operátor módosíthatott, addig most akár több is változtathat rajta, a valóságos evolúcióhoz hasonlóan, ahol nem kötött a változtatások száma.

A mutációt az *IS* (*Insertion Section)* és *RIS (Root Insertion Section)* áttevődések követik. Az előbbi egy-egy találomra választott gén egy-egy találomra választott, funkcióval kezdődő, adott hosszúságú szakaszát másolja át a gén fejében egy találomra választott másik pozícióra. Ami a fejen túlnyúlna, azt „levágja". A RIS áttevődés ugyanezt teszi azzal a különbséggel, hogy a fejben nem találomra helyezi el a másolt részt, hanem a gén legelejére, mondhatni a „gyökérre" másolja rá. Ezt követi a *gén áthelyezés,* mely a kromoszóma egy-egy találomra választott génjét mozgatja át a kromoszómán belül egy tetszőleges, véletlenszerűen kiválasztott helyre. Ezután – a genetikus algoritmusoknál vázoltaknak megfelelően – az egy és a két-pontú kereszteződések változtatják az átmeneti populáció elemeit, mígnem végül a *gén keresztezés* teszi véglegessé az átmeneti generációt, létrehozva az új generáció végleges formáját. Az új generációval pedig minden kezdődik újra elölről.

Összefoglalva, a génkifejező programozás az elődök jó tulajdonságait ötvöző, nagyságrendekkel hatékonyabb, globálisan konvergens sztochasztikus kereső-eljárás. Természetesen a konvergencia „sebességét és minőségét", s így az „optimumot" a paraméterek, és a jósági függvény megválasztása határozza meg.

3. Tervkészítés Problémája

Az előzőekben láthattuk, hogy a jó („optimális") megoldásra tett törekvéseket mennyi, de mennyi tényező befolyásolja. A bizonyítottan helyes és teljes tervezés – azaz, amikor a visszaadott

terv (ha létezik) mindenképp jó, „zsákutcába” jutva pedig biztosak lehetünk abban, hogy nincs megoldás – csak egyszerű környezetek, egy- vagy többállapotú problémáknál működött. Eshetőségi vagy felderíthetőségi problémák esetén legtöbbször már semmit se garantálhattunk. Ebben a fejezetben a következő kérdésekre keressük a választ: Mi az „optimális” terv? Mi a tervkészítés bonyolultsága? Egyáltalán, mindig megválaszolhatók-e ezek a kérdések?

3.1. AZ OPTIMÁLIS TERV FOGALMA

Az „optimalitás” viszonylagos fogalom. Ahhoz, hogy „optimális” megoldásról beszélhessünk, kell valamiféle viszonyítási alap, valamilyen jósági mérce, vagy kritérium, aminek alapján meghatározhatjuk egy-egy megoldás jóságát. Ebben a szakaszban a dolgozat saját definíciót ad az „optimalitás” fogalmára. A későbbiekben „optimális” címszó alatt mindig az itt bevezetett fogalmat értjük majd.

Ha a megoldások terét keresési-térnek tekintjük, akkor ezen belül különbséget tehetünk lokális és globális optimum közt. Míg az előbbi a keresési tér egy-egy választott részének legjobb megoldása, addig az utóbbi a teljes keresési térre vonatkozó, „optimális” megoldás. Célunk az, hogy olyan értelmes mércét állítsunk fel, mely az adott keresési tér elemeit céljainknak megfelelően osztályozza. Ha a keresési tér a – pl. 2.2.2. szakaszban megismert – tervek tere, akkor az „optimális” terv ismérvei intuitíve a következők lehetnek: legyen minél „rövidebb”, minél „robusztusabb” és adjon minél „jobb megoldást”. A terv annál rövidebb, minél kevesebb lépésből áll a végrehajtása; annál robusztusabb, minél több „esetben” képes célhoz érni; és annál jobb megoldást ad, minél inkább olyan cél-állapotot eredményez, amely igényeinknek megfelel. Az előbbiek formalizálásához a 2.1.1. szakaszban megismert leírásmódra támaszkodtam.

Legyen adott a lehetséges környezeti állapotok egy S halmaza, a lehetséges cselekvések egy A halmaza, egy $s_0 \in S$ kiindulási állapot, továbbá egy G cél-függvény, amely a környezetek halmazából képez a valós számok halmazába, azaz $G: S \to R$. Legyen $S_G \subseteq S$ azon környezeti-állapotok halmaza – nevezzük ezeket cél-állapotoknak –, melyekre G maximális, azaz, ahol $\forall s_i, s_j \in S_G$-re teljesül, hogy $G(s_i) = G(s_j)$ és $\neg\exists s_k \in S \setminus S_G$, melyre $G(s_k) \geq G(s_i)$.

Mivel a cselekvéseket a környezetben egy ágens kivitelezi, ezért beszélhetünk az ágens történetéről, s így karakterisztikus viselkedéséről (lásd. 2.1.1) is. Legyen az ágens karakterisztikus viselkedése, azaz lehetséges történeteinek halmaza H. Ekkor egy $h \in H$ történet i. környezeti állapotát $h_s(i)$, míg az abban végrehajtott cselekvést $h_a(i)$ jelölje. Egy $h \in H$ történet hossza a benne megtalálható cselekvés-átmenetek száma, amit röviden $|h|$-val jelölünk. Ha a történetnek nincs vége, akkor hossza végtelen. Egy $h \in H$ történet *minimális cél-hossza* $\min_i h_s(i) \in S_G$, gyakorlatilag – a történet kiinduló-állapotából – egy cél-állapot eléréséhez szükséges cselekvések

minimális száma. Ha $\neg\exists i$, melyre $h_s(i) \in S_G$ teljesül, akkor $h \in H$ minimális cél-hossza végtelen. Jelölje $|h|_G$ a $h \in H$ történet cél-hosszát.

Ekkor egy *t'* terv, amelynek – ágens általi – végrehajtása az *S* környezet egy $s_0 \in S$ állapotából kiindulva a $H_{t'} \subseteq H$ lehetséges történetek halmazát eredményezi, akkor tekinthető „optimálisnak”, ha:

(O1) $$\sum_{h \in H_{t'}} |h|_G = \min_t \sum_{h \in H_t} |h|_G$$

(O2) $$\sum_{h \in H_{t'}} G(h_s(|h|)) = \max_t \sum_{h \in H_t} G(h_s(|h|))$$

Az (O1) feltétel szerint tehát az a terv optimális, amelyet végrehajtva, átlagosan a lehető legkevesebb a cél-állapotok eléréséhez szükséges lépések minimális száma, továbbá a terv lehetőség szerint minden esetben „célba ér” (vagy legalábbis „célt érint”). Az (O2) feltétel szerint az a terv optimális, amelynek végállapotai lehetőség szerint minden esetben cél-állapotok, hiszen az (O2) kifejezés akkor ad maximális értéket. Meg kell jegyeznünk, hogy (O2)-nél feltételeztük, hogy *t'* végrehajtása véges hosszú történeteket eredményez. Összefoglalva: (O1) felelős a terv „rövidségéért” és „robusztusságáért”, míg (O2) a „hasznosságáért”.

Érdemes észrevenni, hogy, ha *G* bináris – azaz $G: S \rightarrow \{0,1\}$ úgy, hogy $G(s) = 1$, ha $s \in S$ egy cél-állapot, míg $G(s) = 0$ egyébként –, akkor az (O1) és (O2) feltételek „optimális” cél-orientált (lásd. 2.1.1.3) tervkészítést implikálnak, míg, ha *G*-re nincs megkötés, akkor haszon-orientáltat (lásd. 2.1.1.4), hiszen ekkor *t'*-vel *G* – mint haszon-függvény – maximalizálását kell elérnünk (minimális lépésszám mellett). Cél-orientált esetben természetesen – a haszon-orientált esettel ellentétben – nem a *G* függvény határozza meg (implicite) a cél-állapotokat, hanem, minthogy adottak, a *G* függvény van inkább „hozzájuk igazítva”.

Láthattuk, hogy a tervet egy-egy kiinduló-állapot viszonylatában neveztük „optimálisnak”, vagy „nem optimálisnak”. Természetesen lehetséges volna az „optimalitás” fogalmát függetleníteni a kiinduló állapottól, s így olyan „univerzális tervekre” törekedni, melyek „bárhol, bármikor, bármilyen esetben optimálisan célra-vezetők”, ám, mivel ennek megfontolása sajnos (messze) túlmutat a dolgozat keretein, ezért nem is igen törekedtem vizsgálatára.

A szakasz első kérdésére tehát adtunk egy választ. Azonban még nem válaszoltunk arra, hogy minden esetben létezik-e az így definiált „optimum”. A válasz sajnos kettős, ugyanis, bár (O1) és (O2) – ha a terv nem eredményez végtelen történetet – kiszámítható, mégse mindig használható mérce az „optimalitás” mérésére. Gondoljunk csak arra, hogy több olyan terv is lehetséges, amely minden esetben a lehető leghamarabb (a legjobb) célba ér. Ekkor a két terv közti különbség „elmosódik”, s így az „optimum” nem egyértelmű. Talán megnyugtató lehet az a tudat, hogy a legtöbb „optimum” definíciónak – minthogy nem zárja ki több különböző, ámde azonos jóságú megoldás létezését – számolnia kell ezzel a problémával.

3.2. A FELADAT NEHÉZSÉGE

Most pedig próbáljunk választ adni a fejezet elején említett második kérdésre – becsüljük meg a tervkészítés nehézségét! Ahhoz, hogy ezt megtehessük, be kell vezetnünk a komplexitással kapcsolatos főbb alapfogalmakat, majd pedig konkrétan definiálnunk kell, hogy mit is értünk pontosan „a tervkészítés nehézsége" címszó alatt.

Ebben a szakaszban két döntési-probléma kerül előtérbe: egyrészt választ szeretnénk kapni arra, hogy mi a terv létezésének eldöntésére alkalmas algoritmus minimálisan szükséges lépésszám-igénye (TERV-LÉTEZÉS), másrészt arra, hogy mi annak az algoritmusnak a minimálisan szükséges lépésszáma, amely el tudja dönteni egy-egy tervről, hogy célba ér-e (TERV-VÉGREHAJTÁS).

Elsőként csak *környezet-specifikus* eseteket vizsgálunk. Ez azt jelenti, hogy egy-egy környezet-leírás viszonylatában, adott kiinduló-állapot mellett vizsgáljuk meg a két döntési probléma nehézségét. A szakasz végén röviden kitérünk majd a *környezet-független* eseteket feldolgozó elméleti eredményekre is, hivatkozás szintjén említve meg a számunkra fontosabbakat.

Mielőtt azonban rátérünk az eredmények összefoglalására, „nagy vonalakban" be kell vezetnünk a megértésükhöz szükséges főbb komplexitás-elméleti alapfogalmakat. Ehhez többnyire az irodalomban elterjedt jelölés-rendszerre (Papadimitriou, 1994) támaszkodunk.

Döntési problémának nevezzük annak az eldöntését, hogy egy adott w bemenet része-e egy $S = \{w \mid P(w)\}$ $P(w)$ tulajdonságot kielégítő elemek halmazának. Egy U algoritmust akkor nevezünk polinom-idejűnek, ha a döntési problémára a bemenet hosszának polinomjával arányos számú lépésben képest választ adni, azaz – ha $|w|$ a bemenet hossza, és $t_U(w)$ az algoritmus minimálisan szükséges lépésszáma a w bemeneten, akkor – $t_U(w) \leq p(|w|)$, ahol $p(|w|)$ a bemenet hosszának polinomja. Azon döntési problémákat, melyekre létezik polinom-idejű algoritmus, a **P** probléma-osztály elemeinek tekintjük. Egy döntési probléma az **NP** osztályhoz (más jelöléssel a Σ_1P osztályhoz) tartozik, ha $w \in S$ eldöntése (azaz a $P(w)$ tulajdonság teljesülésének vizsgálata) ekvivalens a $\exists u P(u,w)$ teljesülésének eldöntésével, ahol adott u,w esetén $P(u,w)$ eldöntése polinom-időben megtehető. Az u-t a w „tanújának" nevezzük (Rónyai és társai, 1998). Hasonlóan, ha $w \in S$ eldöntése ekvivalens $\forall u P(u,w)$ teljesülésének eldöntésével, akkor a problémát **co-NP**-belinek, más szóval Π_1P-belinek nevezzük. Hasonlóan, egy döntési probléma Σ_kP-beli, ha eldöntése ekvivalens $\exists u_1 \forall u_2 \ldots P(u_1, u_2, \ldots, u_k, w)$ teljesülésének eldöntésével, és Π_kP-beli, ha eldöntése $\forall u_1 \exists u_2 \ldots P(u_1, u_2, \ldots, u_k, w)$ teljesülésének eldöntésével ekvivalens. Egy döntési probléma **PTÁR**-beli, ha $\exists k : k \leq p(|w|)$ úgy, hogy a probléma Σ_kP vagy Π_kP-beli. Jelöljön C egy tetszőleges probléma-osztályt. Egy problémát akkor nevezünk C-nehéznek, ha minden C-beli probléma „redukálható rá", azaz létezik egy *Karp-redukció* (Rónyai és társai, 1998), amely bármely C-beli problémát képes polinom-időben visszavezetni az említett problémára. Egy probléma C-

teljes, ha *C*-nehéz, és egyben *C*-beli is. Az említett problémák egy-egy probléma-osztály legnehezebbjei közé tartoznak. Meg kell még jegyeznünk azt, hogy, mivel valószínűleg $P \neq NP$, ezért – ahogy a tapasztalatok is mutatják – az NP-beli problémák megoldásához exponenciális, azaz közelítőleg 2^n-lépésszámú algoritmusok szükségesek, ahol $n = |w|$, a bemenet hossza. Hasonlóan, a Σ_2P, vagy Π_2P-beli problémák megoldásához már 2^{2^n} nagyságrendű lépésszámot igénylő algoritmusokra van szükség, nem is beszélve a PTÁR-beli problémákról… A P-hez, NP-hez és PTÁR-hoz hasonlóan definiálhatnánk L, NL és LTÁR, továbbá EXP, NEXP és EXPTÁR osztályokat is (Erol és társai, 1995), melyek már nem polinomiális, hanem logaritmikus, illetve exponenciális lépés-igény szempontjából osztályozzák a problémákat. Most pedig térjünk rá a tervkészítés komplexitás-elméleti eredményeinek vizsgálatára.

Elsőként tehát környezet-specifikus esetekben vizsgáljuk a tervkészítés nehézségét. Ezen belül a probléma-környezetek (lásd. 2.1) két fajtáját különböztetjük majd meg: *determinisztikus* és *véletlenszerű* környezeteket. Determinisztikus esetben a döntési probléma legyen a következő (TERV-LÉTEZÉS): Adott *D* környezet-leírás (amely tartalmazza a kezdeti állapotot és a cselekvések lehetséges kimeneteleinek leírását) és *f* cél ismeretében döntsük el, hogy létezik-e olyan *t* terv, amelyre $|t| \leq p(|D|)$ teljesül (azaz ahol a tervhossz a környezet „komplexitásának" polinomjával becsülhető) és amely megvalósítja *f*-et. Az idevonatkozó eredményeket a 3.1.táblázat foglalja össze.

Determinisztikus tervkészítési környezetek	
Döntési Probléma / **Terv típusa**	TERV-LÉTEZÉS
Teljes információ	NP-TELJES
Részleges információ, nincs érzékelés	Σ_2P-TELJES
Korlátos érzékelés	Σ_2P-TELJES
Korlátlan-érzékelés	PTÁR-TELJES
Részleges információ, teljes érzékelés	Π_2P-TELJES

3.1.táblázat. Tervezés bonyolultsága determinisztikus környezetben (Baral és társai, 1999)

A „teljes információ azt jelenti, hogy mind a környezetleírás, mind pedig a cselekvések kimenetele teljesen ismert (lásd. 2.2.1.: *egyállapotú problémák*). A második vizsgált eset arra vonatkozik, amikor mind a környezetről, mind a cselekvések kimeneteléről csak hiányos információink vannak, továbbá a terveinkben nem szerepelnek érzékelő cselekvések (lásd. 2.2.2.4). A harmadik eset az előbbi eset véges számú érzékelés lehetőségével kiegészítve. A negyedik eset a harmadik esettel azonos eltekintve attól, hogy az érzékelések számára itt már nem szabunk korlátot. Az utolsó, ötödik esetben az előzőekhez hasonlóan, csak részleges információnk van a környezetről és a cselekvések lehetséges kimeneteleiről, ámde a környezet minden jellemzőjének érzékelése (pl.

minden állapot-jellemzőt leíró állítás meglétének vizsgálata) lehetséges számunkra. Az eredmények sajnos sehol sem garantálnak polinom-idejű megoldást.

A véletlenszerű környezetek vizsgálati eredményeinek összefoglalásához definiálnunk kell a véletlenszerűséget kezelni képes algoritmusok, és a hozzájuk tartozó probléma-osztályok fogalmát. A PP (Probabilistic Polynomial) – az RP (Randomized Polynomial) probléma-osztályhoz „hasonlóan" – azon S nyelvek osztálya, melyekre $x \in S$ akkor, és csak akkor, ha létezik olyan polinom-tárkorlátos nem-determinisztikus M Turing-gép (azaz algoritmus), amely x-et többször fogadja el, mint ahányszor elutasítja. A PL (Probabilistic Logspace) probléma-osztályt hasonlóan definiálhatjuk, leszámítva, hogy ott egy logaritmikus-tárkorlátos Turing-gép meglétét kötjük ki. Adott C és C' tetszőleges probléma-osztályok esetén a $C^{C'}$ probléma-osztály azon nyelvek halmaza, amelyek *C-Turing redukálhatóak* a C' probléma-osztály nyelveinek halmazára. Az eddig felsorolt probléma-osztályok egymásba-ágyazottsága a következő (forrás: Littman és társai, 1998):

$$L \subseteq NL \subseteq PL \subseteq P \subseteq \begin{matrix} NP \\ co-NP \end{matrix} \subseteq PP \subseteq \begin{matrix} NP^{PP} \\ co-NP^{PP} \end{matrix} \subseteq PTÁR \subseteq EXP$$

Az eddigiek ismeretében már definálhatjuk és értelmezhetjük a véletlenszerű környezetek esetén felmerülő döntési problémákat és hovatartozásuk. A TERV-VÉGREHAJTÁS problémája szerint (jelenleg) arra kell választ adnunk, hogy: Adott D környezet (amely tartalmazza a kezdeti állapotot, a lehetséges cselekvéseket, azok kimenetelét és a cél-állapotok explicite adott halmazát), t terv (melyre $|t| \leq |D|$) és Θ küszöbszám mellett teljesül-e P(„t végrehajtása cél-állapotba vezet a D környezetben")$>\Theta$ egyenlőtlenség? A másik probléma az előzőekben is látott TERV-LÉTEZÉS aktualizált változata, miszerint: Az előbbi módon adott D környezet, Θ küszöbszám és $z \leq |D|$ küszöbérték esetén teljesül-e P(„Létezik olyan z hosszú t terv, melynek végrehajtása cél-állapotba vezet a D környezetben")$>\Theta$ egyenlőtlenség? E döntések komplexitását mutatja a 3.2. táblázat.

A táblázatban több, eddig definiálatlan fogalom is szerepel. A környezetet s*imának* nevezzük, ha a környezeti állapotok explicite fel vannak sorolva, míg *propozíciónálisnak*, ha az állapotokat propozíciónális logikai adat-struktúrákkal reprezentáljuk. Az utóbbi, bár sokkal tömörebb leírást biztosít, a TERV-LÉTEZÉS és a TERV-VÉGREHAJTÁS szempontjából sokkalta bonyolultabb (lásd. 3.2.táblázat). Az említett két döntési probléma bonyolultságát – mindkét környezet-reprezentáció mellett – több különböző tervtípus esetén szemlélteti a 3.2. táblázat. Ha a tervre vonatkozólag semmilyen megkötés sincs, vagy, ha a terv-hossz polinom-korlátos, akkor a TERV-VÉGREHAJTÁS problémájának bonyolultsága a környezet reprezentációjától függetlenül ismeretlen bonyolultságú. Ellenben – ugyanezen terv-típusok esetén – a TERV-LÉTEZÉS problémájára már akár polinom-rendű algoritmus is adható. Ez azért lehetséges, mert konkrétan kikötöttük a terv hosszát. A teljesen rendezett tervek vizsgálata nem jelent újdonságot, hiszen a 2.2.2. szakaszban már említettük őket.

Véletlenszerű tervkészítési-környezetek				
Döntési Probléma / **Terv típusa**	Sima környezet-leírás		Propozíciónális környezet-leírás	
	TERV-VÉGREHAJTÁS	TERV-LÉTEZÉS	TERV-VÉGREHAJTÁS	TERV-LÉTEZÉS
Nincs megkötés	-	P-TELJES	-	EXP-TELJES
Polinom hosszúságú	-	P-TELJES	-	PTÁR-TELJES
Feltételes Ismétlődő	PL-TELJES	NP-TELJES	PTÁR-TELJES	PTÁR-TELJES
Feltételes Teljesen Rendezett	PL-TELJES	NP-TELJES	PP-TELJES	NP^{PP}-TELJES
Teljesen Rendezett	PL-TELJES	NP-TELJES	PP-TELJES	NP^{PP}-TELJES
Részben Rendezett (optimista)	NP-TELJES	NP-TELJES	NP^{PP}-TELJES	NP^{PP}-TELJES
Részben rendezett (átlagos)	NP-NEHÉZ	NP-TELJES	PP-TELJES	NP^{PP}-TELJES
Részben Rendezett (pesszimista)	CO-NP-TELJES	NP-TELJES	CO-NP^{PP}-TELJES	NP^{PP}-TELJES

3.2.táblázat. Tervezés bonyolultsága véletlenszerű környezetben (Littman és társai, 1998)

Feltételes, teljesen rendezett tervek azok, amelyek teljesen rendezettek, de már – az FRRT-nél (lásd. 2.2.2.4) látott – feltételes lépések is szerepelhetnek bennük. Ezek kiterjesztései a feltételes ismétlődő tervek, melyekben egy-egy lépés többször is végrehajtódhat. A részben rendezett tervek vizsgálata három szemszögből történt: pesszimista, átlagos és optimista módon. Mivel a részben rendezett terveknek több sorba-rendezése is lehetséges, ezért gyakorlatilag teljesen rendezett tervek egy halmazának foghatók fel. Optimista esetben a részben rendezett tervet azon sorrendezése alapján ítéltük meg, amely a legnagyobb valószínűséggel eredményezett cél-állapotot. Pesszimista esetben a legrosszabb lehetséges sorrendezés alapján ítéltük meg, míg átlagos esetben az összes lehetséges sorrendezés átlaga alapján. A 3.2. táblázatban összefoglalt eredmények szerint tehát – két speciális esettől eltekintve – véletlenszerű környezetekben legtöbbször nem adható determinisztikus (polinom-rendű) algoritmus se a TERV-LÉTEZÉS, se a TERV-VÉGREHAJTÁS problémájára függetlenül a terv típusától, vagy a környezet reprezentációjától.

Ha eltekintünk a környezet specifikációjától, akkor – környezet-független esetben – a 3.3. táblázat foglalja össze a TERV-LÉTEZÉS problémájának eldönthetőségét:

A 3.3. táblázat szerint a funkció-szimbólumok bevezetése az operátor-leírásba – független az egyéb tényezőktől – általánosan **eldönthetetlenné** teszi a TERV-LÉTEZÉS problémáját, hacsak nem vezetünk be speciális megkötéseket (Erol és társai, 1995) a környezetre vonatkozólag. Ha a funkció-szimbólumok hiányában végtelen számú konstans megléte lehetséges, akkor a TERV-LÉTEZÉS problémája ugyancsak **eldönthetetlen** mindaddig, amíg nem zárjuk ki a több lehetséges

kezdőállapot és a negált előfeltételek, vagy következmények lehetőségét. Az említett opciók majd mindegyikének kizárásával azonban a TERV-LÉTEZÉS problémája **eldönthetővé** tehető.

Lehetnek-e az operátorok leírásában funkció-szimbolumok?	Előfordulhat-e végtelen számú konstans?	Előfordulhat-e végtelen számú különböző kiindulási állapot?	Megengedettek-e negált előfeltételek és/vagy következmények?	A **TERV-LÉTEZÉS** problémája eldönthető-e?
igen	igen/nem	igen/nem	igen/nem	ELDÖNTHETETLEN
	nem	nem	nem	ELDÖNTHETŐ
nem	igen	igen	igen/nem	ELDÖNTHETETLEN
		nem	igen	ELDÖNTHETETLEN
			nem	ELDÖNTHETŐ
	nem	nem	igen/nem	ELDÖNTHETŐ

3.3.táblázat. Környezet-független TERV-LÉTEZÉS problémájának eldönthetősége (Erol és társai, 1995)

Összefoglalva az eddigieket, láthattuk, hogy összetettebb, valósabb környezetek és problémák esetén már a terv létezésének eldöntése sem oldható meg egykönnyen. Ha nem korlátozzuk a környezet, illetve a terv leírását, akkor analitikusan kezelhetetlen bonyolultságúvá, vagy akár eldönthetetlenné válhat a megfelelő terv meglétének (lásd. TERV-LÉTEZÉS) ellenőrzése. Nem-determinisztikus környezetekben már a terv megfelelőségének (lásd. TERV-VÉGREHAJTÁS) eldöntése sem oldható meg kivárható időn belül. A probléma bonyolultsága pedig – eddigi tudásunk alapján – nem megbecsülhető, hacsak nem teszünk komoly megkötéseket a terv struktúrájára nézve. Mindebből tehát az következik, hogy az „optimális" tervek (O1) és (O2) feltételének biztosítása – a TERV-LÉTEZÉS és a TERV-VÉGREHAJTÁS problémák bonyolultságának köszönhetően – általános esetben sajnos nem oldható meg analitikusan.

4. Tervkészítő módszerek

Láthattuk, hogy az „optimális" tervkészítés feladata (lásd. 3.1) eshetőségi és felderíthetőségi problémák (lásd. 2.2.1) esetén analitikusan kezelhetetlen bonyolultságú (lásd. 3.2). A gyakorlatban alkalmazott tervkészítő rendszereknek valahogy mégis helyt kell állniuk, meg kell birkózniuk a valós környezet kihívásaival. A következőkben ilyen rendszerekre mutatunk példát. A szakaszt a rendszerek egy lehetséges osztályozása zárja.

4.1. KORÁBBI MEGKÖZELÍTÉSEK

A következőkben tehát gyakorlati tervkészítő módszereket és rajtuk alapuló tervkészítő rendszereket sorolunk fel időrendi sorrendben, a teljesség igénye nélkül.

A tervkészítés gyökerei egészen a XX. század '50-es éveiig nyúlnak vissza, mikoris Herbert Simon és Allen Newell kifejlesztették „LT (LOGIC THEORIST)" nevű rendszerüket (Newell és Simon, 1956). Ez volt az első rendszer, amely heurisztikákra építve feláldozta a „teljességet" a

hatékonyság kedvéért. Propozícionális kalkulusban adott logikai tételek bizonyítására volt képes a tételekből történő *visszafelé-következtetés* által. Ehhez volt hasonló Gelernter „GTPM (GEOMETRY THEOREM-PROVING MACHINE)" rendszere (Gelernter, 1959), az első program, amely konjunktív rész-célokat is képes volt kezelni. A következtetéshez felhasználta a szimmetria tényét, és az előzőleg belátott tény-állításokat is. Az első jelentős próbálkozásnak mégis a „GPS (GENERAL PROBLEM SOLVER)" tekinthető (Newell és Simon, 1961). Bár „emberi intelligenciát" igénylő problémák megoldására tervezték, mégsem bizonyult annyira általánosnak, mint amilyennek szánták. Elsőrendű logikai bizonyításokat, logikai fejtörőket és egyéb „egyszerűen leírható" logikai feladványokat volt képes kezelni. Ez volt az első tervező-módszer, amely különválasztotta a problémára és a tárgy-tartományra vonatkozó tudást. Itt jelent meg először az *eszköz-cél* analízis módszere, amely során a rendszer az aktuálisan vizsgált „objektum" és a „cél" közti különbség alapján egy – táblázatból előkeresett – „cselekvést" kezdeményezett a különbség csökkentésére. Ha az adott cselekvés az adott objektumon nem volt értelmezhető, akkor – rekurzíve – próbálta meg a megfelelő „alakra" hozni. Hasonló – tétel-bizonyítási – elveken (újabban már a *rezolúciót* (Robinson, 1965) is latba vetve) működött Green „QA3" rendszere (Green, 1969), amely robot-irányítási feladatok megoldását tűzte ki célul. Ehhez először formalizálta a problémát, majd a tétel-bizonyító által megoldást keresett rá az állapot-térben. Az állapotokat a logikai állítások – „szituációkat" reprezentáló – változói szimbolizálták. A cél-feltétel egy egzisztenciálisan kvantifikált logikai állítás volt, melyet a tétel-bizonyító – egy megfelelő szituációt reprezentáló – változó-behelyettesítéssel próbált meg kielégíteni.

A tervkészítés „időszámítása" mégsem e rendszerekkel, hanem sokkal inkább Richard Fikes és Nils J. Nilsson „STRIPS (STANFORD RESEARCH INSTITUTE PROBLEM SOLVER)" nevezetű rendszerével (Fikes és Nilsson, 1971) kezdődött. „Vezérlési szerkezetét a GPS alapján alakították ki, az előfeltételek és következmények kezelésére pedig a QA3 egy változatát használták fel beépített modulként." (Russell és Norvig, 1995). Leírásmódja lényegében a szituáció kalkulus és a Green által bevezetett formalizmus továbbfejlesztése, amit gyakorlatilag minden későbbi tervkészítő megörökölt. A dolgozatban vázolt RRT (lásd. 2.2.2.1) formalizmusához képest a megvalósítás fő különbségei, hogy a következmény rész ponált és negált állításait különválasztva, egy *„add-list"* és egy *„delete-list"* formájában tárolja, továbbá, amíg az RRT részben rendezett, addig a STRIPS teljesen rendezett terveket állít elő. A STRIPS képes a különböző rész-célok lépései közti *átlapolódások* kezelésére is, így hát anno az elsők közt oldotta meg a „Sussman anomáliát" (lásd. 10.2). Azonban eredeti változata a konjunktív célokból fakadó problémákból kifolyólag nem volt teljes. „A STRIPS vezette be a *háromszög-tábla* fogalmát, amely a teljesen annotált tervek egy hatékony ábrázolása" (Russell és Norvig, 1995). A STRIPS-nek volt továbbá egy – lényegében magyarázat-alapú – tanulási mechanizmusa is, amely tervek általánosításának tárolását tette lehetővé. Az így kapott terveket *„macrop"*-nak (mint makró-operátor) nevezték el, melyek lényege, hogy a tervben szereplő konstansokat lehetőség szerint változókkal helyettesítik. A

STRIPS-et az SRI International kutatóintézet „Shakey” robot-projektjében alkalmazták először. A módszer bizonyítottan PTÁR-teljes (Canny, 1985; Chapman, 1987; Bylander, 1992).

Az említett „Sussman anomáliát” a „HACKER” nevezetű rendszer (Sussman, 1973) tesztelése kapcsán publikálták. A rendszer *képességek* megszerzését, épülését és fejlődését modellezte. Képességnek tekintjük a környezet valamely problémájának megoldására szolgáló eljárások halmazát. Ha új problémával szembesült, melyhez még nem volt megfelelő eljárás a birtokában, megtervezte azt. A visszalépéses technika helyett inkább *nyomkövető szakértőket* használt a rész-célok ütközéseinek feloldására. Így a STRIPS-szel egyetemben ez a rendszer is alkalmas volt a „Sussman anomália” orvoslására. Sajnos azonban nemhogy nem volt teljes, de még a megoldásul kapott tervek se voltak feltétlen „optimálisak”. Ellenben David Warren „WARPLAN” nevezetű rendszere (Warren, 1974) – előremozdítások és időbeni visszalépések segítségével (100 Prolog-sorban) – már képes volt a „Sussman anomáliára” „optimális” megoldást adni. A rendszer „WARPLAN-C” nevű változatában (Warren, 1976) jelent meg először a feltételes tervkészítés elve. Az „INTERPLAN” a (Tate, 1975) WARPLAN-hez hasonlóan képes volt a „Sussman anomália” megoldására, amihez úgynevezett *pillanat listákat* használt a védet kapcsolatok fenyegetéseinek feloldására, lehetővé téve a tervlépések közti tetszőleges átfedéseket.

E rendszerekkel nagyjából egy időben jelent meg az „ABSTRIPS (ABSTRACT STRIPS)” (Sacerdoti, 1974), amely korának első absztrakt, hierarchikus tervkészítője volt. Ez a módszer még nem a – 2.2.2.3. szakaszban ismertetett HD-RRT-nél látott – hierarchikus dekompozíció elvét használta, hanem úgynevezett *közelítő hierarchiák* alapján kezelte az absztrakt operátorokat. Ennek lényege, hogy a tervkészítés – bár ugyanazon operátorok fordulnak elő a terv minden absztrakciós szintjén – az operátorok előfeltételeihez *fontossági szinteket* rendelve, a magasabb fontossági szintektől halad lefelé, azaz először csak a legfontosabb előfeltételeket veszi számításba, majd az így kialakított tervet tovább bővíti a kevésbé fontos előfeltételek hatásait is figyelembe véve egészen addig, mígnem már minden operátor minden előfeltétele szerepel a tervben. Bár az ABSTRIPS képes volt a hierarchikus tervkészítés megvalósítására, tervei még mindig teljesen rendezettek voltak.

Az első hierarchikus, részben rendezett tervkészítőnek a „NOAH (NETS OF ACTION HIERARCHIES)” tekinthető (Sacerdoti, 1975), amely – javítási műveletek végrehajtását támogató – szakértői rendszerként működött. Egyik módszere az *állapotkényszerítés* volt, amivel a környezetet – bizonytalanság esetén – egy ismert állapotba „kényszerítette”. A NOAH hierarchikus megoldásai még a közelítő hierarchiák elvén alapultak. A részben rendezett tervek reprezentációjára *procedurális hálók* szolgáltak. Az úgynevezett *többes hatások táblázata (THT)* volt hivatott az egyes hálók csúcsai (terv-lépések) által „beiktatott és kitörölt” propozicionális logikai állítások nyilvántartására. A rész-célok közti kölcsönhatások az egyes állítások – egynél több csúcs általi – módosításának hatására jöttek létre, így a THT táblázatok figyelésével megoldható volt a fenyegetések (tervet veszélyeztető lépések) kiszűrése. A THT táblázatok felügyeletét *kritikusok*

végezték, melyek a fenyegetéseket a terv átrendezésével oldották meg, továbbá kiszűrték a tervben többszörösen előforduló (redundáns) operátorokat és adott esetben változó-behelyettesítést is végeztek a tervkészítés elősegítésére. A NOAH a *legkisebb megkötés* elvét alkalmazta olyan tekintetben, hogy a tervet csak végszükség esetén rendezte sorba. A rendszer egy sikeres kiterjesztése volt a „NONLIN (NON-LINEAR)" (Tate, 1977), amely lehetővé tette heurisztikák alkalmazását, illetve *függőség-vezérelt* visszalépéssel egészítette ki a NOAH-t. A NONLIN volt az első olyan tervkészítő, „amely külön algoritmust használt az előfeltételek igazságértékének eldöntésére részleges tervekben" (Russell és Norvig, 1995). Elektromos turbinák generáljavításához használták.

Hasonló rendszerek jelentek meg más diszciplínák kutatási területein is. Molekuláris genetikai kísérletek tervezésére szolgált a „MOLGEN (MOLECULAR GENETICS)" (Stefik, 1981). A rendszernek két kulcspontja volt: egyrészt *kényszereket* vezetett be a részproblémák közti interakció explicit, deklaratív reprezentációjára, másrészt *több-szintű tervezést* tett lehetővé. Ez utóbbi három szintre volt bontható: a legmagasabb absztrakciós szinten, a *stratégia térben* az épp követendő tervkészítési stratégia került meghatározásra. Egy szinttel lejjebb, a *vázlat térben* a legalsó szint, a *tervezési tér* terveivel tervezhettünk, mintha csak egy meta-terv lépései volnának. A szintek közül csak a legalsó volt probléma-specifikus. A MOLGEN-hez hasonló kényszereket vezetett be a „SIPE (SYSTEM FOR INTERACTIVE PLANNING AND EXECUTION MONITORING)" is (Wilkins, 1988). Mint ahogyan a nevéből is sejthető, a rendszer lehetővé tette a terv végrehajtásának felügyeletét (lásd. 2.2.3.3), s így az újratervező rendszerek előfutárának tekinthető. Ezen felül bevezette a tervezésbe az *erőforrás* fogalmát. Gyakorlatilag minden előfeltételt egy-egy bináris erőforrásnak tekintett, amely vagy szabad, vagy foglalt. Képes volt továbbá az erőforrásokat több különböző lépéshez is hozzárendelni. Egyéb előnyeinek és általánosságának köszönhetően felmerült komolyabb, valós tartománybeli alkalmazhatósága. Többek közt repülőgép-anyahajók leszállófedélzetének tervkészítési műveleteiben, többszintes épületek tervezésében és egy sörgyár gyártásszervezésében is segédkezhetett.

Ezzel nagyjából egy időben jelent meg a „TWEAK" (Chapman, 1987), amely a hierarchikus, részben rendezett tervkészítés első – bizonyítottan – helyes és teljes módszerei közé tartozik. Érdekességet jelent a tervek előállításának a módja, amely a *kényszer küldés* elvén alapszik. Ezek szerint a terv fokozatosan – újabb és újabb kényszerek beiktatásával – alakul ki. Újabb ugrást jelentett az „SNLP (SYSTEMATIC NON-LINEAR PLANNER)" (McAllester és Rosenblitt, 1991) megjelenése, amely bevezette a *szisztematikus* részben rendezett tervkészítés fogalmát. Szisztematikus tervkészítés alatt azt értjük, hogy a tervezés során egyetlen teljes vagy részben rendezett tervet se vizsgálunk meg egynél többször. A részben rendezett tervet itt definiálták először operátorok, okozati kapcsolatok és sorrendi kényszerek halmazának hármas adatstruktúrájaként. A rendszert gyakorlatilag a NONLIN egyszerűbb változatának *kiterjesztéssel* történő kiegészítésének tekinthetjük. A kiterjesztés – a rezolúció (Robinson, 1965) kiterjesztés lemmájának megfelelően –

lényegében behelyettesített elsőrendű-logikai állításokon végzett rezolúciós bizonyításokhoz garantál olyan bizonyítást, mely ugyanezen állítások behelyettesítetlen változataira vonatkozik. Ezért a tervkészítés megengedi, hogy újabb és újabb operátorokkal bővítsük a tervet más tervbeli lépések előfeltételeinek kielégítése végett. Az előbbi megfontolásokat gyakorlatilag a 2.2.2.1. szakaszban tárgyalt RRT algoritmus foglalja össze.

Az RRT-ben használt leírónyelvet először az „O-PLAN (OPEN PLANNING)" nevű rendszer (Currie és Tate, 1991) egészítette ki az idő- és az erőforráskényszerek általános kezelésének módjával. Továbbá ez a módszer adott elsőként lehetőséget a HD-RRT-nél (lásd. 2.2.2.3) látott hierarchikus terv-reprezentációra és dekompozícióra. Heurisztikákat kínált a terv-keresés gyorsítására, miközben minden döntése okát regisztrálta. Jó tulajdonságai folytán szívesen alkalmazták (és alkalmazzák most is). Számos konkrét implementációja létezik a gyakorlatban: a gyártásszervezéstől a logisztikáig, az űrkutatástól a szoftverfejlesztésig. A kiterjesztések RRT-DUNF esetén (lásd. 2.2.2.2) látott letisztult egysége először az „UCPOP"-ban (Penberthy és Weld, 1992) jelenik meg. Az UCPOP egy hierarchikus, részben rendezett, többágenses tervezést támogató, feltételes következményeket is kezelni képes tervkészítő rendszer, az „ADL (ACTION DESCRIPTION LANGUAGE)" (Pednault, 1986) egyik legteljesebb megvalósítása. Máig ezt a – bizonyítottan helyes és teljes – módszert tekintik a részben rendezett tervkészítés kulcspontjának.

A '90-es évektől napjainkig számos egyéb tervkészítő rendszer született. Számuk az utóbbi években – az igényekkel összhangban – rohamosan megnőtt, ezért a dolgozat, ha teljes felsorolásukra nem is, de a legfontosabbak közlésére próbálkozást tehet. A feltételes tervkészítés két alappillére az „UWL" (Etzioni és társai, 1992) és a „CNLP" (Peot és Smith, 1992), amelyek elvei az FRRT (lásd. 2.2.2.4) algoritmusban kerültek összefoglalásra. Valószínűségi következmények kezelésére is alkalmas feltételes tervkészítési módszer a „C-BURDIAN" (Draper és társai, 1994). Az „UMCP (UNIVERSAL METHOD COMPOSITION PLANNER)" (Erol és társai, 1994) a hierarchikus dekompozíció (akkoriban *HTN – „Hierarchical Task Network Planning"* néven emlegetett) elvének formális analízisére adott lehetőséget. Nagyjából ebben az időben rekedt meg a klasszikus módszerek továbbfejlesztése, és jelentek meg új eljárások. Ezek közül említésre méltó a „GRAPHPLAN" (Blum és Furst, 1995) és egy, a későbbiekben „SATPLAN"-nak elkeresztelt módszer (Kautz és Selman, 1996). Mindketten két fázisra bontották a tervkészítés folyamatát. Az előbbi egy *terv-gráfot*, míg az utóbbi *jól-formált propozíciónlális kalkulusbeli konjunktív normálformájú állítások* rendszerét hozta létre a probléma reprezentációjára, majd ebben „kereste meg" a megoldást. A két módszer előnyeit ötvözte a „BLACKBOX" (Kautz és Selman, 1998), amely a standard STRIPS-jelölésben megadott tervkészítési problémákat először terv-gráffá konvertálta, a terv-gráfot „CNF wff" formára hozta, majd a SATPLAN módszerével megoldotta. A BLACKBOX újabb verziói már PDDL (McDermott és társai, 1998) nyelven reprezentált problémákat is képesek voltak fogadni. E módszereket követte a hierarchikus dekompozíció – UMCP-nél látott – hagyományait felelevenítő „SHOP (SIMPLE HIERARCHICAL ORDERED PLANNER)" (Nau és társai,

1999), amely nagyságrendekkel hatékonyabbnak bizonyult a BLACKBOX-nál. A *sorrendezettség* azt jelenti, hogy a tervlépéseket a majdani végrehajtásukkal azonos sorrendben (mondhatni a részcélok szintjén rendezve a tervet) veszi figyelembe a (progresszív) tervkészítő.

Születtek azonban újszerűbb megközelítések is. A „MIPS (MODEL CHECKING INTEGRATED PLANNING SYSTEM)" (Edelkamp és Helmert, 2000) a modell-ellenőrzés módszerét vetette latba a tervkészítés során. Ez a rendszer használta először a *bináris döntési diagrammokat (BDD)* a környezet állapotainak tömör reprezentációjára. A „TLPLAN (TEMPORAL LOGIC PLANNER)" (Bacchus és Ady, 2001) a temporális-logikát hívta segítségül, lehetővé téve az „idő" fogalmának tisztán logikai alapon történő kezelhetőségét. A módszer a környezet – tetszőleges komplexitású – elsőrendű temporális-logikai reprezentációjának kezelésére képes – túl a tétel-bizonyításon – csupán a modell-ellenőrzés alapján. Az „FF (FAST-FORWARD PLANNING SYSTEM)" nevezetű rendszer (Hoffmann és Nebel, 2001) ezzel szemben – mondhatni klasszikus módon, az ABSTRIPS-hez hasonlóan – az operátorok – egyfajta heurisztikus – *relaxációjával* próbálkozik. A „VHPOP (VERSATILE HEURISTIC PARTIAL ORDER PLANNER)" (Younes és Simmons, 2002) a SHOP-pal ellenben már csak részleges sorrendezettséget kíván, hasonlóan a SHOP legújabb változatához, a „SHOP2"-höz (Nau és társai, 2003).

4.2. MEGKÖZELÍTÉSEK FAJTÁI

Az eddig felsorolt módszerek alapján (lásd. 10.1) a tervkészítés két főbb fajtáját különböztethetjük meg: az *analitikus* és a *heurisztikus* módszereket. Az analitikus módszerek matematikai úton, bizonyítottan próbálnak meg megoldást adni a tervkészítés problémájára, ami sajnos csak – a valós, fizikai környezethez mérten – igen „egyszerű és korlátos" környezetek esetén lehetséges (lásd. 3.2). A heurisztikus módszerek ezzel szemben nem pusztán matematikai alapon, hanem a feladat valamely probléma-specifikus aspektusának kihasználásával, bizonyos paraméterek önkényes hangolásával, vagy más, intuitív meggondolás alapján keresik a megoldást. Így akár nagyságrendekkel is hatékonyabbak lehetnek analitikus vetélytársaiknál mind a megoldás sebességének, mind pedig a megoldható feladatok körének tekintetében. A „gyorsaság és rugalmasság" növelésének azonban ára van: az analitikus eljárások „megbízhatóbbak és robusztusabbak", hiszen bizonyítottan adnak (többnyire) „optimális" megoldást egy-egy (egyszerűbb) feladatra, illetve a megoldás előállítását (általában) garantálni is tudják. Mindennek ellenére, eddig még egyik fajta megközelítés sem tudta garantálni az eshetőségi és felderíthetőségi problémák (lásd. 2.2.1) „optimális" megoldását (lásd. 3.1) általános esetben.

A következőkben egy olyan új heurisztikus megközelítést mutatunk be, amely képes lehet e nehézségek elvi kiküszöbölésére, és – megfelelő gyakorlati megvalósítás esetén – a tervkészítés feladatának „általános" és „optimális" megoldására.

5. Megoldási javaslat: Evolúciós Ágens

Ez a fejezet foglalkozik a dolgozat központi tárgyát képező új elgondolás megfogalmazásával, kidolgozásával és megvalósításával. A feladatkiírás szerint egy olyan „tervkészítő rendszer” létrehozása volt a cél, amely a tervkészítési folyamat „módszereinek és sémáinak” megadása nélkül teszi lehetővé az „optimális” terv elkészítését.

Ez egyrészt jelentheti azt, hogy a rendszernek „magától kell rájönnie” arra, hogy „miképpen kell tervet készíteni”, másrészt pedig azt is, hogy a rendszer a tervet **nem** előre megadott cselekvések, környezet-leírás és egyéb tervezési információ alapján állítja elő, **hanem** az említett információkat is „maga szerzi be”, rendszerezi és használja fel. Tehát mind a tervkészítés „*mit*”-jére, mind a „*hogyan*”-jára „maga ad választ”. Továbbá – a kiírás szerint – ez a „válasz optimális”.

A kiírás ezen informális értelmezése igen komoly feladatot takar, melyet valójában már az előző – elméleti bevezetést adó – fejezetek során „körbejártunk”. Foglalkoztunk *ágensekkel* (lásd. 2.1), melyek hozzáférhetetlen, dinamikus és nem-determinisztikus környezetekben, eshetőségi és felderíthetőségi problémákkal küszködve, végső soron „a világ egy megfelelő modelljével” kellett, hogy rendelkezzenek ahhoz, hogy képesek legyenek helytállni. Foglalkoztunk *tervkészítéssel* (lásd. 2.2), mely során a tervkészítő rendszereknek (lásd. 4. fejezet) komplex, a valósághoz igen hasonló környezetben kellett boldogulniuk, ami – mint kiderült – adott bonyolultság felett analitikusan kezelhetetlen nehézségű feladat (lásd. 3. fejezet). Foglalkoztunk továbbá az *evolúció* számítógépes modelljével (lásd. 2.3), amely az „optimális” megoldás előállításának egy lehetséges – globálisan optimális – módja. Összefoglalva: a három tárgy-terület mindegyike a kiírásban szereplő feladat egy-egy részének megoldására törekszik, s így egyesítve elgondolásaikat, előnyeiket és eredményeiket, olyan rendszer hozható létre, amely „együttes erővel” már képes lehet a feladat megoldására.

5.1. AZ ALAPÖTLET

Az elképzelés során tehát három fogalom találkozik egymással: az „*ágens*”, a „*tervkészítés*” és az „*evolúció*” fogalma. A kiírásban szereplő rendszert tervkészítő ágensnek (lásd. 2.2.3) tekintve, megoldásom alapgondolata a következő:

„Legyen adott egy környezet, egy probléma-tér, melyben ágensek egy populációja fejlődik nem-felügyelt evolúció mintájára úgy, hogy egy-egy ágens egy-egy saját, bensőleg felügyelt evolúció útján érleli ki a környezete felé legjobbnak vélt cselekvéseket”

Az megoldás alapötletének leírásában szereplő fogalmakat – bár az előző fejezetek már részben és többféleképpen is definiálták –, ahhoz, hogy kezelni tudjuk, „közös nevezőre” kell hoznunk. A „környezet”, az „ágens” és az „evolúció” fogalmát általánosságban a következőképp definiálhatjuk egymás viszonylatában:

(**DK**) Környezetnek nevezzük a feladatot, vagy feladatokat reprezentáló probléma-teret, ahol a probléma, problémák, vagy tetszőleges részhalmazuk megoldása ekvivalens a feladat, vagy feladatok egy részhalmazának megoldásával.

(**DE**) Evolúciónak nevezzük azt a folyamatot, mely a környezet bármely, azonos típusú elemhalmazát, ún. populációját természetes szelekció mintájára változtatja az elemek környezet szempontjából előnyös attribútumainak függvényében.

(**DÁ**) Ágensnek nevezzük azt az autonóm, adaptív probléma-megoldó rendszert, amely a környezetben zajló valamely evolúció részeseként bizonyos feladatok megoldására törekszik.

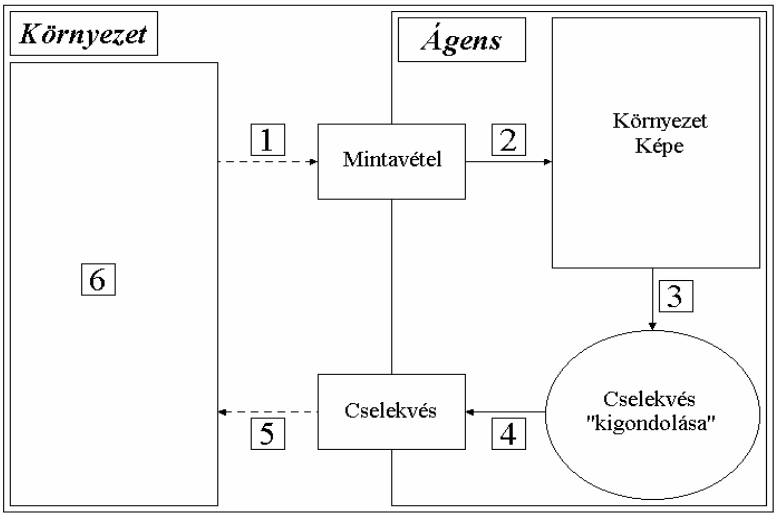
5.2. MEGOLDÁS KIDOLGOZÁSA

Tehát, mint látjuk, „két-szintű" evolúcióról van szó, melynek alsóbb szintjén az ágensek cselekvéseket evolválnak, míg a felsőbb szinten már maguk az ágensek versengnek egymással a fennmaradásért. A „két-szintűséget" elsősorban az indokolja, hogy – mivel az evolúció számítógépes modellje bizonyítottan globálisan konvergens (Holland, 1975; Goldberg, 1989), ezért – „evolúciók evolválásával" (megfelelően „gazdag" és „adaptív" ágens-leírás esetén) olyan evolúcióhoz (evolváló ágenshez) juthatunk, amely „optimálisan optimális", azaz optimálisan paraméterezett (kialakított) evolúciót „futtat". Az optimális paraméterezés lehetővé teheti az ágens számára, hogy olyan optimumokat állítson elő, melyek egy valóban globálisan optimális „mérce" (lásd. 2.1.1.4) szerint (globálisan) optimálisak. Így, ha az ágens mércéje megfelelően paraméterezett „meta-evolúció" (ágens-evolúció) eredménye, az ágens által előállított megoldás is „optimális" lesz a környezet viszonylatában. Mivel azonban a mércét – mint láttuk – nem-felügyelt evolúció útján érleljük, ezért valójában nincs is szükség „paraméterezésre", s így újabb „meta-meta-evolúciókra": ***a környezet*** *felépítése* ***implikálja*** *a megfelelő mérce (s így a megfelelő evolváló ágens)* ***kialakítását***, avagy azokat az attribútumokat, amelyek alapján az adott ágens az adott környezetben előnyösebb helyzetbe kerül és érvényesül. Természetesen minden evolúció átka, hogy a „jó" megoldáshoz tartás konvergenciája – „rossz" implementáció és/vagy feladat-reprezentáció mellett – kivárhatatlanul lassú, vagy akár divergens is lehet. Mint említettük, *az ágens-leírásnak kellően „gazdagnak" és „adaptívnak" kell lennie.* Ehhez az ágens-evolúció helyes megvalósítása szükséges.

Az előbbiek pontosításához először is meg kell vizsgálnunk az ágens és a környezet kapcsolatát, továbbá azt, hogy milyen ismérvek alapján értékeljük az egyes – beágyazott tervkészítő rendszereknek (lásd. 2.2.3.4) megfelelő – ágenseket, mit is tekintünk valójában a kiírásban szereplő feladat egy *„jó"* megoldásán.

5.2.1. Evolúciós ágensek absztrakt architektúrája

Mint ahogyan már utaltunk rá, az evolúciós ágens – a 2.1.1.4. szakaszban vázolt haszon-orientált absztrakt ágens-architektúrákhoz hasonlóan – egyfajta haszon-maximalizálást végez azáltal, hogy cselekvéseit egy bensőleg felügyelt evolúció útján érleli ki, hiszen a cselekvések szelekciójának megvalósításához egyfajta haszon-függvényre, avagy jósági mércére (lásd. 2.3) van szüksége. Azonban az evolúciós ágens egy nagyon lényeges szempontból **eltér** a haszon-orientált ágensektől: esetünkben az ágens-definíció **része** a környezetben zajló, nem-felügyelt evolúció. Az ágenst akkor tekintjük evolúciós ágensnek, ha – túl a cselekvések bensőleg történő kievolválásán – egy nem-felügyelt evolúció részese. Az evolúciós ágens architektúra tehát szerves részét képezi a környezetnek. Az evolúciós ágens ilyetén (definíció-szintű) beágyazottságát szemlélteti az 5.1. ábrán látható absztrakt ágens-architektúra.



5.1. ábra: Evolúciós ágens absztrakt architektúrája

Az ábrán az ágens-környezet interakció folyamata látható. Ennek lépései a következők:

1. Az ágens szenzorjaival mintavételezi a környezetet.
2. Az észlelések nyomán bővíti a környezetről alkotott eddigi képét, azaz előállítja a környezet egy „interpretációját”.
3. Eddigi tudása alapján felállít egy jósági-mércét, mely egy-egy környezeti állapot jóságát *(pl. az ágensnél lévő erőforrások számát, a környezet valamely kvantitatív, vagy kvalitatív jellemzőjét, vagy a többi ágens állapotának jóságát, stb.)* hivatott számára meghatározni.
4. „Kigondol” egy cselekvést, melyről úgy véli, hogy a végrehajtása nyomán előálló környezeti állapot jósága maximális.
5. Végrehajtja a cselekvést.
6. A módosított környezetben folytatja az 1. lépésnél.

Az ágens formális működése ezen az absztrakciós szinten gyakorlatilag megegyezik a 2.1.1.3 és 2.1.1.4 szakaszokban ismertetett cél- és haszon-orientált ágensekével.

Az 5.1. ábrán látható beágyazottságnak köszönhetően az ágens már akár – a környezet részeként – magát is érzékelheti, s így képet alkothat önmagáról a környezet viszonylatában. A 2. fejezetben bemutatott ágens-architektúrák közül csak a reaktív architektúrák (lásd. 2.1.2.2) vallottak hasonló elveket, elválaszthatatlannak tekintve az „intelligens viselkedést" a környezettől. Sajnos azonban kikötötték, hogy az ágens a környezet semmifajta reprezentációjával (képével) sem rendelkezhet. Valóban, az evolúciós ágens egy lényegi sajátossága, hogy *viselkedése* nem pusztán a környezettől „független", környezettel való kommunikációjának a megnyilvánulása, hanem a környezettel való „egyesülése" folytán létrejövő „viszony" eredménye. Egyrészt láthattuk, hogy a környezet szelektálja az ágenseket. Ez közvetve határozza meg egy-egy ágens aktuális viselkedését. Szublimáltabb, ámde mégis közvetlenebb módon befolyásolja az ágensek viselkedését a felépítésük: gyakorlatilag egy evolúciót valósítanak meg az evolúcióban. Bizonyos fajta önhasonlóság jelenik meg a struktúrájukban, mintha csak a környezet egy „leképződése" valósulna meg bennük. Ez a fajta önhasonlóság „intelligens viselkedés" megnyilvánulását eredményezheti abban az esetben, ha a környezetnek (mint aktív, dinamikus tényezőnek) elég lehetősége van arra, hogy az egyes ágensek benső evolúcióit a külső (meta) evolúcióhoz szabályozza, „igazítsa". Meg kell jegyeznem, hogy az *önhasonlóságot* többen is a természet (Barnsley, 1988) illetve az „intelligens viselkedés" (Hofstadter, 1979) fontos jellemzőjének tartják.

Valójában azért van szükség „intelligens viselkedésre" a kiírásban szereplő feladat általános megoldásához, mert másként nem valósítható meg az „optimális" tervkészítés tetszőleges, pl. felderíthetőségi problémák esetén (lásd. 2.2.1), ahol a környezet, a cselekvések modelljét, illetve a velük való tervezés mikéntjét a rendszernek színtisztán magától kell megtalálnia. Képzeljünk csak el egy olyan rendszert, amely képes felderíthetőségi problémák „optimális" megoldására általános esetben! Egy ilyen rendszer, kellően gyors és jól kivitelezett megvalósítás esetén, képes lenne bármely emberi teendő önálló elvégezésére, sőt… Bár az intelligens rendszerek tervezőinek – kimondva, kimondatlanul – egy ilyen rendszer megvalósítása a céljuk, számunkra csak elvi szinten, egy absztrakt ágens-architektúra erejéig jelenthet perspektívát az „intelligens viselkedés" Turing-tesztet kielégítő változatának megalkotása.

5.2.2. Evolúciós ágensek konkrét architektúrája

A mérnöki gyakorlatban többnyire nem létezik se általános feladat, se általános megoldás. Minden problémának van valamilyen probléma-specifikus aspektusa, amelyhez egy-egy speciális megoldás tartozik. Esetünkben kénytelenek leszünk ésszerű megszorításokat tenni az evolúciós ágens absztrakt architektúrájára vonatkozólag annak érdekében, hogy megvalósítható legyen. Elsőként az említett „gazdag és adaptív" ágens-leírás megalkotására és az ágens „jóságának" megragadására teszünk törekvést: „Gazdag"-nak tekintjük az ágens-leírást, ha az ágens képes a környezet kellően magas szintű reprezentációjára. „Adaptív"-nak tekintjük, ha képes egy, a feladatot kellően magas szinten reprezentáló jósági-mérce felállítására. Számunkra már – a

fentiektől eltérően – az a rendszer is megfelel majd, amely akkor „jó”, ha az ágens az 5.1.ábrán látott hat lépés ismételt végrehajtásával a feladatot reprezentáló probléma-térben előállított környezeti állapottokkal képes a feladat optimális megoldását jelentő környezeti állapotok minél hatékonyabb megközelítésére. Két kulcs-kérdés van:

(1) Hogyan állítja elő az ágens a jósági-mércét?

(2) Hogyan „gondolja ki” a legjobb cselekvést?

Kezdjük a másodikkal! Tegyük fel, hogy a jósági-mércét előállító „módszer” már ismert, s így a mérce is adott. Ekkor, ha a jósági-mércét fitness-függvényként (lásd. 2.3) fogjuk fel, amely adott cselekvéseket az általuk megvalósított környezeti állapotok jósága alapján osztályoz, akkor a legjobb cselekvés „ki-gondolása” megfelel a legjobb cselekvés e jósági mérce alapján történő „ki-evolválásának”, azaz evolúció útján történő „kiérlelésének”. Ahhoz, hogy a jósági mérce előállításáról érdemben szót ejtsünk, előzőleg a következőkre kell választ adnunk:

(a) Milyen a környezet felépítése?

(b) Milyen az ágens felépítése?

(c) Mit és hogyan észlel az ágens a környezetből?

A környezet probléma-független, általános felépítése lehet a következő: A környezet ágens által érzékelhető része a **külső reprezentáció**, amely *„objektumokból”, „ágensekből”, „lehetséges cselekvésekből”, „folyamatokból” és a köztük lévő „kapcsolatokból”* áll, ahol az egyes alkotórészek definíciója feladat-függő. Egy $s \in S$ külső reprezentációt, ahol S a lehetséges külső reprezentációk halmaza, ekkor egy ötös adatstruktúrával írhatunk le: $s = \{O, A, T, P, C\}$, ahol O az objektumok halmaza *(objects)*, A az ágenseké *(agents)*, T a lehetséges cselekvéseké *(transitions)*, P a folyamatoké *(processes)* és C a kapcsolatoké *(constraints)*, vagy más néven kényszereké. Ezek részletesebb kifejtésébe a későbbiekben bocsátkozunk.

Az ágens felépítése a következő: Az ágens működése három fő lépésre osztható: (1) érzékeli a külső reprezentációt, (2) gondolkodik és (3) cselekszik, módosítva a külső reprezentációt. Vegyük sorra a lépések végrehajtásához szükséges ágens-komponenseket. Az ágens a külső reprezentációt *„szenzorain”* át érzékeli, és *„effektorain”* át befolyásolja. Egy $s \in S$ külső reprezentáció szenzorikus mintavételezése nyomán egy $i \in I$ **benső reprezentáció** áll elő az ágensen belül, amely *„objektumok”, „ágensek”, „elemi cselekvések”, „folyamatok”, köztük lévő „kapcsolatok”* és egy *„összetett jósági mérce és döntési tényezők”* együttese. Formálisan a benső reprezentáció egy $i = \{O', A', T', P', C', F\}$ hatos adatstruktúra, ahol F elemi jósági mércék és döntési tényezők halmazainak a kettőse. Az első lépést (1), avagy az *Érzékelés:* $S^* \times I^* \to I$ függvény által leírt folyamatot **interpretációnak** nevezzük. Az interpretáció külső reprezentációk és benső reprezentációk véges, de nem korlátos sorozataiból képez a benső reprezentációk halmazába. A második lépés (2) a gondolkodás folyamata. Ezt egy *evolváló* egység végzi, melynek

a benső reprezentáció a bemenete és egy $t \in T'$ elemi cselekvésekből alkotott „*feltételes cselekvés-sorozat*”, más néven *„terv”,* avagy *„program”* a kimenete. Az ágens „gondolkodásáért” felelős evolváló egységet megvalósító függvény tehát formálisan a következő: *Gondolkodás:* $I \rightarrow M$, ahol M a lehetséges Turing-gépek halmaza. Az evolváló egység gyakorlatilag az elemi cselekvésekből kialakított *„terveket”*, mint egyedeket, versengteti egymással felügyelt evolúció mintájára, ahol a szelekció alapja a *„jósági mérce”*. Végül (3) az előálló Turing-gépet megvalósító tervet az ágens végrehajtja. Ennek formális definíciója: *Cselekvés:* $M \rightarrow T$.

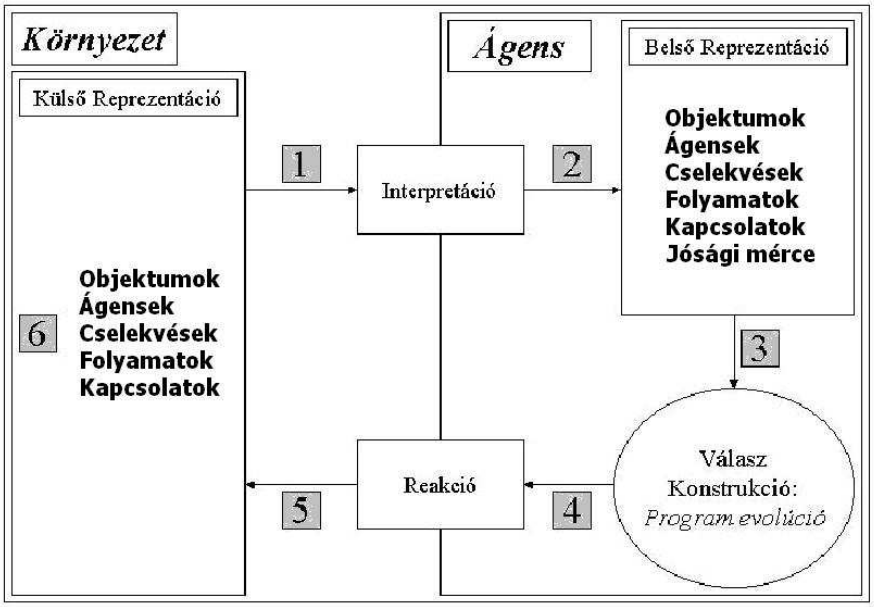
Láttuk, hogy a jósági mérce előállítása az interpretációs szakaszra tevődik, amikor az ágens a környezet külső reprezentációjából előállítja a belsőt. Ez azért van így, mert **az ágens a külső reprezentációt csak a benső reprezentáció formájában érzékeli „tudatosan”**. Ebből következően – minthogy a jósági mérce egy-egy cselekvés-sor eredményeképp előálló környezeti állapot megítélésére szolgál –, az ágens a benső evolúció során az egyes cselekvés-sorokat csak a belső reprezentáción tudja kiértékelni. Így tehát a jósági mérce, mint függvény, bemeneti paraméterként csak a benső reprezentáció elemeivel, illetve a cselekvés-sor jellemzőivel dolgozhat, mivel csak ezekhez van „hozzáférése”. – Tehát az ágens a jósági mérce előállításakor csak a benső-reprezentációra hagyatkozhat. A kérdés már csak az, hogy:

- *Mi is pontosan a jósági-mérce előállításának elve?*
- *Hogyan tűzi ki az ágens maga elé a célt, és mi ösztönzi arra, hogy ezt megtegye?*

Az ágens, minthogy szenzorai adottak, ha aktív állapotban van, akkor önkénytelenül is szembesül környezetével, akarva-akaratlan is észleli azt. Ebből következik, hogy az ágens „motivációját” a jósági mérce észlelésenkénti automatikus újraértékelése, s így az esetlegesen előálló új „cél” megjelenése jelenti. Az előállítás mikéntje lehet a következő: Kódoljuk az egyes ágenseket egy-egy $a_X \in X$ genotípussal (lásd. 2.3.1) úgy, hogy minden ágens-entitás tartalmazza az önmagát leíró genotípust, azaz adott $A \subseteq S$ ágensek esetén $\forall a \in A$-ra $a_X \subseteq a$ legyen. Ekkor egy-egy választott $c \in a_X$ *kromoszómával* kódolhatunk egy-egy *generatív szabálybázist*, amelyet az észlelési mechanizmus – a külső reprezentációban meglévő genotípusból kinyerve – fel tud használni arra, hogy a – benső reprezentáció részét képező – jósági mércét előállítsa. Gyakorlatilag minden, ágensen belüli „mozzanat”, minden ágens-komponens „lekódolható” ilyesfajta kromoszómákkal (lásd. 2.3.3).

Lássuk tehát az ágens 6-lépéses élet-ciklusának kiterjesztett, immár konkrétabb változatát:

1. ***Az ágens mintavételezi a környezetet.***
2. ***Az észleléseket interpretálja, előállítva a benső reprezentációt.***
3. ***A benső reprezentáció alapján inicializál egy benső evolúciót.***
4. ***Kievolvál egy optimálisnak vélt feltételes cselekvés-sort (tervet).***
5. ***Végrehajtja a tervet.***
6. ***Az immár módosított környezetben folytatja az 1. lépésnél.***



5.2. ábra: Evolúciós ágens konkrét architektúrája

Bár az élet-ciklus minden eleme kulcsfontosságú a rendszer működése során, mégis az „interpretáció" az, ami külön figyelmet érdemel az elv további kifejtésében. Nevezetesen arról van szó, hogy, mivel az interpretáció egy *konverzió* a külső, és a belső reprezentáció között, ezért előfordulhat, sőt legtöbbször elő is fordul, hogy (1) a két reprezentáció különbözik. (2) mivel a külső reprezentáció magába kell, hogy foglalja a belső reprezentációt – hiszen az ágens része a környezetnek (s így minden jellemzője is az) –, így a belső reprezentációnak (ami a külső reprezentáció képe) is tartalmaznia kellene ágenseket úgy, hogy benső reprezentációjuk tartalmazhat ágenseket úgy, hogy… Ennek szab határt az, hogy „az interpretáció általi transzformáció felbontása véges", azaz mások interpretációját csak véges mélységig vagyunk képesek interpretálni. Ez tehát megoldja a (2)-es esetet. Az (1)-es esetre a következő koncepció érvényes: Különböztessük meg a külső reprezentációt, azaz az ágens által érzékelhető környezetet a környezet teljes egészétől!

(**GV**) **Globális valóságnak** nevezzük a környezet teljes egészét.

(**LV**) **Lokális valóságnak** nevezzük az ágens által érzékelhető globális valóságot, azaz a globális valóság ágens által érzékelhető részét: a külső reprezentációt.

Az *interpretáció* értelemszerűen a lokális valóságból állítja elő a benső reprezentációt. Ha a lokális valóság a szenzorikus észlelés „valamifajta fogyatékossága" folytán olyan mértékben eltér a globális valóságtól, hogy annak egyértelmű visszaállítása a lokális valóság ismeretében már nem lehetséges, akkor a benső reprezentáció is szinte biztos, hogy el fog térni a külső reprezentációtól, hiszen a globális valóság nem hozzáférhető részeinek hatását a külső reprezentációra nézve nem képes követni. Ezzel tehát visszavezettük a (2)-es esetet, azaz a külső és benső reprezentáció közti „eltérést" az észlelés „fogyatékosságára". Hasonló elveket fogalmaz meg a fenomenológia

filozófiája (Heidegger, 1954), amikor – leegyszerűsítve – arról beszél, hogy a világnak csupán a „felszínét érzékeljük”, s mögötte „a mély láthatatlan”. Most pedig összegzésképpen lássuk az előbbi elképzelések egy formális megfogalmazását:

Legyen G a lehetséges **környezetek** (globális valóságok) halmaza. A **külső reprezentációk** (lokális valóságok) halmazát ebből úgy származtatjuk, hogy a halmaz egy-egy eleme azon $G' \subseteq G$ globális valóságok metszete, melyek ugyanazt az érzetet eredményezik, s így megkülönböztethetetlenek. Formálisan: $S = \left\{ \bigcap_{g \in G'} g \mid G' \subseteq G, \neg\left(\exists g_i, g_j \in G', \textit{Érzékelés}(g_i) \neq \textit{Érzékelés}(g_j)\right) \right\}$.

Így hát az S halmaz elemeihez külön-külön más-más érzékelés tartozik, avagy az S halmaz az ágens számára megkülönböztethető külső reprezentációk halmaza. Példának okáért tegyük fel, hogy G elemei, azaz a globális valóságok mind-mind különbözők, s így metszetük rendre üres. Ekkor, ha az ágens nem érzékeli köztük a különbséget, akkor külső reprezentációja az üres halmaz lesz, avagy az összes globális valóság metszete. Ez reális, hiszen, ha a különböző környezeteket azonosnak látja, akkor gyakorlatilag nincs érzékelése, s így külső reprezentációja sem. Ha ellenben az ágens képes különbséget tenni bármely két (teljesen) különböző globális valóság között, akkor külső reprezentációinak halmaza ekvivalens lesz a globális valóságok említett halmazával. Ez ugyancsak reális, hiszen G-nek nincs olyan két-elemű részhalmaza, hogy azon belül az ágens ne tudna különbséget tenni, s így az ágens érzékelése minden egy-elemű halmaz megkülönböztetésére alkalmas, tehát teljesen érzékeli a globális valóságot. A további feltevések próbálgatását a Tisztelt Olvasóra bízzuk.

Tovább folytatva, egy $s \in S$ lokális valóságon belüli $o \in O$ objektum egy o_D objektum-leírás és egy o_A attribútum-halmaz kettőse, azaz $o = \{o_D, o_A\}$, ahol o_D az objektum egyedi azonosítója, míg o_A jellemzőinek halmaza. Hasonlóan $a \in A$ ágens egy a_D ágens-leírás, egy a_A attribútum-halmaz, a főbb ágens-komponensek, illetve a a_I benső reprezentáció négyese, azaz $a = \{a_D, a_A, \{\textit{Érzékelés}, \textit{Gondolkodás}, \textit{Cselekvés}\}, a_I\}$. Az $s \in S$ lokális valóságon belüli lehetséges $t \in T$ cselekvések $t : G \to \wp(G)$ alakú (akár nem-determinisztikus) függvények, melyek egy-egy környezeti állapotból akár környezeti állapotok egész halmazát képesek előállítani. Hasonlóan definiálhatjuk a $p \in P$ folyamatokat, melyek a környezet – ágenstől független – alakulását szimbolizálják, s így $p : G \to \wp(G)$. Az $s \in S$ lokális valóságon belüli $c \in C$ kapcsolatok a környezet „rejtett szabályszerűségeit” szimbolizálják, melyek tetszőleges összefüggésbe hozhatják a külső reprezentáció bármely két elemét. Például jelenthetik egy ágens attribútum-halmazának változói és egy objektum attribútum-halmazának változói értékkényszereket (lásd. 2.2.2). A tervkészítésnél látott „okozati kapcsolatok” a cselekvésekbe és a környezet leírásába vannak implicite beágyazva, így az ágens a benső reprezentáció kapcsolatai közt őket is szerepeltetheti. A „sorrendi kényszerekkel” hasonló a helyzet. A kapcsolatok tárolása, reprezentációja már az ágens-implementáció kérdéskörához tartozik, amivel a következő

szakaszban foglalkozunk behatóbban. Most még foglaljuk össze az evolúciós ágens működésének lényegét – a 2.1.2. szakaszban látottakhoz hasonló – pszeudó-kódok segítségével.

```
function Evolúciós_Ágens(g ∈ G) {
        var i∈ I
        while Élek(g) do {
                i := Érzékelés(g, i)
                [t, i] := Gondolkodás(i)
                g := Cselekvés(g, t)
        }
}
```

Az evolúciós ágens működésének absztrakt vázlata tehát – az ágens szintjén – nagyjából megegyezik a cél- illetve haszon-orientált ágensek absztrakt szinten definiált működésével. Persze ne felejtsük el, hogy az evolúciós ágens már absztrakt szinten is a környezetébe van ágyazva azáltal, hogy definíció szerint részese egy nem-felügyelt evolúciónak. Így már absztrakt szinten is hasonlít a beágyazott tervkészítő ágens konkrét architektúrájára (lásd. 2.2.3.4). A konkrét architektúra tehát már a beágyazottság részleteit is fel kell, hogy fedje. Például rávilágíthat arra, hogy az *Érzékelés* függvény azáltal, hogy bemenetül kapja a globális valóságot, az ágens génállományát, azaz genotípusát is megkapja, hogy annak alapján előállítsa azokat a függvényeket, melyek a benső reprezentáció előállításáért felelősek. Rávilágíthat arra is, hogy a *Gondolkodás* függvény „mögött" miképpen húzódik meg az *evolváló* egység működése, amely a „gondolatok", avagy „feltételes tervek", más néven „programok" evolúció útján történő „érleléséért" felelős. Konkretizálás gyanánt lássuk az *Érzékelés* és a *Gondolkodás* ágens-komponensek működését:

```
function Érzékelés(g ∈ G,i ∈ I) returns i'∈ I {
        a_X := Genotípusom(g)
        f := Genotípus_dekódolása(a_X)
        s := Függvényhívás(f, Globálisból_lokális_valóság, g)
        return Függvényhívás(f, Interpretáció, s, i)
}
```

Az ágens *Érzékelés* komponensének működése ezen az absztrakciós szinten kissé eltér a megszokott függvényekétől. Bár valójában az ágens része, itt most mégis úgy tüntettük fel, mintha az ágens „felett való" volna. Ez az első két lépés, és az utolsó két lépés „elvi" különbségéből következik. Az első két lépés előállít egy *f* függvény-halmazt, amely az ágens gyakorlati értelemben vett érzékelésének felel meg. Gyakorlatilag ezen *f* függvényhalmaz függvényeinek „meghívásával" bontakozik ki az ágens valódi érzékelő mechanizmusa. Hasonlatképp azt tudnám mondani, hogy olyan ez, mintha az – ágens gyakorlati érzékelő mechanizmusát reprezentáló – *f* függvényhalmaz előállítása „tudat-alatt", míg függvényeinek végrehajtása „tudatosan" történnék. Ilyen szempontból tehát az *Érzékelés* függvény generikusan, ágenstől függetlenül valósít meg egy „érzékelés"-nek nevezett elvet, amely egy-egy ágens tekintetében más-más módon jelentkezik. A *Gondolkodás* komponens konkrét működési elve a következő:

```
function Gondolkodás(i ∈ I) returns (t ∈ M, i'' ∈ I) {
      [f, i'] := Előfeldolgozás(i)
      [t, i'] := Programok_evolúciója(f, i')
      i'' := Utófeldolgozás(t, i, i')
      return [t, i']
}
```

A „gondolkodás” elvét megvalósító ágens-komponens működése röviden igen egyszerű: a környezet adott (benső) reprezentációját kapja a bemenetén, ami tartalmazza a lehetségesnek „vélt” cselekvéseket, a környezet „vélt” leírását és egy jósági mércét. Célja az, hogy olyan feltételes cselekvéssorozatot találjon, amelyet, ha az ágens a „vélt” környezetben végrehajtana, akkor a jósági mérce szerint a cselekvések eredménye maximálisan hasznos volna (nem feltétlen az adott ágens számára). Természetesen a „vélt”, azaz interpretált környezet, a benső reprezentáció már (rég) nem a „valódi” külső reprezentáció, hanem annak csak egy – fenomenológiailag szólva – „felszíne”, képe, avagy interpretált változata. Itt szeretném megjegyezni, hogy, ha az ágensek nem-felügyelt evolúciója olyan ágenst hoz létre (márpedig globálisan optimális módszer lévén garantáltan létrehozza, **ha** előnyös) amely a külső reprezentációt már kellő pontossággal képes interpretálni, s így a benső és a külső reprezentációk közti különbség elenyészik, akkor az ágens által kievolvált tervek a környezetben, a külső reprezentációban is globálisan optimálisak kell, hogy legyenek (ha az azonos a globális valósággal), hiszen a benső reprezentációban a felügyelt evolúciónak köszönhetően már biztosan azok. Visszatérve, ahhoz, hogy a *Gondolkodás* komponens képes legyen az evolúció lefuttatására, először is ki kell „emelnie” a jósági mércét a bemeneti benső reprezentációból, másrészt pedig el kell készítenie a benső reprezentáció egy „hiteles másolatát” annak érdekében, hogy a későbbiekben evolvált programokat zavartalanul tesztelhesse rajtuk. Miután ezek rendelkezésre állnak, a benső reprezentáció „másolatán” lefuttatja a cselekvések, tervek, vagy akár programok evolúcióját. Ennek eredménye egy olyan *t* cselekvés, terv, vagy program, amely a benső reprezentáción végrehajtva képes a jósági mérce maximalizálására. A program evolúció során módosult benső reprezentáció „másolata” is megőrzésre kerül tanulságképp. A kievolvált cselekvés, illetve az evolúció folyamatát rögzítő adatok alapján – az *Utófeldolgozás* keretében – módosítjuk a benső reprezentációt, majd e a módosított („a gondolkodási folyamat tanulságait leszűrő”) változattal, illetve az említett cselekvéssel térünk vissza.

Összefoglalva megállapíthatjuk, hogy sikerült (elviekben) konkretizálnunk az evolúciós ágens absztrakt architektúrájának főbb komponenseit, s így bebizonyosodhatott, hogy az evolúciós ágens-architektúra képes lehet a diplomaterv feladatkiírásban vázolt feladat (elvi) megoldására. A konkrét architektúra által specifikált implementáció tervezésének és megvalósításának folyamatát a következő fejezetek tárgyalják. Az implementáció futási eredményei, melyek alátámasztják az elképzelés létjogosultságát, a 6. fejezetben olvashatók.

5.3. LEHETSÉGES MEGVALÓSÍTÁS

Ebben a fejezetben betekintést nyerhetünk az előbb vázolt alapelvek megvalósításának érdekesebb, fontosabb részleteibe. Mivel az implementáció elvi specifikációjának az előbbi (5.2.2) szakasz eleget tett, ezért itt már csak az elv gyakorlati megtervezésével és megvalósításával foglalkozunk.

5.3.1. Implementáció tervezése

Az 5. fejezetben tárgyalt elvek egy leegyszerűsített változatát valósítottam meg, ahol az egyes ágensek interpretációja úgymond „*átlátsztó*”, azaz nem a globális valóság modelljének hiányos ismeretében képzik a belső reprezentációt, nem szükséges hosszadalmas tanulás útján elsajátítaniuk a globális valóság egy helyes modelljét, hanem készen kapják, s így pontosan tudják, hogy egy-egy lokális valóság mit implikál a globális valóságra nézve. Szemléletesen azt mondhatnánk, hogy az ágensek pillanatnyilag készen kapják a „játék-szabályokat” a „játék” megkezdése előtt, avagy nem maguknak kell menet közben rájönniük. Ebből következőleg megoldásom még nem képes felderíthetőségei problémák (lásd. 2.2.1) általánosan „optimális” kezelésére (lásd. 3.1). Ellenben eshetőségi problémákra már garantáltan „optimális” megoldások adhatók, hiszen nincs szükség modell-tanulásra, megengedett, hogy az ágensnek a környezet egy helyes és teljes modellje álljon a rendelkezésére, s így az 5.2.2. szakaszban tárgyalt konkrét architektúra teljes megvalósítását feltételezve (az említett egyszerűsítés mellett, természetesen), valóban globális „optimális” lehessen.

Ezt úgy lehetne elképzelni, hogy például az ágens tudva, hogy az észleléseket milyen felépítésű, milyen elemekből álló világban kapta, megpróbálja kikövetkeztetni, hogy akkor mi is lehet a világ pillanatnyi állapota, ha ő éppen ezt, vagy azt érzékelte. Ha ismeretesek számára a lokális valóság és a globális valóság közti összefüggések, azaz, hogy például mikor és minek a hatására érez valamit, akkor tudni fogja, hogy egy-egy érzékelés alkalmával milyen valóság környezi, környezheti. Implementációm tehát megkívánja, hogy az ágens a benső reprezentációkat a globális valóság struktúrájának hiánytalan ismeretében alakítsa ki. Az előállított benső reprezentációkon belül azonban már mindenféle megkötés nélkül, az 5. fejezetben leírt elveknek megfelelően „gondolkozik”, a környezet struktúrája által adott „rejtett szabályszerűségeket” is figyelembe véve. A következő kérdések merülnek fel:

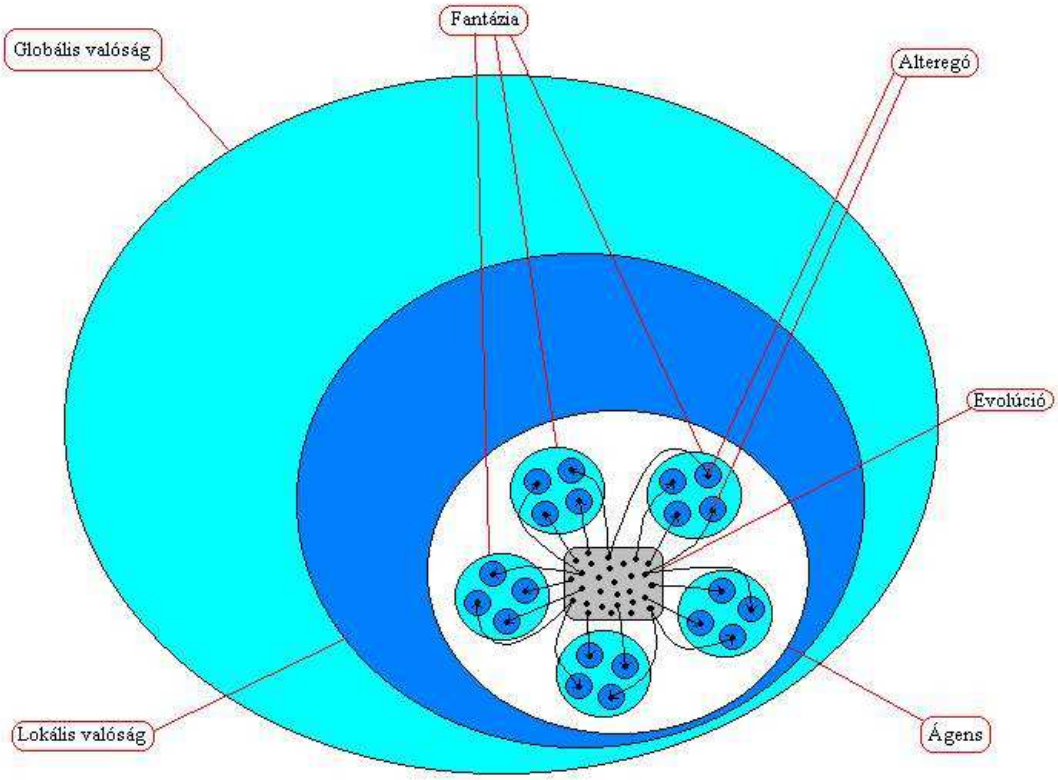
(1) Mit jelent az, hogy az ágens „gondolkozik”, és miképpen teszi ezt?

(2) Mik azok a „rejtett szabályszerűségek”?

Ahhoz, hogy a fenti kérdésekre választ adhassunk, felül kell vizsgálnunk az ágens élet-ciklusát, ami a lokális/globális valóság-koncepció nyomán enyhén átértékelődik. Újabb fogalmak vetődnek fel, amiknek megértéséhez segítséget az 5.3. ábra nyújthat segítséget.

Az ágens kiterjesztett élet-ciklusa tehát a következő:

1. Az ágens észlel egy lokális valóságot.
2. A globális valóságról eddig gyűjtött ismeretei (tudása) alapján a lokális valóságból legenerálja a globális valóság egy, vagy több lehetséges benső reprezentációját, ún. *fantáziákat*.
3. A fantáziákhoz inicializál egy evolúciót, melyben önnön *alteregóit* helyezi el, azaz olyan ágenseket, melyek saját lehetséges cselekvéseit, feltételes cselekvés-sorozatait szimbolizálják a fantáziákon belül.
4. Az evolúció futtatása során rendre *azonosul* az egyes alteregókkal, hogy kinyerhesse a fantáziákon leghatékonyabb feltételes cselekvés-sort, azaz a globális valóságban leginkább életképesnek remélt alteregót.
5. Végrehajtja az alteregó által szimbolizált cselekvést, módosítva a globális valóságot.
6. Az új globális valóságban az 1. lépésnél folytatja.



5.3. ábra: Ágens felépítése és összefüggése környezetével

Az 5.3. ábrán tehát egy ágens látható, beágyazva a globális valóságba úgy, hogy abból csak egy lokális valóságot érez közvetlen környezetének. Az ágensen belül jelen van egy populáció, amely az ágens alteregóiból áll. A benső evolúció során tehát az ágens ezeket az alteregókat, avagy cselekvéseket, terveket, vagy programokat evolválja. Az alteregókat az ágens rendre belehelyezi az eddigi tapasztalatok és az aktuális lokális valóság alapján lehetségesnek vélt benső

reprezentációkba, avagy fantáziákba. Ha meggondoljuk, akkor, ha – az 5.2.2. szakaszban elmondottak szerint – egy-egy globális valóság több lokális valóságot, azaz külső reprezentációt is implikálhat, akkor egy-egy lokális érzet többféle globális valóságból is származhat. Ezért lehet, sőt, kell több fantáziát kialakítani. A fantáziákon belül az ágens alteregói az ágenssel analóg módon csak egy lokális-fantáziával szembesülnek. Ezt az ágens „generálja" számukra a globális és a lokális valóság kapcsolatának ismeretében. Az alteregók e populációját evolválva, az ágens arra törekszik, hogy a fantáziákon minél hatékonyabb, minél sikeresebb alteregót „tenyésszen ki", hogy aztán a környezet felé az adott alteregó által reprezentált cselekvést, tervet, vagy programot hajtsa végre abban „reménykedve", hogy a globális valóságra vonatkozó modellje helyes. Természetesen, mint ahogy ezt már fentebb említettük, az ágens momentán a világ egy hiánytalan modelljével rendelkezik. Ámde ez a megkötés még bőven megengedi az eshetőségi problémák kezelését. Visszatérve az evolúció folyamatához, *azonosulásnak* nevezzük azt, amikor az ágens egy-egy alteregót „le-lefuttat" egy-egy fantázián. (Megjegyzés: Több-ágenses környezetben lehetőség van arra is, hogy az ágens egy másik ágens viselkedését, aktivitását reprezentáló alteregóval azonosuljon, miközben megint egy másik ágens jóságát tartja szem előtt. Alapesetben azonban az ágens a saját „fejével" gondolkozik, azaz saját magával azonosul a fantáziákon belül, és a cselekvések kimenetelénél is a saját hasznát veszi elsősorban figyelembe…).

A „rejtett szabályszerűségek", azaz az – 5.2.1. szakaszban bevezetett – kapcsolatok esetünkben a globális valóság olyan „szabályszerűségei", amelyek például azt szabják meg, hogy az ágens egy adott globális valóságban vett adott elhelyezkedése esetén, a globális valóság adott állapotában milyen észleléshez, azaz lokális valósághoz jut – **mit érez**. Mivel a lokális valóság a globális valóságnak csupán egy „leszüremlése", ezért, mint már említettük, előfordulhat, hogy egy-egy lokális valóság több globális valóság eredményeképp is előállhat egy-egy ágens számára. Ekkor az ágens, bár ismerheti a „rejtett szabályszerűségeket", mivel azok nem egyértelműek, ezért nemigen lesz képes egyértelmű, a globális valósággal ekvivalens fantáziát generálni – több fantázia előállítása szükséges. Ezek közül természetesen lehet, hogy valamelyik – ha a rejtett szabályszerűségek, és világ modellje teljes – épp a globális valósággal azonos fantázia lesz. Ezért is érdemes a fantáziákat súlyozni aszerint, hogy mennyire tartja őket az ágens valóságosnak. A fantáziák súlyozása azt befolyásolja, hogy a rajtuk „futtatott" alteregók értékelését milyen súllyal, a többi fantázián produkált eredményéhez képest milyen arányban vesszük figyelembe az adott alteregó jósági értékének kialakításakor. „Rejtett szabályszerűségeknek" tekinthetők továbbá a globális valóság elemei közti összefüggések is. Értelemszerűen a „rejtett szabályszerűségek" is lehetnek a lokális valóság részei, hiszen az ágens képes kell, hogy legyen arra, hogy „felismerje" őket. Lássunk most egy rövid, ám gyakorlatilag releváns példát arra, hogy mi a globális, és a lokális valóságok közti, sokszor lehelet-finom különbség! Legyen a globális valóság egy sakk-tábla, rajta bábukkal. Tegyük fel, hogy az ágens mindig csak egy-egy négyzeten belül helyezkedhet el, s a sajátján túl, csak a szomszédos nyolc négyzetet „látja". Ekkor azt mondanánk, hogy az ágens lokális

valósága az a kilenc négyzet, amit „lát”. Ámde mi van akkor, ha az ágens képes egy olyan cselekvésre, amely a fehér királyné irányába fordítja? Honnan tudja az ágens, hogy hol van a fehér királyné, ha a sakk-táblából csak kilence négyzet pillanatnyi ismerete áll a rendelkezésére? Első nekifutásra azt mondhatnánk, hogy az ágensnek van egy „globális érzéke” is, ami segít számára betájolni a királynőt. Ámde ugyanilyen erővel lokálisnak is tekinthetnénk a királynő „betájolását” lehetővé tevő érzetet, hiszen lehet, hogy az ágens egy „látáson túli” érzék segítségével operált. Tehát el kell, hogy rugaszkodjunk attól a „képtől”, hogy valaminek az „*érzékelése*” egyenlő a „*látásával*”. Túl azon, hogy a „*látás*” is csak **egy** érzék a sok közül, még az sem definit, hogy mit értünk alatta, hiszen gyakran a természetes nyelvi megfogalmazás folytán allegorikus, metaforikus, figuratív értelemben használjuk. Tehát, ha valamit nem látunk, de tudjuk például a térben vett elhelyezkedését, az nem annak a következménye, hogy globális ismeretünk van a tárgy helyét illetően, hanem sokkal inkább annak a következménye, hogy „*módunk van rá*”, hogy tudomást szerezzük róla. Ez a „mód” teszi a tárgy helyét a mi lokális észlelésünk részévé. Természetesen a lokális észlelés is csak a globális valóság része… Ha tehát az ágens birtokában van például egy GPS készüléknek, amely megadja számára a királynő orientációját és koordinátáit, akkor saját helyzete ismeretében képes lehet arra, hogy a megfelelő irányba forduljon anélkül, hogy globális tudásra lenne szüksége. Ha az ágens egy robot, akkor a GPS akár integrálva is lehet, mint egyfajta „érzék”. – Tehát összegezve azt mondhatjuk, hogy lokális valóság mindaz, amit az ágens a globális valóságból érzékel, azaz: **minden, amit az ágens érezhet**. (Hiszen definíció szerint a lokális valóság az ágens „érzet-világa”…) Ezáltal tehát egy olyan, konzisztens és általános fogalomhoz jutottunk, amely egységesen kezeli a globalitás és lokalitás tényét, nem úgy, mint amikor tetszőlegesen választjuk meg, hogy éppen mit tekintünk lokálisnak, vagy épp globálisnak. Ezek után kérdések lehetnek a következők:

- *Hogyan tud az ágens a lokális valóságból a globálisra következtetni?*
- *Ennek alapján hogyan állítja elő a fantáziákat?*
- *Az előállított fantáziákon belül pontosan hogyan „gondolkozik”?*

A kérdések így együtt sajnos túl összetettek ahhoz, hogy nyomban megválaszoljuk őket. A következő fogalmak bevezetése mellett azonban már könnyen a megválaszolásuk:

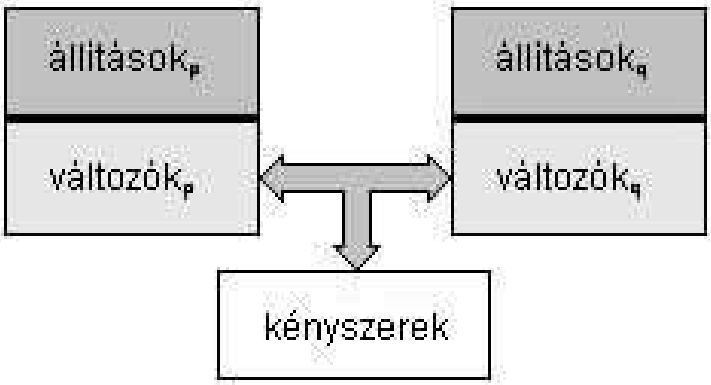
(1) Tapasztalat

(2) Következtetés

A tapasztalatot (1) az ágens a folyamatos működés során szerzi meg. A tapasztalat több reprezentációja lehetséges. Egyrészt beszélhetünk *lineáris emlékezetről.* Ebben az esetben az ágens pusztán csak azt rögzíti, hogy mit érzet s erre mit cselekedett. Ha a környezet modelljében lehetősége van a helyváltoztatásra, s erről tudomása van (megállapodásunkhoz híven az implementációban a környezet teljes modellje az ágens rendelkezésére áll), akkor kialakíthat egy *memória térképet*, amely azt mutatja meg, hogy hol és mit érzett, és ennek hatására mit cselekedett.

Ha ezen túl még az idő fogalma is megjelenhet a rendszerben, akkor a cselekedetekhez egy-egy végrehajtási időpontot is rendelhetünk. A következtetési (2) mechanizmus megvalósítása a kényszer logikai programozás elvén (Van Hentenryck, 1989) alapszik. Eszerint logikai állítások változóira tehetünk megkötéseket, s így azok nem nyerhetnek a kikötésnek ellentmondó behelyettesítést.

Most már pontosíthatjuk a lokális és a globális valóság kapcsolatát. Az ágens benső reprezentációja ugyebár a külső reprezentáció, illetve a globális valóság egyfajta modellje. Ha e modellt úgy tekintjük, mint elsőrendű logikai állítások halmazait, melyek változóira értékkényszerek vonatkoznak, akkor a kényszerek felelnek meg a kapcsolatoknak, míg az állítások az ágenseknek, objektumoknak, cselekvéseknek és folyamatoknak. Az 5.4. ábrán az ilyen jellegű modellek általános vázlata látható.

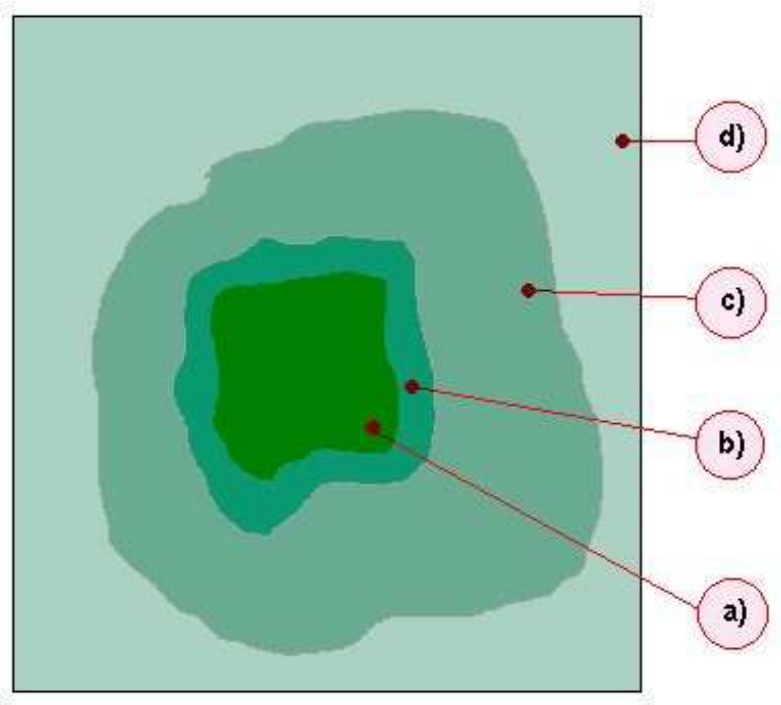


5.4. ábra: Kényszer-logikai modellek általános elve

Ha ismert, hogy a lokális valóságot leíró állítások változói, avagy az érzetek milyen kapcsolatban állnak a globális valóság különböző tényezőit leíró benső reprezentáció logikai állításainak változóival, attribútumaival, akkor adott lokális valóság esetén, azaz a lokális valóság – érzékelés folytán történt – adott behelyettesítése mellett a benső reprezentáció állításaiban szereplő változók érték-tartománya is szűkül, vagy adott esetben egyes változók – a definit kényszereknek köszönhetően – akár máris behelyettesítést nyerhetnek. Ekkor kiemelve a benső reprezentáció valamely érdekesnek tetsző állítás-halmazát, majd változóinak szisztematikus (és ellentmondásmentes) behelyettesítésével elérhető, hogy mind az említett állítások, mind pedig a velük kapcsolatos majd minden állítás behelyettesítődjék. A benső reprezentáció egy-egy ilyen behelyettesítését tekintjük fantáziának.

Tehát az ágens a fantáziákat tapasztalatai nyomán állítja elő oly módon, hogy a kapcsolatok révén összeegyezteti őket a benső reprezentációval. Minden új észlelés valamilyen újabb kényszert von magával a környezet benső reprezentációjára nézve, csökkentve az állítások attribútumainak „mozgási”- avagy értéktartományát, monoton redukálva a lehetséges behelyettesítések számát. Továbbá – **nagyon fontos** megállapodás, miszerint – csak azok a tényállítások jelennek meg a fantáziákban, melyekre történt „utalás” a tapasztalatok során, azaz amelyek változóira vonatkozó kényszer valamikor megjelent a rendszerben egy adott tapasztalásnak köszönhetően. Magyarán: *csak az jelenik meg a fantáziákban, amit az ágens érzékelt, és ami ezzel összefügg a kényszer-*

logikai kapcsolatok révén. Példának okáért képzeljük el azt, hogy ágensünk látás-érzékelő berendezése meghibásodott, ám hő-érzékelése még megfelelő. Tegyük fel továbbá, hogy az ágens környezetet leíró kapcsolatai közt szerepelnek azok a kényszerek, miszerint, ha meleget érzékel, akkor vagy tábortűz van a közelben, vagy egy radiátor. Ekkor, ha ágensünk meleget érzékel, két fantáziát fog előállítani: egyet amelyben a közelében tábortűz van, és egy másikat, amelyben egy radiátor van. Ha azonban ágensünk hallószervekkel is rendelkezik, és fülébe jut a tűz sercegése, továbbá rendelkezik olyan kapcsolattal, mely azt állítja, hogy, ha sercegést érzékel, akkor a közelben csak tábortűz lehet, akkor azok a behelyettesítések, melyek egy radiátor, egy radiátor és egy tábortűz meglétét állítják – inkonzisztenciájuk folytán – meghiúsulnának, s csak egyetlen behelyettesítés maradna, miszerint tábortűz van a közelben. Ily módon az ágens az említett kapcsolatok és érzékelő mechanizmusok, továbbá a környezet-modell birtokában képes volt kikövetkeztetni, hogy a meleg és sercegés együttes élménye a globális valóságban egy tábortűz meglétét jelenti a közelében. Amíg azonban az ágens nem érzékel se meleget, se sercegést, eszébe se jut, hogy tűz lehet a közelben, hiszen semmi sem utal rá. Egy tipikus fantáziát, avagy a „zöld-mező" effektust szemlélteti az 5.5. ábra.



5.5. ábra. Fantáziák tipikus behelyettesítettsége

A „zöld-mező" effektusnak nevezem – képletesen – azt, ha adott egy – az 5.5.ábrán látható – zöld mező, melyet aktuálisan – és szemléletesen – befed a köd. Az ágens dolga az, hogy feltérképezze, hogy hol milyen színű a fű. Az ábra a) része, a fantázia azon részét mutatja, amelynek megfelelő globális valóság már tökéletes bizonyossággal ismert, azaz teljesen behelyettesített, mivel a tapasztalatok egyértelmű utalást tettek rá vonatkozólag, avagy az ágens „bejárta"; b) a mező azon területe, amelyet az ágens még nem járt be, csupán csak látott, avagy

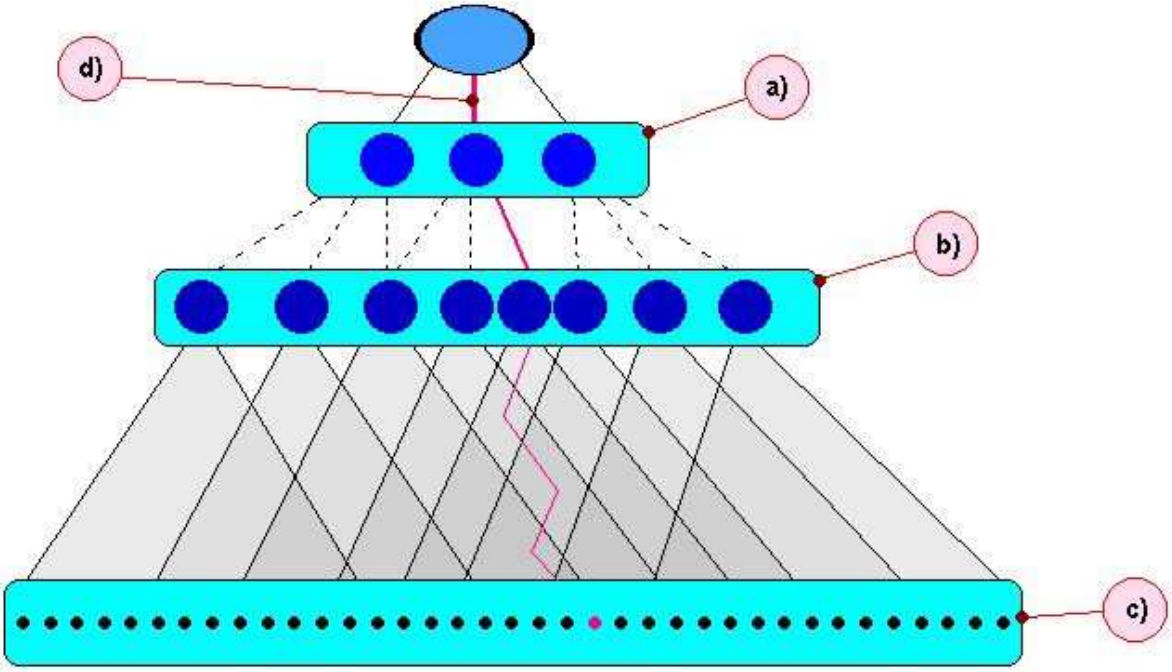
érzékelt. Sajnos a köd elmossa a színeket, így ágensünk nem tudja, hogy szürkére, vagy zöldre gondoljon, így behelyettesítetlenül hagyja azon terület a füvének színét leíró állítás megfelelő attribútumát. Tehát ezen a részen olyan állítások szerepelnek, melyek változóinak értékkészlete szűkített – ámde még nem behelyettesített – tartományban mozog annak következtében, hogy bár történtek rájuk vonatkozó érzékelések, ezek sajnos még nem voltak egyértelműek; a látott territórium után következik a látatlan, amelyet az ágens még nem tapasztalt meg c), de módjában áll olyan cselekvést kiviteleznie, mellyel közvetve érzékelheti. Például odaküldheti társát, hogy nézze meg, majd újságolja el számára. Ez a territórium csak akkor jelenik meg a fantáziában, ha legalább közvetve érzékeltük; ez után következik az a globális valóság azon része d), amelyről se biztos érzékelésünk – se többértelmű, ám közvetlen érzékelésünk – se semmifajta közvetett érzékelésünk nincsen. A globális valóság e territóriuma csak akkor jelenik meg a fantáziákban, ha valamilyen „utalás" történik rá, azaz legalább közvetve, de érzékeljük.

Ha az aktuális érzékelés, és az eddigi tapasztalatok folytán az ágens előállított egy benső reprezentációt, akkor abban a „gondolkodás" lényegi része, a cselekvések (avagy gondolatok) evolúciója a következőképpen zajlik: (1) az ágens a benső reprezentáció szerinti lehetséges elemi cselekvésekből GEP-kromoszómákat (lásd. 2.3.3) állít össze, ez lesz az algoritmus kezdő-populációja. A kromoszómák funkciói az érzékelések előállításáért felelős cselekvések, míg terminálisai az egyéb, például hely, vagy helyzetváltoztató, vagy tetszőleges más, nem érzékelő cselekvések lesznek. Továbbá, **nagyon fontos** végre konkrétan is kimondanunk, hogy egy-egy kész GEP-kromoszóma egy-egy feltételes tervet reprezentál, ahol a terv egy-egy feltételes lépése a GEP-gén-lánc egy-egy génje (lásd. 2.3.3). (2) az ágens a benső reprezentáció esetlegesen behelyettesítetlen részeinek minden lehetséges – kényszereknek megfelelő – behelyettesítését előállítja, majd ezeket súlyozza aszerint, hogy melyiket-milyen valóságosnak tekinti. (3) az előállított kromoszómákat úgy evolválja a fantáziákon, hogy egy-egy kromoszóma jóságát egy-egy generációban úgy állítja elő, hogy az adott kromoszóma által reprezentált cselekvést, tervet, avagy programot mindegyik fantázián lefuttatja, a futások eredményét pedig súlyozva összeadja. (4) miután lezajlott az ágensen belüli felügyelt evolúció, előáll az utolsó generáció legjobb egyede – ez lesz az ágens által környezete felé végrehajtandó terv.

Sajnos azonban nem garantált, hogy a cselekvés valóban azt eredményezi, amit az ágens a fantáziákban elképzelt. Még akkor sem, ha a környezet által reprezentált probléma nem felderíthetőségi, hanem eshetőségi. Ez azért lehetséges, mert a bizonytalanságok számában exponenciálisan sok fantáziát állíthatunk elő, s így heurisztikákra (pl. fantáziák valóságossága) van szükségünk ahhoz, hogy csökkentsük a cselekvések evolválásánál felhasználtak számát. Így hát elképzelhető, hogy – a környezet teljes modelljével dolgozva – azok között, amiket kihagyunk, épp ott rejlik az a fantázia, amelyik pontosan ekvivalens a globális valósággal, vagy legalábbis az általunk megtapasztalt részével, a külső reprezentációval. Az is elképzelhető, hogy nem hagytuk ki, csak nem vettük kellő súllyal figyelembe, úgymond elveszett a többi változat közt. Mindezek a

jelenségek – még egyszer hangsúlyozom – csak azért lehetségesek, mert eshetőségi problémákat reprezentáló környezetekben előfordulhatnak olyan érzeti benyomások, olyan lokális valóságok, melyek nem definiálják pontosan az őket kiváltó globális valóság állapotát. Gondoljunk csak vissza a tábortűz-radiátor példára. Ha csupán a meleget érezzük, nem tudjuk garantáltan eldönteni, hogy a közelben tábortűz, vagy radiátor van-e. A probléma eshetőségi, hiszen lehetséges, hogy mi radiátorra tippelünk, és odalépve rátesszük a fázós kezünket, miközben valójában tábortűz van ott, amely annak ellenére megégeti a kezünk, hogy a fantáziáinkban egy kellemesen melengető radiátoron pihentettük őket.

Az eshetőségek okozta bizonytalanság implikálja, hogy az evolvált tervek hossza nagyobb kell, hogy legyen, mint a belőlük ágensgondolkodási-ciklusonként végrehajtott cselekvések száma. Ezáltal egyfajta újratervező (lásd. 2.2.3.3), vagy még inkább beágyazott tervkészítő ágens architektúrát (lásd. 2.2.3.4) kapunk, amely megfelelő implementáció esetén akár még felderíthetőségi (jelenleg eshetőségi) problémák általános, „optimális" megoldására is alkalmas lehet azáltal, hogy egy két-szintű evolúció „sarok-pontja". Az evolúciós ágens egy-egy evolválási (gondolkodási) ciklusát szemlélteti az 5.6. ábra:



5.6.ábra. Az ágens evolúciós gondolkodási folyamatának szemléltetése

Az ábrán látható fastruktúra csúcsa a benső reprezentáció. Első szintje ***a)*** a benső reprezentáció minden lehetséges behelyettesítése, azaz a fantáziák. A fastruktúra *k.* szintje ***b)*** az *1.* szint valamely fantáziájából *(k-1).* cselekvés felhasználásával elérhető fantáziák halmaza. Az ábrán szereplő fa-struktúra olyan értelemben szemlélteti az ágens gondolkodási ciklusát, hogy a leveleihez, azaz a fastruktúra *N.* szintjéhez ***c)*** vezető összes lehetséges út az ágens egy adott gondolkodási ciklusának egy adott generációjában végrehajtott összes *(N-1)* hosszú tervnek feleltethető meg. Azaz egy-egy terv végrehajtása valamely fantáziában egy-egy utat reprezentál ***d)*** a

fastruktúrában, míg a közben előálló fantáziáknak az adott út fastruktúra-beli csúcsai feleltethetők meg.

Az egyes terveket többféle módon is megítélhetjük. Ennek részletezésére az implementáció megvalósítása kapcsán térünk majd ki. Most csak annyit mondanánk, hogy az 5.6. ábra egy-egy útjának értékelése egy-egy terv értékelésének felel meg. Azonban a fastruktúra nemcsak az evolúció során változik generációról generációra, hanem evolúcióról evolúcióra is változik. Ennek érzékeltetésére tegyük fel, hogy bár –az előbbieknek megfelelően – *(N-1)* hosszú terveket érlelünk, azokból az evolúció végeztével – akármelyik terv is bizonyul legjobbnak a végén – csak egy lépést hajtunk végre. Amikor a terveket készítettük, akkor az a fantázia, amely a végén választott terv-lépés végrehajtása következtében előállt, ott szerepel a fastruktúra második szintjén. Arra gondolnánk, hogy esetleg a következő evolúcióban ez lesz az új fastruktúra első szintjének valamely eleme, ha nem a gyökere (teljesen behelyettesített benső reprezentáció esetén); sajnos azonban ez legtöbbször (bonyolultabb környezetekben) nem igaz. Már eshetőségi problémát reprezentáló környezetben is előfordulhat az, hogy az adott lépés által előállított globális környezet interpretációjaként előálló benső reprezentáció egyik behelyettesítése sem felel meg a lépés által előzőekben előállított fastruktúra-beli fantáziának. Még az is lehet, hogy az új benső reprezentáció egyik behelyettesítése sem szerepelt az előző fastruktúra első szintjén. Így hát az ágens működése olyan, hogy egy-egy érzékelést követően mindig mélyen (pl. *(N-1)* lépésben) belegondol a környezet lehetséges jövőjébe, majd a legszimpatikusabb választást (már-már a játék-elmélet MINIMAX algoritmusához hasonlóan) meglépi. Sajnos, amíg pl. a MINIMAX-nál a környezet statikus kell, hogy legyen, azaz a probléma egyállapotú, magyarán mindig tudjuk, hogy cselekvéseinknek mi lesz az eredménye, s ez alapján „stratégiázunk”, addig esetünkben ez nem lehetséges. Miután megléptük azt a lépést, és „újragondoljuk” a fastruktúrát, még az is lehet – az előzőekben felvázolt gondolatmenet következményeként –, hogy egészen váratlan, előre be nem tervezett csúcsban „landolunk”. Így hát az ágens működése – még egyszer mondom – újratervező jellegű, amennyiben minden lépése után újra, és újra előretervez *(N-1)* lépést, majd lép, majd megint tervez, majd megint lép, sit.

Azok után, hogy kellően sok szemszögből megvizsgáltuk a megtervezett megoldást, rátérhetünk az általam megvalósított implementáció érdekesebb technikai részleteinek rövid összefoglalására.

5.3.2. Implementáció megvalósítása

Mivel a teljes megvalósítást Prolog-ban (Szeredi és Benkő, 2000) implementáltam, ezért az ágens tudás-bázisa (világra vonatkozó állításainak halmaza) logikai deklaráció formájában adott. A logikus következtetés már ennek alapján történik a korlát logikai programozás (Szeredi, 2002) felhasználásával. Ha tehát a „rejtett szabályszerűségek” logikai korlátok formájában adottak, akkor a múlt ismeretében, azaz tudva, hogy mikor-mi történt, képesek vagyunk megmondani, hogy a

jelenben még ismeretlen állapotú helyekre mi vonatkozik, esetleg még a pontos állapotot is képesek vagyunk megmondani. Erre lehet példa a következő: Adott egy 3x3-as négyzetrács. Az ágens csak a négyzetrács valamely négyzetében helyezkedhet el, mivel a négyzetrácsot falak veszik körül. Az ágens aktuálisan csak az általa elfoglalt négyzetet „érzékeli". Pontosabban csak azt tudja megmondani, hogy ahol éppen van, ott érez-e bűzt, vagy szellőt. A bűz annak a következménye, hogy valamely közvetlenül szomszédos cellában egy szörnyeteg lakozik. A szellő pedig árkot jelez valamely közvetlen szomszédos rácspontban. Értelemszerűen legfeljebb négy közvetlenül szomszédos cella, avagy rácspont lehetséges: a fenti, a lenti, a jobb és a baloldali. Továbbá igaz még az is, hogy a szörnyeteg által birtokolt cellában is bűz uralkodik. Ha ekkor a négyzetrácsot koordinátázzuk úgy, hogy vízszintes irányban lesz az X-tengely, míg függőleges irányban az Y, továbbá a „tábla" bal alsó sarkának koordinátái *[1,1],* akkor a „rejtett szabályszerűségek" logikai megfogalmazása a következő:

Ha szörny található *[I,J]*-ben,

akkor *[I,J]*-ben, **és** *[I-1,J]*-ben, **és** *[I+1,J]*-ben, **és** *[I,J+1]*-ben, **és** *[I-1,J]*-ben bűz érezhető.

Ha nincs szörny *[I,J]*-ben,

akkor **sem** *[I,J]*-ben, **sem** *[I-1,J]*-ben, **sem** *[I+1,J]*-ben, **sem** *[I,J+1]*-ben, **sem** *[I-1,J]*-ben nem érezhető bűz.

Ha árok található *[I,J]*-ben,

akkor *[I,J]*-ben, **és** *[I-1,J]*-ben, **és** *[I+1,J]*-ben, **és** *[I,J+1]*-ben, **és** *[I-1,J]*-ben szellő érezhető.

Ha nincs árok *[I,J]*-ben,

akkor **sem** *[I,J]*-ben, **sem** *[I-1,J]*-ben, **sem** *[I+1,J]*-ben, **sem** *[I,J+1]*-ben, **sem** *[I-1,J]*-ben nem érezhető szellő.

Továbbá $1 \le I, J \le 3$ kell, hogy legyen.

Mindebből az ágens a következő logikai korlátokat állítja elő:

Ha bűz érezhető *[I,J]*-ben,

akkor **vagy** *[I,J]*-ben, **vagy** *[I-1,J]*-ben, **vagy** *[I+1,J]*-ben, **vagy** *[I,J+1]*-ben, **vagy** *[I-1,J]*-ben szörnyeteg lakozik.

Ha nincs bűz *[I,J]*-ben,

akkor *[I,J]*-ben, **és** *[I-1,J]*-ben, **és** *[I+1,J]*-ben, **és** *[I,J+1]*-ben, **és** *[I-1,J]*-ben sincs szörny.

Ha szellő érezhető *[I,J]*-ben,

akkor **vagy** *[I,J]*-ben, **vagy** *[I-1,J]*-ben, **vagy** *[I+1,J]*-ben, **vagy** *[I,J+1]*-ben, **vagy** *[I-1,J]*-ben árok található.

Ha nincs szellő *[I,J]*-ben,

akkor *[I,J]*-ben, **és** *[I-1,J]*-ben, **és** *[I+1,J]*-ben, **és** *[I,J+1]*-ben, **és** *[I-1,J]*-ben sincs árok.

Továbbá $1 \le I, J \le 3$ kell, hogy legyen.

Ezeket a logikai állításokat korlátként értelmezve, a játék-tér cellái közti összefüggések leírásához jutottunk. Kezdetben az ágens emlékezete még nem tartalmaz információkat a cellák állapotára vonatkozóan, így minden *[I,J]* érték esetén behelyettesítetlen az, hogy van-e az adott cellában árok, vagy szörnyeteg. Ahogy az ágens lassanként bejárja a játék-teret, tapasztalatokat gyűjtve, tudás-bázisa is bővül. Egy ilyen tudásbázis lehet pl. a következő:

***[2,3]*-ban nincs se bűz, se szellő.**
***[1,2]*-ben van bűz, de nincs szellő.**

A fenti két állításból az ágensnek ki kell tudnia következtetni, hogy a szörny vagy *[1,1]*-ben, vagy *[1,2]*-ben helyezkedik el. Ha *[2,3]*-ban nincs se bűz, se szellő, akkor a logikai korlátok következtében **se** *[2,3]*-ban, **se** *[1,3]*-ban, **se** *[3,3]*-ban, **se** *[2,2]*-ben nincs se bűz, se szellő. Azonban *[1,2]*-ben van bűz, s így **vagy** *[1,2]*-ben, **vagy** *[2,2]*-ben, **vagy** *[1,1]*-ben, **vagy** *[1,3]*-ban van szörny. Mivel tudjuk, hogy se *[1,3]*-ban, se *[2,2]*-ben nem lehet szörny, ezért „szűkítjük" a szörny lehetséges előfordulásainak tartományát *[1,1]-re* és *[1,2]-re*. Tehát amíg nem egyértelmű, hogy mi a szörny tartózkodási tartománya, addig az ágens az esélyes elemeket bizonytalannak tekinti, míg az egyértelműen nemlegeseket kizárja. Ha most az ágens tovább vándorolva eljut az *[1,3]*-as cellába, ahol azt tapasztalja, hogy nincs **se** bűz, **se** szellő, akkor kizárva az *[1,2]*-es cellát a szörny lehetséges előfordulásaiból, a lehetséges tartomány *[1,1]*-re zsugorodik, amiből az következik, hogy a szörny az *[1,1]*-es cellában található.

Ha az ágens a „*rejtett összefüggéseket*" helyesen „*hámozta ki*" a lokális valóságból, akkor, ha a környezetben nincs ellentmondás, az ágensnek is ellentmondásmentes, egyértelmű eredményre kell jutnia. Ha vagy a környezetben, vagy az ágens által kihámozott „*rejtett összefüggésekben*" az ellentmondás lehetősége rejlik, akkor előbb-utóbb vagy az emlékezet, vagy az emlékezet alapján következtetett tudásbázis ellentmondásossá válik. Ez történik akkor, ha például az ágens a *[2,1]*-es cellába érve nem tapasztal bűzt, vagy, ha a környezet ellentmondásos, akkor az addigi tapasztalatokkal kerül ellentmondásba a mostani észlelés. Mindkét esetben vagy frissítenie kell a tudásbázis egy részét, vagy be kell vennie azt a speciális esetet, hogy bár *[1,2]*-ben bűz uralkodik, és szörny csak az *[1,1]*-ben lehetne, most mégsincs, azaz egy *„[I,J] = [1,1]-ből következik, hogy [I,J]-ben, és [I+1,J]-ben nincs szörny"* alakú kivételt.

A Prolog CLP(FD) könyvtárát (Szeredi, 2002) használtam a logikai következtető-rendszer megalkotásakor, amely lehetővé teszi, hogy a fentebb említett logikai következtetést általános értelemben kiterjesszük úgy, hogy az ágens bárminemű következtetést megtehessen a környezettel kapcsolatban a következő általános elv használatával: Tegyük fel, hogy a lokális valóság aktuális állapota logikai deklaráció formájában adott, melynek *N* argumentuma van, melyek értéke rendre *IGAZ*, vagy *HAMIS*, annak megfelelően, hogy az adott argumentumnak megfelelő érzet fennáll-e. Ez megfelel egyfajta bináris kódolásnak, ahol a *k*. bit jelzi, hogy a *k*. típusú érzet fennáll-e. Minden – legyen bármily komplex – lokális valóság „lekódolható" ilyen módon. Ha ekkor felírunk 2N szabályt, ahol az első N darab az N argumentum által meghatározott érzetek meglétének világbeli

következményeit írja le rendre úgy, hogy megadja, hogy, ha az adott érzet fennáll, akkor minek kell, és minek nem kell léteznie a globális valóságban, míg a második N darab ellenkezőleg, az érzet hiányának következményeit írja le, akkor egy **általános korlát bázist** kapunk, amelyből a CLP következtető-gép már automatikusan képes meghatározni a biztosan fennálló, a bizonytalan és a nem létező globális világ béli tényeket!

Látható, hogy, mivel az általános korlát bázis előállítása kézenfekvő módon automatizálható, az ágens képes lesz annak önálló, meglévő tudásbázishoz igazított megalkotására, így biztosítva a logikus következtetést általános esetben is. Felmerülhet a kérdés, hogy:

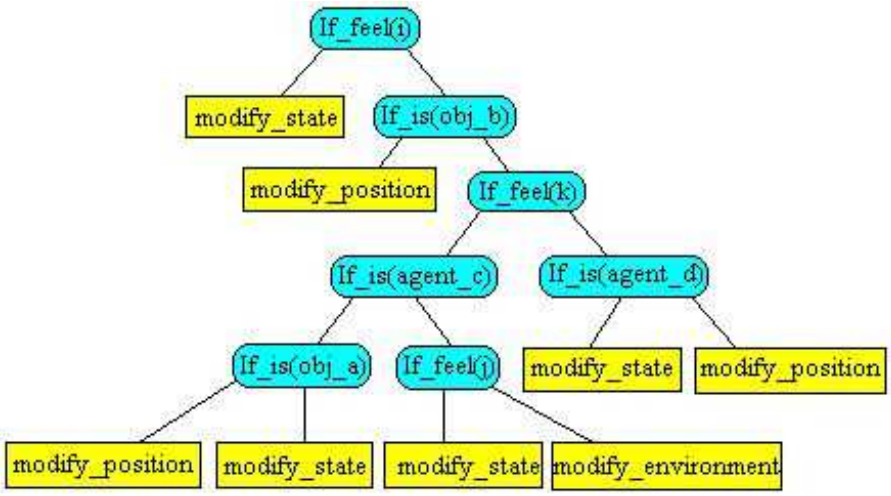
- *Mi értelme van egy-egy ágenst ilyen „komoly" logikai következtetéssel felruházni?*

A válasz szinte kézenfekvő. Ugyebár az ágens benső reprezentációja tartalmaz objektumokat és más ágenseket, ezért értelmes lehet olyan elemi cselekvések előállítása, melyek a globális valóság állapotában ezek meglétét firtatják. Érdemes lehet azt kérdezni, hogy: *Van-e előttem szörny, vagy árok? Az eddigi tapasztalatok alapján, merre lehet a legközelebbi erőforrás lelőhely?…stb.* Tehát az ágensnek máris adatik az objektumok és ágensek különböző típusainak számával azonos számú elemi cselekvési lehetősége, melyek azt vizsgálják, hogy az adott objektum, vagy ágens megléte következik-e az eddigi tapasztalatokból a globális világ adott pontjára vonatkozólag, vagy sem. Ez egy „kognitív" jellegű cselekvés. Az ágens által végezhető elemi cselekvéseket a következőképp osztályozhatjuk:

- **Vizsgáló-cselekvések**
  - *Érzékelő*
  - *Kognitív*
- **Módosító cselekvések**
  - *Hely-módosító*
  - *Helyzet-módosító*
  - *Környezet-módosító*

Az érzékelő cselekvések a lokális valóság adott érzetének meglétét vizsgálják. A kognitív cselekvések a globális valóság kikövetkeztethető tényeinek meglétét vizsgálják. A módosító cselekvések pedig az ágens helyét, helyzetét, illetve környezetének állapotát képesek módosítani.

Gondoljuk csak meg, mi történik akkor, amikor az ágens evolválja az alteregóit! A fentebb említett elemi cselekvésekből állít össze feltételes cselekvés-sorozatokat úgy, hogy, - ha visszaemlékszünk a GEP algoritmus leírására (Ferreira, 2001), akkor jelen esetben – a nem-terminálisoknak a vizsgáló cselekvéseket felelteti meg, míg a terminálisoknak a módosító-jellegű cselekvéseket. Megvalósításomban az ágens kromoszómákat érlel, amelyek adott számú génből állnak össze. Egy-egy gén egy-egy feltételes cselekvést reprezentál. A kromoszóma, avagy alteregó jóságát az ágens a gének egymás utáni lefuttatásával, majd a végállapot, és a futás paramétereinek megítélése alapján dönti el, ahol egy-egy gén a következő programhoz hasonló struktúrájú:



5.7. ábra: Példa egy döntési fa formájában adott feltételes cselekvésre

Az ábra magyarázata: Az ábrázolt fa-struktúra egy feltételes cselekvést ábrázol, ahol a benső (*kék*) csúcsok vizsgáló cselekvéseket reprezentálnak, míg a fa levelei (*sárga csúcsok*) módosító cselekvéseket. Adott lokális valóság esetén, ha a fának megfelelő programot végrehajtjuk, egyetlen módosító cselekvés fog végrehajtódni. Az ábrázolt fának megfelelő pszeudo-kód a következő:

```
IF FEEL(I) THEN
        MODIFY STATE
ELSE
        IF IS OBJECT(B) THEN
                MODIFY POSITION
        ELSE
                IF FEEL(K) THEN
                        IF IS AGENT(C) THEN
                                IF IS OBJECT(A) THEN
                                        MODIFY POSITION
                                ELSE
                                        MODIFY STATE
                        ELSE
                                IF FEEL(J) THEN
                                        MODIFY STATE
                                ELSE
                                        MODIFY ENVIRONMENT
                ELSE
                        IF IS AGENT(D) THEN
                                MODIFY STATE
                        ELSE
                                MODIFY POSITION
```

Látható, hogy a fa *Top-Down* jellegű bejárása történik, azaz elindulunk a fa gyökeréből, végrehajtjuk a gyökér-elem által reprezentált cselekvést, majd a cselekvés kimenetelének megfelelően haladunk tovább a fa ágain, mígnem egy levélhez nem érünk. A fenti feltételes cselekvés zárójelezett alakja:

```
if_feel_i(
        modify_state,
        if_is_obj_b(
                modify_position,
                if_feel_k(
                        if_is_agent_c(
                                if_is_obj_a(
                                        modify_position,
                                        modify_state
                                ),
                                if_feel_j(
                                        modify_state,
                                        modify_environment
                                )
                        ),
                        if_is_agent_d(
                                modify_state,
                                modify_position
                        )
                )
        )
)
```

A zárójeles alak megfelel egy szabványos Prolog-kifejezésnek, melyben a vizsgáló cselekvések „`If_Xy(A, B) :- Xy -> A; B.`" alakú klóz-deklarációk, ahol a két attribútum, A és B változó, akár egy hasonló kifejezés is lehet a behelyettesítést követően.

Ha tehát a prolog-interpreter a fenti zárójeles kifejezést kapja meg úgy, hogy a feltételek az előbb említett formában lettek deklarálva, akkor a kifejezés által képviselt program végrehajtása *Top-Down* módon fog történni, azaz a következőképp:

1. A Prolog veszi az IF_FEEL_I(A,B) kifejezést, ahol A és B változók.
2. A-nak a MODIFY_STATE, B-nek pedig a IF_IS_OBJ_B(C,D) behelyettesítést adja a mintaillesztést követően.
3. Az így előálló IF_FEEL_I(MODIFY_STATE, IF_IS_OBJ_B(C,D)) fejet lecseréli a klóz törzsével, azaz (I -> MODIFY_STATE; IF_IS_OBJ_B(C,D))-vel, majd az így kapott kifejezést meghívja.
4. A kifejezés jelentése az, hogy, ha I igaz, akkor a feltétel első tagjával helyettesíti az eddigieket, ha pedig nem igaz, akkor a második taggal. Így, ha I mondjuk igaznak bizonyul, akkor az egész eddigi kifejezés redukálódik egy sima MODIFY_STATE kifejezéssé.
5. A Prolog meghívja az újonnan előállt kifejezést, aminek már az implementációjától függ a további kimenetel. Tegyük fel, hogy azzal, hogy meghívjuk ezt a kifejezést, mellékhatásként megváltozik az ágens állapota a globális valóságban…

Ha például LISP-et használtunk volna a fenti zárójeles kifejezés kiértékelésekor, akkor a LISP interpreter először a fa-leveleivel kezdte volna, azaz a fa-struktúrát, mint egy összetett függvény-hívást, lentről fölfelé, *Bottom-Up* módon értékelte volna ki, ami ugyebár nem felel meg elvárásainknak. *Mégis, miért használjuk a fa-struktúrával történő cselekvés-leírást?* – kérdezhetnénk. A válasz egyrészt a GEP (lásd. GEP) megvalósításában rejlik, hiszen az ott használt kromoszóma-dekódolás során hasonló, ún. prefix-zárójeles struktúrák jönnek létre, amik

természetszerűleg reprezentálhatók a fent ábrázolt fához hasonló struktúrákkal, másrészt meg azért, mert így kézenfekvő egy „robot-jellegű” ágens cselekvéseinek a leírása. A robot irányítása ugyebár *HA-AKKOR* jellegű utasításokból áll. Ezek megjelenítésére tökéletesen alkalmas a fenti, *Top-Down* módon kiértékelt fa-struktúra.

Máris eljutottunk addig, hogy képesek legyünk megindokolni a Prolog – mint az implementációra kijelölt nyelv – választásának okát. Azért a Prolog-ot tartom a legalkalmasabbnak az „ágens-evolúciós” elv megvalósítására, mert:

1. Vagy a deklaratív (Prolog, Mercury…), vagy a funkcionális (LISP, SML…) nyelvek jelentenek alternatívát akkor, ha az imperatív program-nyelvek kínálta probléma-leírásnál sokkalta tömörebb kódot szeretnénk.
2. Alapjában véve, mind a deklaratív, mind a funkcionális nyelvek generikusabbak az adatstruktúrák reprezentálásakor, mint imperatív vetélytársaik (hiszen nem típusosak), s így könnyebben megvalósítható általunk egy GEP-jellegű program-evolúció.
3. A Prologot mégis azért preferáltam a többi, például funkcionális (és persze imperatív) nyelvekkel szemben, mert a mintaillesztésnek köszönhetően képes ***(1)*** visszafelé következtetni. Azaz, ha adott egy f(A,B) :- … alakú klóz, ami gyakorlatilag egy egy-változós függvénynek felel meg funkcionális esetben, ahol B = f(A), akkor a Prolog, ha a klóz törzse megfelelően lett megírva, képes arra, hogy, ha csak az egyik változó értéke adott, kikövetkezteti a másikat, azaz A-ból B-t, és viszont. Ezzel egy csapásra megoldott a függvény inverzének implementációja is… ***(2)*** Képes több bemenetű, több kimenetű függvények kezelésére úgy, hogy a bemenetek és kimenetek halmaza akár tetszőlegesen választható, hiszen, ha adott egy f(A1,…,An) :- … alakú klóz, akkor, ha a klóz törzse megfelelően lett implementálva, {A1,A2,…,An} halmaz akár tetszőleges részhalmazának ismeretében meghatározható a többi változó értéke (ha ez lehetséges). Alapjában, ha például Ai-ket úgy indexeljük, hogy az első k-darab legyen behelyettesített, akkor az említett klóz egy k-bemenetű, (n-k) kimenetű függvénynek felel meg. ***(3)*** A Prolog képes arra, hogy, ha egy változónak több behelyettesítése is lehetséges, akkor visszalépések során felsorolja az összes lehetséges megoldást. Azaz f(*A*) deklarációra alapjában *A* összes lehetséges behelyettesítését megadja, s így implementálhatók úgymond a „több megoldást adó” függvények is! ***(4)*** Már említettem, hogy a Prolog a mintaillesztésnek köszönhetően az interpretálandó klózokat *Top-Down* módon értelmezi logikai feldolgozó-mechanizmusának köszönhetően. Ez, mivel GEP-alapú evolúciós algoritmust alkalmazok (*hiszen ez a genetikus programozási technikák jelenleg legújabb és leghatékonyabb változata*), bár részlet-kérdésnek tűnhet, mégis nagyban hozzájárul ahhoz, hogy a Prolog-ot részesítsük előnyben az egyetlen valódi vetélytárssal, a LISP-el szemben.

4. Továbbá, a Prolog-nak két, számunkra igen fontos kiterjesztése is létezik: A *Definite Clause Grammars* (DCG), és a *Constraint Logic Programming* (CLP) kiterjesztések. Amíg az előbbi akár környezet-függő (!) nyelvek feldolgozását is lehetővé teszi, addig a CLP egy robosztus, tudományosan teljesen megalapozott, kurrens és fejlődésben lévő logikai következtető rendszer. Ráadásul ezen alapszik az ágens megvalósítási tervezete.
5. Végül meg kell említenem, hogy – bár nem annyira közismert, de – a Prolog a Java-val általában összemérhető futásidőt produkál.

A fenti indokok a Prolog mellett szólnak, de: *Mi szól ellene?* Egyrészt a Prolog grafikus interfészei önmagukban még viszonylag kezdetlegesek, bár C-vel, és Java-val is integrálható. Sajnos az integráció sokszor nehézkes és lassítja a program futását. Programom egyébként a Prolog Tcl/Tk (*Tickle-Teekay*) script-nyelvi kiterjesztését használja a grafikus megjelenítés céljából. Másik nehézsége a Prolog-nak a program-nyelv sajátos jellegéből fakad – a Prolog teljesen rekurzív, s így általában igencsak nehezen oldható meg benne az imperatív gondolkodás érvényesítése, a ciklusok behozatala, illetve a programok futásának nyomkövetése, azaz a *debuggolás*.

Összefoglalva, ebben a fejezetben a Prolog, az evolúciós módszerek, különösképp a GEP algoritmus, a haszon-orientált absztrakt ágens architektúrák, az eshetőségi tervkészítők és a beágyazott ágensek elveinek ötletes szintézise nyomán létrejött új konkrét tervkészítő ágens architektúra implementációjának főbb érdekességeit mutattuk be. A következőkben a megvalósítás tesztelését, a tesztelésekkel kapcsolatos nehézségeket és a teszteredmények értékelését foglaljuk össze.

6. Elemzés

Munkám kezdetekor azt a célt tűztem ki magam elé, hogy a manapság közhasználatban lévő, intelligens tervkészítőknél jobbat, általánosabbat és hatékonyabbat készítsek egy olyan, saját elképzelés alapján, amely képessé teheti a rendszert arra, hogy – kellő megvalósítás mellett – akár még a kilátástalan komplexitású problémák esetén is képes legyen az „optimális" tervkészítésre, s így a valós környezetekhez hasonló bonyolultságú tartományokon belüli „valóban intelligens" döntéshozatalra. A következőkben e célkitűzés eredményességét elemzem.

6.1 A TESZTELÉS NEHÉZSÉGEI

A tesztelés nehézsége elsősorban annak a következménye, hogy „soft-computing" módszerekre, azaz heurisztikákra támaszkodunk. Ekkor a paraméterezés megfelelő megválasztása jelenti a legnagyobb gondot, mivel nagyban befolyásolja az algoritmus futási jellemzőit, de főkent eredményességét. Sajnos a paraméterek megválasztása legjobb esetben is csak tapasztalati úton történhet. A másik nehézséget a tesztelésre szolgáló feladatok megválasztása jelentette, hiszen az „intelligens viselkedés" és „optimális tervkészítés" mérésére nincsenek abszolút mércék. Nehézséget okozott továbbá az is, hogy a releváns tesztelés elvégzéséhez szükséges gépidő komolyabb feladatok esetén hónapokban mérhető.

Elsőként a paraméter-választást befolyásoló tényezőket vegyük sorra. A következő paraméterek megválasztására van szükség egy-egy GEP-alapú evolúcióhoz (lásd. 2.3.3):

- Generációk száma
- Populáció mérete
- Kromoszóma hossza
- Egy-egy gén fejhossza
- Mutációs valószínűség
- IS-áttevődés valószínűsége
- RIS-áttevődés valószínűsége
- Gén-áttevődés valószínűsége
- Egy-pontú kereszteződés valószínűsége
- Két-pontú kereszteződés valószínűsége
- Gén-kereszteződés valószínűsége

A fenti 11 paraméter megválasztása kulcsfontosságú egy-egy futás alkalmával. A legtöbb paraméter hatása sejthető, ám vannak olyanok is, amelyek hatásáról csak homályos elképzeléseim voltak a tesztelés során. Lássunk egy felsorolást az egyes paraméterek hatásairól:

1. A ***generációk számának*** növekedtével, mivel a GEP monoton konvergens, az optimális megoldás valószínűsége is nő.
2. A ***populáció mérete*** biztosítja a sokrétűséget, s így a jó „séma” felbukkanásának lehetőségét. Természetesen egy nagyobb populációban könnyebb „elveszni”, azaz előfordulhat, hogy, mivel a jó „séma” jósága elenyészik az össz-jósághoz képest, ezért a rulett-kerék alapú szelekció során az adott sémát képviselő egyed kimarad a következő populációból.
3. A ***kromoszóma hossza*** az ágens által előállított feltételes tervek hosszának felel meg, hiszen egy-egy gén egy-egy feltételes cselekvés. Ha a kromoszómát túlontúl nagyra választjuk, akkor, bár az algoritmus igencsak lelassul, a jó megoldás valószínűsége számottevően megugorhat. A rövid kromoszóma általában alacsonyabb komplexitású megoldást jelent, s így a megoldás minősége is behatárolódik.
4. A ***gén fejhossza*** arányos a gén által kódolt feltételes cselekvés komplexitásával, kifejező-erejével. Azonban a hosszabb géneknél általában nagyobb arányban vannak a kihasználatlan szegmensek.
5. A ***mutációs*** valószínűség, mivel ugyebár egy-egy „bit”-re vonatkozik, nem függ explicite se a gén, se a gén-lánc hosszától. Általában az egyszázad alatti érték javasolt.
6. Az ***IS áttevődés*** szignifikánsan átformálja az adott gén által kifejezett feltételes cselekvést. Nem érdemes túl nagynak választani bekövetkezi valószínűségét, mivel akkor a jó sémának nem lesz lehetősége megmaradni, és elterjedni a populációban, hiszen egy-egy IS meghívás rendre szétrombolja majd.
7. A ***RIS áttevődés*** a legveszélyesebb, mégis sokszor a leghasznosabb GEP operátor. Teljes egészében átformálja a gén által reprezentált cselekvést, s így adott esetben szükséges lehet ahhoz, hogy a programok által reprezentált, kevéssé folytonos keresési térben lökést adjon az evolúciónak, azaz kibillentse a lokális optimumból.
8. A ***gén-áttevődés*** a RIS áttevődés gén-lánc szintű megfelelőjének tekinthető. Ha a gén-láncot szekvenciálisan hajtjuk végre, akkor igen nagy változásokat tud okozni egy-egy kromoszóma jósági értékében. Igazan csak hosszú kromoszómák esetén érdemes nagyobbnak választani, egyébként az egytized körüli érték javasolt.
9. Az ***egy-pontú kereszteződés*** kulcs-fontosságú. Ez az operátor örökíti át a jó sémákat a populáció egyedei közt, ez biztosítja a séma terjedését. Érdemes úgy választani, hogy szinte minden kiválasztásra kerülő egyed-páros kereszteződjék, azaz 0.85 és 1 közötti érték szokott általában alkalmasnak bizonyulni.
10. A ***két-pontú kereszteződés*** túl nagy valószínűséggel történő bekövetkezése igencsak megzavarhatja az evolúció menetét. Egy-pontú rokonával ellenben megvan az a jó tulajdonsága, hogy képes a kromoszómának csak egy szeletét lecserélni. Ez gyakorta hatékonyabb, mint ha a töréspont utáni egész szakaszt lecserélnénk.

11. A ***gén-rekombináció*** a kereszteződés gén-lánc szintű megvalósulásának tekinthető avval az eltéréssel, hogy itt, mivel egy-egy gén cseréjének a valószínűségéről van szó, nem szakaszokat, hanem tetszőleges gén-pozíciókat cserélünk fel. Csak hosszabb kromoszómák esetén érdemes nagynak választani, mivel akkor a többi operátor nem feltétlen garantálja a kellő genetikai sokrétűséget. Sajnos azonban nem szabható meg, hogy mely részeket cserélje, így egy adott valószínűség mellett, egyenletes eloszlással választja ki a cserélendő géneket. Ha értéke például ½-ed, akkor a gén-lánc fele találomra ki lesz cserélve más elemekre, ami nagyban megnehezítheti a konvergenciát.

Természetesen a tervkészítő módszer felhasználója részéről nem várhatjuk el, hogy a fenti paramétereket ismerje, s így képes legyen kellő körültekintéssel beállítani őket. Az evolúciós ágens elve tehát az egyes ágenseken belüli GEP-alapú program-evolúciók paraméterezését automatikusan – az ágensek génjeiben kódolva –, maga oldja meg. Sajnos a releváns tesztek elvégzéséhez e paraméter-gének különböző változataival rendelkező ágensek igen nagy száma lehet szükséges. A tesztek során az ágensek többnyire 50 generációval, 31 fős populációval, változó kromoszóma és gén-hosszal, 0.01-es mutációs, 0.1-es IS és RIS áttevődési, változó gén áttevődési, 0.85-ös egy-pontú kereszteződési, 0.2-es két-pontú kereszteződési és változó gén-rekombinációs valószínűségekkel dolgoztak.

A másik problémát a tesztelési feladatok megválasztása jelentette. Két különböző alapfeladattal dolgoztam a tesztek során. Mindkettő mikro-világ. Bár egyik se volt a valósághoz mérhető komplexitású, mégis jellemzően olyan viselkedés-mintákat kívánt meg az ágensek részéről, melyek az eshetőségi és felderíthetőségi problémák megoldásához elengedhetetlenek lehetnek egy adott bonyolultsági szint fölött. Vizsgáltam a rendszer futási paramétereinek alakulását, illetve az előállított megoldások (tervek) „optimalitását”. Ezeket a következő fejezet részletesen ismerteti.

A releváns teszteléshez szükséges gépigény jelentette a harmadik, ámde legkevésbé elméleti jellegű nehézséget. Ha meggondoljuk, akkor 11 szabad paraméter esetén, egy-egy paraméter 3-3 értékével dolgozva csupán, 2 feladattal és feladatonként 3 variációval, egy-egy ágensnek 50-100 lépést engedélyezve és minden egyes futtatást 10-szer megismételve máris kb. $3^{11} \cdot 2 \cdot 3 \cdot 100 \cdot 10 \approx 10^9$, azaz *egymilliárd* (!) felügyelt evolúció lefuttatására lenne szükségünk a releváns, minden variációt lefedő tesztek elvégzéséhez. Ehhez becsléseim szerint éves nagyságrendű időre lenne szükség egy Pentium 4-es típusú, momentán csúcskategóriájú számítógépen. Szerencsére a feladat természeténél fogva jól párhuzamosítható, így a tesztelést az MIT tanszék Mesterséges Intelligencia laborjának gépein, és egy saját géppark felhasználásával, párhuzamosan végeztem el. Természetes a tesztelés még így se lett volna elvégezhető, ha nem szabok ésszerű korlátot a feladatok típusára és a releváns paraméter-tartományra vonatkozóan. Továbbá a futtatás ütemezése és a statisztikák feldolgozása ekkora adatmennyiségnél manuálisan már nem megoldható feladat. Ezért script-jellegű Prolog-programokat írtam a tesztelés automatizálása érdekében. Lássuk, hogyan zajlottak a tesztek!

6.2 A TESZTELÉS MENETE

Egy új ágens-architektúra tesztelése nem kézenfekvő feladat, mivel az ágens-környezetek empirikus vizsgálatára nem léteznek még jelenleg tesztelési szabványok. Egy-egy rendszerhez külön-külön kell kialakítani a megfelelő teszt-környezetet, tesztelési lehetőségeket.

Mivel a feladat-kiírás „optimális" tervkészítést kíván meg, ezért megoldásomat egyrészt az eddigi tervkészítő módszerekkel hasonlítom össze képességek és lehetőségek tekintetében. Másrészt, mivel ágens-architektúráról van szó, az eddigi megközelítésekkel való rövid összevetés sem maradhat el. A gyakorlati tesztelés során sajnos más módszerekkel nemigen állt módomban összevetni az eljárást, mivel egyrészt az egyes módszerek saját implementációja a diploma-terv megvalósítására szánt idő alatt nem megoldható (egy-egy ilyen tervkészítő eljáráson általában évekig dolgozik egy-egy kutatócsoport). Másrészt az alkalmas, elviekben közel azonos képességű rendszerek nemigen érhetők el szabadon, mivel többnyire pénzes állami, vagy katonai kutatási projektek végtermékeiről van szó.

Így hát – alkalmas gépi vetélytárs hiányában – az eljárás gyakorlati tesztelését külön kialakított mikro-világokon végeztem, a megoldások jóságát és a futási paraméterek alakulását vizsgálva. A gépi vetélytárs hiánya azonban nem tántorított el: olyan mikro-világokat választottam, amelyek megfelelő kezelői felülettel ellátva már egy ember számára is kellő kihívást jelentenek. Így módszerem az emberi intelligenciával, az emberi (nem feltétlen tökéletesen logikus, sokszor intuitív) döntéshozatallal szemben is megmérettetésre került. Az ilyen típusú tesztek gondos, körültekintő tervezést igényeltek ahhoz, hogy eredményeik releváns adatokat szolgáltassanak az eljárás jóságára vonatkozólag. – Összegezve, a következő teszteket valósítottam meg:

1. *Más módszerekkel való összevetés*
2. *Mikro-világok*
    a. *Tábla világ*
        - *Ágens kooperativitás*
    b. *Wumpus világ*
        - *Ágens-Ágens ellen*
        - *Ágens-Ember ellen*

6.2.1. Összevetés az eddigi módszerekkel

A legkezdetlegesebb módszer a 2.2.2.1 szakaszban tárgyalt RRT. Míg a részben rendezett tervkészítés alapváltozata csak egyállapotú problémákat képes kezelni, addig módszerem elviekben akár a legnehezebb probléma-osztály, a felderíthetőségi problémák (lásd. 2.2.1) megoldására is alkalmas lehet. Az RRT-nél adott a környezet és cselekvések teljes leírása (azaz a feladatkiírásban szereplő **SÉMÁK**). Módszerem esetében az evolúciós ágens a cselekvés és a környezeti működés sémáit maga állítja elő az interpretációs fázis során. Az RRT a tervek „párhuzamosak" és (a tervbeli

kényszereknek megfelelően) tetszőlegesen sorba-rendezhetők. Az én módszerem esetében a tervek programok, avagy feltételes cselekvéseket megvalósító döntési fák sorozatai, amelyekben szerepelhetnek ÉS/VAGY funkció-szimbólumok is, lehetővé téve a párhuzamos működést. Az RRT nem képes a negáció, diszjunkció és feltételes következmények, illetve feltételes cselekvések, továbbá hierarchikus tervek megvalósítására. Módszerem ezek mindegyikére alkalmas: a negáció és a diszjunkció a konjunkcióval egyetemben funkciónak tekinthető, s így beintegrálható a GEP kromoszómák által reprezentált tervekbe. Ezáltal módszerem képes az RRT-DUNF (lásd. 2.2.2.2) tervleírásának megvalósítására. Mivel a cselekvések feltételes döntési fák, ezért gyakorlatilag az FRRT-vel (lásd. 2.2.2.4) egyenértékű tervek készíthetők. A HD-RRT (lásd. 2.2.2.3) eljáráshoz hasonló tervkészítés valósítható meg, ha a terveket reprezentáló kromoszómák génjeit terminálisoknak tekintjük. Se az RRT, se az RRT-DUNF, se a HD-RRT, se az FRRT nincs felkészítve eshetőségi problémák megoldására. Egyedül az FRRT tud többállapotú problémákat megoldani. Módszerem az eshetőségi problémákon túl elviekben a felderíthetőségi módszerek általános és „optimális" kezelésére is fel van készítve. A HD-RRT esetében újfent adottak az absztrakt cselekvéseket dekomponáló módszerek (**SÉMÁK**). Módszeremnek nincs szüksége dekompozícióra, mivel döntési fák teljesen rendezett sorával reprezentálja a terveket, ahol – túl azon, hogy az egyes terminálisok akár géneknek, sőt egész génláncoknak feleltethetők meg – az evolúció során automatikusan születik meg a részcélok „optimális" megvalósítása, azaz módszerem ebben az esetben is képes a tervkészítés módszereinek (**SÉMÁINAK**) automatikus, önálló megtalálására. Visszatérve az FRRT-hez, a 2.2.2.4. szakaszban látottak szerint az algoritmus készen kapja az érzékelő cselekvéseket. Módszerem elviekben ezek megtalálására is alkalmaz, hiszen a nem-felügyelt ágens-evolúció során a különböző, egymással versengő ágenseknek más, és más az érzékelési, interpretációs és döntési mechanizmusa. A HD-RRT-ben az absztrakt cselekvésekből lesz a kezdeti terv, míg, ha meggondoljuk, az emberi intelligencia esetében ez két lépcsőre bontható: először az elemi cselekvések alakulnak ki, majd állnak össze absztraktabb, átfogóbb cselekvéseket reprezentáló lépésekké, amiket aztán a tervkészítés során adott esetben dekomponálunk. Egy biztos, az absztrakt cselekvések a természetben semmiképpen se adottak apriori úgy, mint ahogy a HD-RRT esetében. Módszerem ebben is rugalmasabb, hiszen kezdetben elemi cselekvésekkel dolgozik. Később alkalmasint az egyes géneket terminálisnak tekintve, úgymond absztrakt cselekvéseket hozhat létre, melyeket nyomban fel is használhat a tervkészítés során. Továbbá a STRIPS alapú tervkészítés alapvető hátránya, hogy rögzíteni kell a terv több specifikus aspektusát: az okozati kapcsolatokat, a sorrendi kényszereket, stb. Módszerem ezeket implicite, magától alakítja ki és felügyeli közvetve betartásuk. Visszatérve a HD-RRT-hez, amely a legtöbb ma működő tervkészítő eljárás alapja, azt kell, hogy mondjuk, hogy a felfelé és lefelé megoldhatóság (lásd. 2.2.2.3) heurisztikája nélkül az algoritmus sebessége (és ráadásul kifejezőereje) azonos az RRT-ével. Mindezen túl, azt is láthattuk, hogy a felfelé és lefelé megoldhatóság feltételei csak speciális esetben teljesülnek, általánosan nem. Ezzel szemben az én

módszerem egy általános, robusztus és „optimális” tervkészítési eljárást ad a két-szintű, egymásba ágyazott evolúciók elvének köszönhetően.

6.2.2. Mikro-világ tesztek

A rendszer tesztelésére két mikro-világot használtam fel. Ezek – az irodalomban ismert nevükön – a Wumpus (Russell és Norvig, 1995) és a Tábla-világ (Zhang és Cho, 1999).

A tábla-világ tesztelése – a wunpus-világgal ellentétben – még az implementáció egy korai időszakában történt, így még nem tartozott grafikus felület a rendszerhez. Ezért a tábla-világ megjelenítését majd egy manuálisan létrehozott ábrán szemléltetem. A wumpus-világ esetében már egy kifejlett grafikus interfész állt a rendelkezésemre, így ennek előnyeit a wumpus-világ tesztjeinek leírásánál a beiktatott „screenshot”-ok láttatják majd.

6.2.2.1. Asztal-világ

Az asztal világ (Zhang és Cho, 1999) egy $k \times n$-es négyzetrács alakú világ, ahol egy asztal, és egy cél helyezkedik el. Az ágens mozgását egy-egy négyzetrácsnyi helyet elfoglaló akadályok nehezítik. A pályán négy ágens található. Az ágensek célja az, hogy a – csupán együttes erővel felemelhető – asztalt átszállítsák a célpozícióba. Ennek megvalósítása kooperációt igényel az ágensek részéről. Ezért a következő módon konfiguráljuk őket. Amikor egy-egy ágens létrehoz egy-egy benső reprezentációt, akkor annak minden fantáziájában négy-négy ágenst helyez el, rendre a saját, és a többi ágens alteregóit (tehát az ágensek tudásának része, hogy rajtuk kívül még *K-1* másik ágens is van a pályán). Egy adott ágens ekkor úgy érlel egy-egy tervet, hogy a fantáziákon belül minden egyes alteregó által „végrehajtatja”, akár többször is, s az ágensek az adott fantázián végzett együttes teljesítménye alapján ítéli meg az adott tervet. Hangsúlyozom, hogy az ágenseken belül, egy-egy fantáziában ekkor homogén kooperációra tett törekvés történik, hiszen minden egyes fantáziabeli alteregó ugyanazon elven cselekszik, ugyanazzal a cselekvéssel (tervvel) éri el adott esetben a környezet által reprezentált feladat megoldásához szükséges kooperációt. Lássuk, hogy milyen módosító-jellegű cselekvések (lásd. 5.3.2) állnak az ágensek (és alteregóik) rendelkezésére:

- **Egyet lép előre**
- **Ellép**: Az ágens a környező nyolc négyzet közül arra lép, amerre az óramutató járásával azonos irányban tett próbálkozásai során elsőként nem ütközik akadályba.
- **Találomra ellép**: Az ágens egy találomra választott irányban lép el helyéről
- **Asztal felé fordul**
- **Cél felé fordul**
- **Helyben marad**

Emellett a következő vizsgáló cselekvések adottak az ágensek számára:

- ***Ha*** ütköztem
- ***Ha*** más ágensnek ütköztem
- ***Ha*** az asztal valamely szomszédos négyzeten van
- ***Ha*** a cél valamely szomszédos négyzeten van
- ***És***: Két tetszőleges cselekvés ÉS-kapcsolata
- ***ÉsÉs***: Három tetszőleges cselekvés ÉS-kapcsolata

A környezet leírásának szabályszerűségei alapján látható, hogy a tábla-világ, bár nem egyállapotú probléma, hiszen egy-egy cselekvésnek több lehetséges kimenetele is lehet, nem is igen eshetőségi, mert a cselekvések mindig kiszámítható, felsorolható következményekkel járnak. Így hát a táblavilág egy többállapotú problémát reprezentál (lásd. 2.2.1).

Nagyon fontos megemlítenem, hogy a tesztelés komplexitásának letörésére mind a wumpus, mind pedig itt a tábla-világ esetén az ágenseket nem egy-egy nem-felügyelt evolúció során állítottam elő, hanem magam láttam el őket egy-egy kezdeti genotípussal. Továbbá, azon túl, hogy a genotípus az előző fejezetekhez híven most is – a modell-tanulás implementációjának hiányában – a környezet teljes és helyes leképzését, azaz az adott ágens benső reprezentációját tökéletesen előállító interpretációs mechanizmust kódolt (lásd. 5.3.1), még ráadásul az ágens genotípusában kódolt „paraméterező mechanizmusokat" is kiiktattam. Így egy-egy ágens a benső komponenseit (pl. fitness függvényét, benső evolúcióját, fantáziáinak előállítását, stb.) nem maga paraméterezte minden egyes lépett lépés után újra, és újra, hanem élet-ciklusa során ezek változatlanok maradtak – én definiáltam őket, azaz én paramétereztem az egyes ágensek „benső működését", pl. „gondolkodását", stb. Ennek a háromszoros egyszerűsítésnek köszönhetően már kezelhető tartományba esett vissza a teszteredmények száma és változatossága.

Visszatérve, az egyes ágensek egyes fantáziáin belül elhelyezett alteregókat két gén hosszúságú kromoszómák reprezentálták, melynek egy-egy génjét az adott ágens más-más jósági mércével mért, magyarán: az ágensek összetett jósági mércét használtak a „kigondolt" tervet megitélésére. Ez a mérce a következő volt:

$$F = \frac{1}{f_1 + f_2}, \text{ ahol}$$

$$f_1 = \sum_{a=1}^{4} \{c_1 \cdot \max(X_a, Y_a) + c_2 \cdot S_a + c_3 \cdot C_a - c_4 \cdot M_a + K\}$$

$$f_2 = \sum_{a=1}^{4} \{c_1 \cdot \max(X_a, Y_a) + c_2 \cdot S_a + c_3 \cdot C_a - c_4 \cdot M_a + c_{5.} \cdot A_a + K\}, \text{ és}$$

X_a az *a* alteregó X-tengely mentén vett távolsága a céltól (f_1 -nél asztaltól, f_2 -nél céltól)

Y_a az *a* alteregó Y-tengely mentén vett távolsága a céltól (f_1 -nél asztaltól, f_2 -nél céltól)

S_a az *a* alteregó által megtett lépések száma

C_a az *a* alteregó összes ütközéseinek a száma

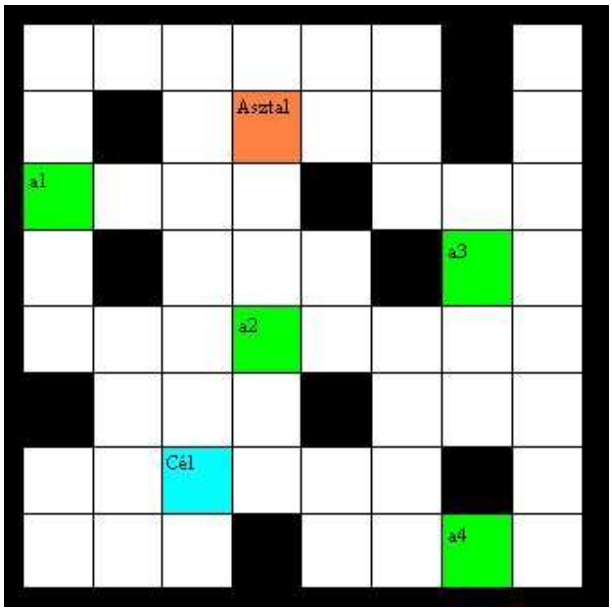
M_a az *a* alteregó kezdő és végállapota közti távolság

A_a az *a* alteregó bírsága azért, ha eltávolodott a többi ágens alteregóitól

c_i *i*. súlyozó-tényező

K pedig egy pozitív konstans normalizáló érték

A tábla-világ globális valóságára (vagy akár egy-egy ágens valamely fantáziájára) lehet példa a 6.1. ábrán mutatott „tábla-állás".



6.2. ábra: A tábla-világ egy lehetséges globális valósága

Általában $K = 40; c_1 = 0{,}25;; c_2 = 1{,}0; c_3 = 3{,}0; c_4 = 0{,}25$ értékek bizonyultak a leghatékonyabbnak az egyes ágensek egyes fantáziáin belüli, alteregók közti homogén kooperáció létrejöttéhez. Egy-egy ágensnek elég volt már 50 generáción át „gondolkodnia", azaz futtatnia benső evolúcióját ahhoz, hogy az alteregók teljesen homogén kooperációját előidéző tervet meglelje. **Fontos** megemlítenem, hogy, mint ahogyan fentebb már utaltam is a lehetőségre, az egyes terveket rövidségük folytán (2 gén = 2 lépés) ciklusosan hajtottak végre az alteregók. A tervek így valójában nem csupán két gén-hosszú kromoszómák által kódolt két-lépéses tervek voltak, hanem 2-periódusú, *N***2* hosszúságú, ciklikus tervek (lásd. 3.2. alfejezet, 3.2. táblázat, *feltételes ismétlődő tervek*). Így a ciklikusan végrehajtott 2-hosszú terveket az ágens jósági mércéje valójában a ciklikus működés általi viselkedés, azaz a fantáziákon való teljes végrehajtásuk alapján ítélte meg. Mindez azonban csak nagyjából 100-150 fős kromoszóma, avagy gondolat-populáció esetén, cirka 300 fejhosszúságú génekkel (avagy gondolatokkal) volt megvalósítható.

Az alteregót teljesen homogén kooperációjának jelentése az, hogy az alteregók az asztal köré képesek gyűlni, majd pedig együttesen képesek azt átszállítani a cél-pozícióba. Látható, hogy a jósági függvény első komponense felelős azért, hogy az alteregók a fantáziákon belül az asztal köré

gyűljenek *(homing)*, míg a második komponens felelős azért, hogy az asztalt együttes erővel a célba juttatásák *(herding)*.

Az tesztelés legmeglepőbb, ámde mégis várva-várt tapasztalata az volt, hogy a fenti módon implementált ágensek, azáltal, hogy a fantáziáikban létrejött alteregóik közt a **homogén kooperáció**, képesek voltak egymás közt a **heterogén kooperáció** megvalósítására! Ezt tehát azt jelenti, hogy egyes (értelmes) élőlényekhez hasonlóan, különböző elvek mentén „kigondolt" különböző cselekvések által voltak képesek kooperálni.

6.2.2.2. Wumpus világ

Adott egy $k \times n$-es négyzetrács, amit falak vesznek körül. Ezt nevezzük a Wumpus-világ (Russell és Norvig, 1995) alapterületének. A rácspontokban különböző objektumok, élőlények és ágensek helyezkedhetnek el. Az objektumoknak két típusa van: *arany* és *árok*. Élőlény csak egy van: a *Wumpus* nevezetű szörnyeteg. Ágensekből tetszőleges számút elhelyezhetünk a pályán. **Fontos** megjegyeznem, hogy a következőkben eltérek a wumpus-világ Russell-féle szabályrendszerétől, mivel azt túl egyszerűnek találtam. Tovább folytatva, az ágenseket a pályán véletlenszerűen helyezzük el véletlenszerűen orientálva. Az ágensek számára a következő módosító-cselekvések állnak rendelkezésre:

- ***Mászik***: Ha az ágens a pálya valamely sarkában van, akkor egy létrán kimászhat a játéktérből.
- ***Balra***, vagy ***Jobbra fordul***
- Egyet ***lép előre***
- ***Lő***: Az ágensnek kezdetben a pályán található Wumpus-okkal arányos számú lövése van. Egy-egy lövés akkor talál, ha abban az irányban, amerre lőtték, legalább egy Wumpus van, egyébként a lövedék falba ütközik, és nesztelen a semmibe vész.
- ***Felvesz***: Ha az ágens épp egy olyan négyzetre ér, amely aranyat tartalmaz, akkor lehetősége van arra, hogy felvegye. A többi cselekvéshez hasonlóan ez is, ha a „sikeres" kimenetel előfeltételei nem állnak fenn a környezetben, hatástalan marad.
- ***Vár***: Ebben az esetben az ágens egy lépés erejéig semmit se tesz, hacsak nem ezt…

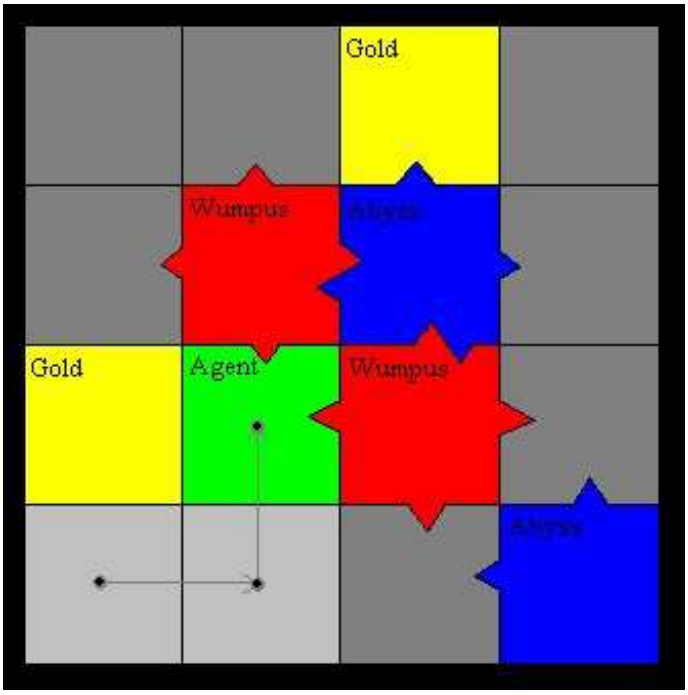
Az ágensek lokális valósága öt érzetből tevődik össze: BŰZ, SZELLŐ, RAGYOGÁS, ÜTKÖZÉS és SIKOLY. A wumpus-világ globális valóságának vannak bizonyos, igen jól definiált „rejtett szabályszerűségei", melyek azt diktálják, hogy egy-egy Wumpus körül (a közvetlen szomszédos négy négyzetben), illetve az őt tartalmazó négyzeten BŰZ legyen érezhető. Az árok körül hasonló módon érezhető SZELLŐ. Az aranyat tartalmazó négyzeten RAGYOGÁS–t észlelhet az arra tévedő ágens, aki, ha a pálya valamely falának ütközik, akkor ÜTKÖZÉS–t érez. Ha valaki lelőtt egy Wumpus-t, akkor a pálya minden pontján egy pillanatra SIKOLY hallható.

Megvalósításomban az ágensek CLP-alapú (Szeredi, 2002) következtető mechanizmusuknak köszönhetően képesek a Wumpus-ok, árkok betájolására (lásd. 5.3.2). A további logika lehetővé tételéért az egyes ágensek kognitív és észlelő, egyszóval vizsgáló cselekvései a felelősek. Ezek a következők:

- ***Ha*** bűzt érzek
- ***Ha*** szellőt érzek
- ***Ha*** ragyogást látok
- ***Ha*** ütközést érzek
- ***Ha*** sikolyt hallok
- ***Ha*** előttem Wumpus van
- ***Ha*** előttem árok van

Mivel az ágens alapjában véve csak az épp aktuális négyzeten érezhető, tapasztalható lokális valósággal szembesül, ezért kénytelen igénybe venni a memorizált tapasztalatait ahhoz, hogy kikövetkeztethesse, hogy hol van, és hol nincs Wumpus, illetve árok, stb. Amíg tehát az első öt említett vizsgáló cselekvés nem végez logikai következtetést, pusztán csak egy-egy lokális valóságbeli érzet meglétét firtatja, addig az utolsó két említett vizsgáló cselekvés működése – megfelelő megvalósítás esetén – logikai következtetésen alapul. Ahhoz, hogy az ágens érdemben feltehesse a *„Van-e előttem Wumpus?”* kérdést, bizonyos fokú tapasztalatot kell szereznie a környezetről, úgymond feltérképezve a lokális valóságot (lásd. 5.5. ábra).

Az ágensek jósági mércéje a tesztek során a felvett aranyak, elpusztított Wumpus-ok és sikeres menekülési kísérletek arányában értékelte az alteregókat. Egy lehetséges wumpus-világ globális valóságára mutat példát a 6.2. ábra.

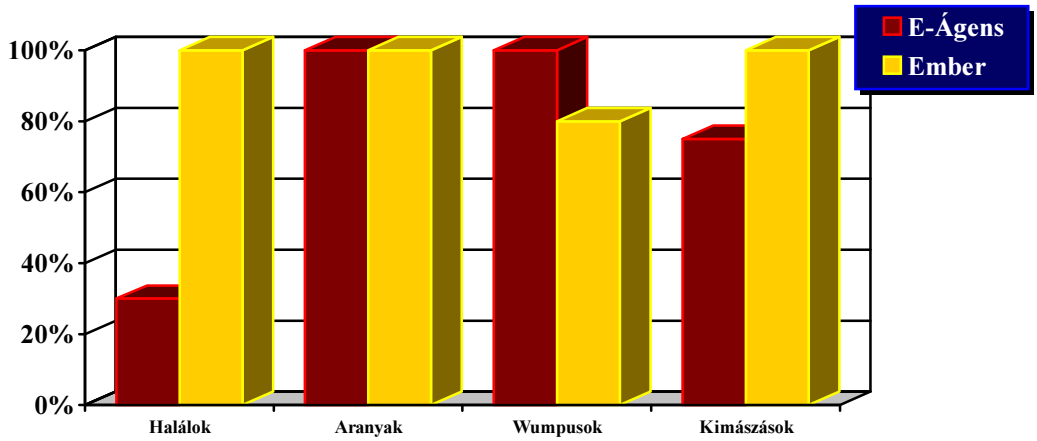


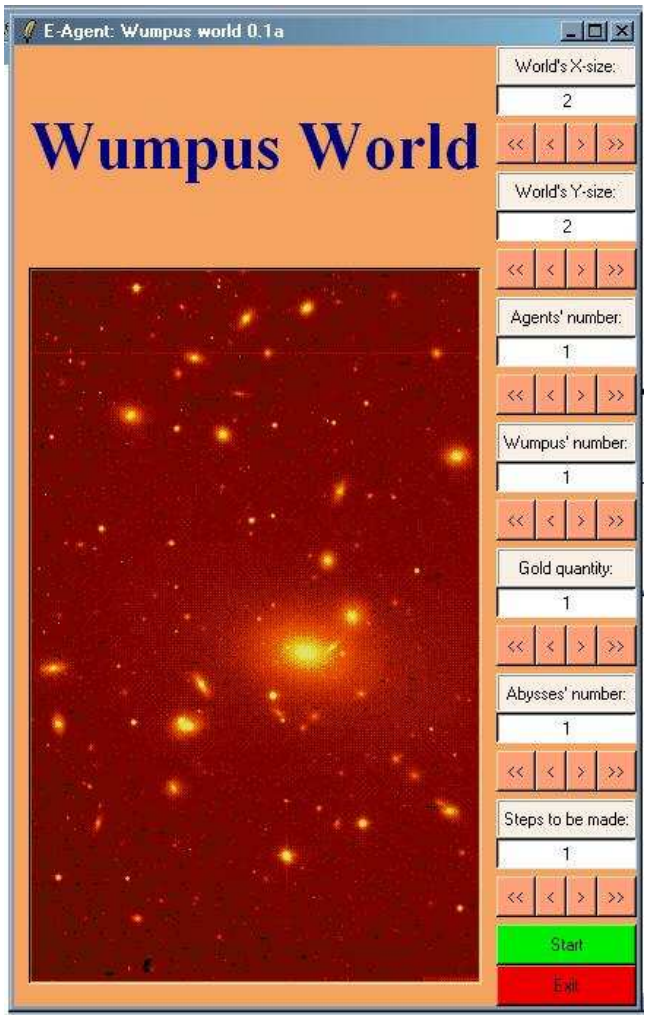
6.2. ábra: Egy lehetséges wumpus-világ

Talán a **legfontosabbat** a végére hagytuk: az ágensek pusztán csak azt a négyzetet érzékelik, amelyikben éppen elhelyezkednek. Magyarán lokális valóságuk mindig a pálya egy-egy négyzete, s így hozzátartozó öt-öt érzete. A lokális és a globális valóság komplexitásának ezen eltérése eredményezi, hogy a wumpus-világ már a tábla-világnál bonyolultabb, életszerűbb, nehezebben megoldható, **eshetőségi** problémát reprezentál. Eshetőségi, ugyanis léteznek olyan állapotok, melyekben – az ágens „szemével nézve" a pályát – nem garantálható egy-egy cselekvés kimenetele. Léteznek olyan helyzetek, amikor a szerencsén múlik az, hogy éppenséggel mi fog történni egy-egy adott cselekvés végrehajtásának a hatására. Egyszerű példája ennek az lehet, hogy, ha az épp tapasztalatlanul a világba „pottyanó" ágens szelet, vagy bűzt érez. Az árok, vagy wumpus bárhol (sőt, többes esetben akár mindenhol) lehet körülötte. Az ágens egyszerűen nem tud jó döntést hozni arra vonatkozólag, hogy *melyik irányba induljon.* Ebben az esetben tehát a világ ágens számára való hozzáférhetetlensége folytán előálló nem-determinizmus okozza az eshetőség „látszatát". Látszat azért, mert valójában (mi tudjuk, hogy) a világ determinált, ám az ágens ezt sajnos nem érzékeli. Számára, ha SZELLŐ–t érez a kezdőállapotban, akkor a szerencsén múlik az, hogy túléli-e az első lépést.

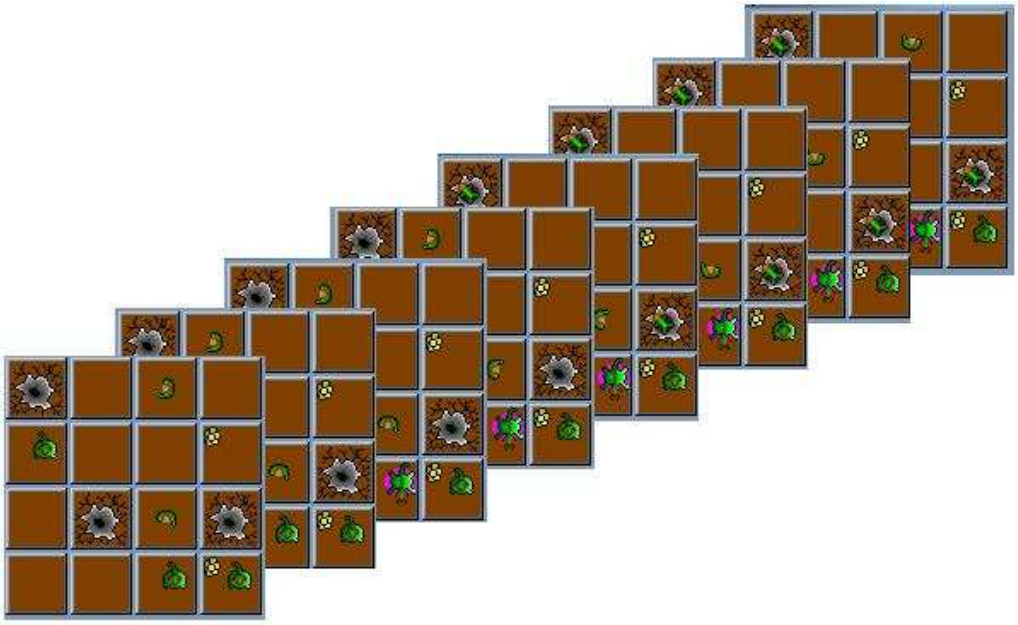
Tehát a wumpus-világ egy alkalmas környezet lehet arra, hogy leteszteljük módszerünk eshetőségi probléma-megoldó készségét. A teszteknek ez esetben két fajtáját végeztem. Egyrészt – a tábla-világnál is említett módon paraméterezve az ágenseket – ágenspopulációkat helyeztem egy-egy wumpus-világba, és azt vizsgáltam, hogy a környezet által reprezentált problémát milyen hatékonysággal oldják meg. Milyen gyakran maradnak életben, és milyen gyakran pusztúlnak el? A meglévő aranyakból hányat képesek összegyűjteni? A wumpus-ok számát milyen mértékig képesek „redukálni"? A függelék 10.2. táblázata felsorol egynéhány ágens-környezet paraméter kombinációt. A paramétereket a teszt-esetek definiálásakor szöveg-fájlban, vagy akár grafikus felületen keresztül is megadhattuk. A grafikus felület e részét mutatja a 6.3. ábra.

A felparaméterezett és előállított wumpus-világon az ágensek rendre egymás után, amolyan „round-robin"-módra (Tanenbaum, 1999) következnek. Érzékelnek, gondolkodnak és lépnek, mint ahogyan azt az 5.2.1. szakasz absztrakt architektúrája megkívánja. A wumpus-világok egy (lásd. 10.1.táblázat, T12) lehetséges alakulását mutatja a 6.4. ábra. Az ember-ágens tesztek végső összegzését pedig a következő diagram mutatja. Ezen az látható, hogy a problémákon vett átlagok problémákra vett átlaga szerint mi az ember és az ágens egymáshoz viszonyított teljesítménye:





6.3.ábra. Implementáció kezelői felülete



6.4.ábra. A wumpus-környezetek alakulása

6.3 ÉRTÉKELÉS ÉS KITEKINTÉS

Összefoglalva megállapíthatjuk, hogy elképzelésem (Kovács, 2003) mindkét esetben helytállónak bizonyult, az algoritmus működőképes, a vizsgált feladatokat megoldotta. Ez azt jelenti, hogy képes volt eshetőségi problémák kezelésére, avagy eshetőségi tervkészítésre. A módszer adaptív, öntanuló – s az evolúciónak köszönhetően – robusztus heurisztika.

Láttuk, hogy az emberekkel való összevetések problémákra vett átlaga szinte minden tekintetben vetekszik az emberi teljesítménnyel. Az egyetlen aspektus, amiben az emberi játékosok jobbnak bizonyultak az ágenseknél, a kimászások száma volt. Ellenben az ágensek átlagban kevesebbszer haltak meg, s azonos aranymennyiség begyűjtése mellett valamivel több Wumpus kiiktatására voltak képesek. Magyarán, a rendszer már egy ilyen egyszerű, csekély méretű eshetőségi probléma esetén is képes volt felülmúlni az emberi következtetés teljesítményét.

Végeredményben, túl a teszteken, az elméletet is figyelembe véve, azt mondhatjuk, hogy egy olyan, kellően általános elvhez jutottunk, amely a vizsgált eseteknél nagyságrendekkel összetettebb problémákra is felskálázható. Az elv általános. A jövőbeni megvalósítás a felderíthetőségi problémák elvi megoldhatóságának gyakorlatba való átültetését fogja megcélozni, lehetővé téve a garantáltan „optimális” tervkészítést valós komplexitású környezetekben, új utakat nyitva az intelligens viselkedés irányában tett törekvések számára.

7. Összefoglalás

A hálózatok, az infrastruktúra, a gazdaság, és szinte minden egyéb, az emberi társadalmat kiszolgáló rendszer összetettségének növekedtével az intelligens rendszerek jelentősége is egyre nagyobbá válik, hiszen előbb-utóbb egyedül már nem leszünk képesek arra, hogy átlássuk a minket körülvevő és támogató rendszerek működését. Manapság is léteznek szakértő-rendszerek, melyek támogatják a különféle tervezést, terv-készítést, ám egyik se képes a terv-készítés betáplált sémái nélkül dolgozni.

Dolgozatomban egy olyan új elvet mutattam be, amely megoldást jelent az automatikus terv-készítés problémájára: elviekben felderíthetőségi problémák, azaz dinamikus, hozzáférhetetlen és nem-determinisztikus környezetekben történő „optimális" tervkészítés megvalósítására ad lehetőséget úgy, hogy se a környezet, se a cselekvések, se egyéb tényezők modelljét nem kell megadnunk. A jelenlegi megvalósítás eshetőségi problémák feladat-specifikus megvalósítását teszi lehetővé.

A következőkben ennek a kiterjesztésére teszek majd próbálkozást, függetlenítve az implementációt környezeti reprezentációktól. Miután az implementációnak a környezet csak egy „input"-ja lesz, a következő lépés az elv teljes implementációja: *egy „optimális" felderíthetőségi tervkészítő megvalósítása.*

8. Köszönetnyílvánítás

Ezennel szeretnék köszönetet mondani mindazoknak, akik nélkül ez a dolgozat nem jött volna létre. Elsősorban konzulensemnek, Dr. Dobrowiecki Tadeusz Tanár úrnak szeretném megköszönni, hogy támogatott, és hitt bennem. Köszönöm Szüleimnek, Barátaimnak és Tanáraimnak.

9. Irodalomjegyzék

Bacchus, F., Ady, M. (2001). *Planning with Resources and Concurrency: A Forward Chaining Approach*, In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2001), pp. 417-424

Baral, C., Kreinovich, V., Trejo, R. (1999). *Computational complexity of planning and approximate planning in presence of incompleteness.* In The International Joint Conferences on Artificial Intelligence (IJCAI).

Barnsley, M.F. (1988). *Fractals everywhere*. Academic Press.

Blum, A., Furst, M.L. (1995). *Fast planning through planning graph analysis*. Proc. IJCAI-95, Montreal, Canada.

Bratman, M. (1987). *Intention, Plans, and Practical Reason* Cambridge, Mass., Harvard University Press.

Brooks, R.A. (1991). *How to build complete creatures rather than isolated cognitive simulators*, In K. VanLehn (ed.), Architectures for Intelligence, pp. 225-239, Lawrence Erlbaum Assosiates, Hillsdale, NJ.

Bylander, T. (1992). *Complexity results for serial decomposability*. In Proceedings of the Tenth National Conference of Artificial Intelligence (AAAI92), pp. 729-734. San Jose, California. AAAI Press.

Canny, J. (1985). *Unpublished Observation*.

Chapman, D. (1987). *Planning for conjunctive goals*. Artificial Intelligence, 32, pp. 333-377

Cramer, N.L. (1985). *A representation for the adaptive generation of simple sequential programs*. In Proceedings of an International Conference on Genetic Algorithms and their Applications, J.J.Grefenstette (ed.), Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 183-187.

Currie, K.W., Tate, A. (1991). *O-Plan: the Open Planning Architecture*. Artificial Intelligence, 52 (1), pp. 49-86.

Darwin, C. (1859). *On the origin of species by means of natural selection*. Murray.

d'Inverno, M., Kinny, D., Luck, M. és Wooldridge, M. (1997). *A formal specification of dMARS*, In M. P. Singh, A. Rao, and M. J. Wooldridge, editors, Intelligent Agents IV (LNAI Volume 1365), pages 155-176. Springer-Verlag: Berlin, Germany.

Draper, D., Hanks, S., Weld, D.S. (1994). *Probabilistic Planning with Information Gathering and Contingent Execution*. In Kristian J. Hammond (ed.) Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94), pp. 31-36. Morgan Kaufmann.

Edelkamp, S., Helmer, M. (2000). *On the implementation of MIPS*. Paper presented at the Fifth International Conference on Artificial Intelligence Planning and Scheduling Workshop on Model-Theoretic Approaches to Planning, Breckenridge, Colorado.

Erol, K., Nau, D.S., Handler, J. (1994). *UMCP: A Sound and Complete Planning Procedure for Hierarchical Task-Network Planning*, In AIPS-94, Chicago.

Erol, K., Nau, D.S., Subrahmanian, V.S. (1995). *Complexity, decidability and undecidability results for domain-independent planning*, Artificial Intelligence, Vol. 76, pp. 75-88.

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., Williamson, M. (1992). *An approach to planning with incomplete information*. In Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning.

Ferguson, I. A. (1992). *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*, Ph.D. Thesis, Clare Hall, University of Cambridge, UK.

Ferreira, C. (2001). *Gene Expression Programming: A New Adaptive Algorithm for Solving Problems*, Complex Systems, Vol.13, issue 2, pp. 87-129

Fikes, R.E., Nilsson, N.J. (1971). *STRIPS: a new approach to the application of theorem proving to problem solving*, Artificial Intelligence, 2 (3-4), pp. 189-208

Fogel, D.B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, Piscataway, NJ: IEEE Press.

Fogel, L.J., Owens, A.J., Walsh, M.J. (1966). *Artificial Intelligence through Simulated Evolution*, New York: John Wiley.

Fujiko, C., Dickson, J. (1987). *Using the genetic algorithm to generate LISP source code to solve the prisoner's dilemma*. In Proceedings of the Second International Conference on Genetic Algorithms, J.J.Grefenstette (ed.), Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 236-240

Gelernter, H. (1959). *Realization of geometry proving machine*. In Proceedings of an International Conference of Information Processing, pp. 273-282 Paris. UNESCO House.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley

Green, C. (1969). *Theorem-proving by resolution as a basis for question-answering systems*. In Meltzer, B., Michie, D., and Swann, M. (eds.), Machine Intelligence 4, pp. 183-205. Edinburgh University Press, Edinburgh, Scotland.

Heidegger, M. (1954). *Aus der Erfahrung des Denkens*. Pfullingen.

Hoffmann, J., Nebel, B. (2001). *The FF Planning System: Fast Plan Generation Through Heuristic Search*, In Journal of Artificial Intelligence Research, Vol. 14, pp. 253-302

Hofstadter, D.R. (1979). *Gödel, Escher, Bach: An eternal golden braid*. Basic Books.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology*, Control, and Artificial Intelligence. MIT Press.

Jordan, M., Jacobs, R. (1994). *Hierarchical mixtures of experts and the em algorithm*. Neural Computation, 6, pp. 181-214

Kautz, H., Selman, B. (1996). *Pushing the Envelope: Planning, Propositional Logic and Stochastic Search*, In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), pp. 1194-1201. MIT Press.

Kautz, H., Selman, B. (1998). *BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving*. Workshop Planning as Combinatorial Search, AIPS-98, Pittsburgh, PA.

Kovács, D.L. (2003). *Evolution of Intelligent Agents: A new approach to automatic plan design,* Proceedings of IFAC Workshop on Control Applications of Optimization, Elsevier. *(being published)*

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Littman, M.L., Goldsmith, J., Mundhenk, M. (1998). *The computational complexity of probabilistic planning*. Journal of Artificial Intelligence Research, volume 9, pp. 1-36

McAllester, D., Rosenblitt, D. (1991). *Systematic nonlinear planning*. In Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), Vol. 2, pp. 634-639. Anaheim, California. AAAI Press.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M.; Weld, D., Wilkins, D. (1998). *PDDL---The Planning Domain Definition Language*. AIPS-98 Planning Committee.

Muller, J.P., Pischel, M. (1993). *The agent architecture inteRRaP: Concept and application*, Research Report RR- 93-26, Deutsches Forschungszentrum fur Kunstliche Intelligenz, Kaiserslautern, Germany.

Nau, D.S., Cao, Y., Lotem, A., Munoz-Avilla, H. (1999). *SHOP: Simple Hierarchical Ordered Planner*. In Proceedings of IJCAI-99.

Nau, D.S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, W., Wu, D., Yaman, F. (2003). *SHOP2: An HTN Planning System*. Journal of Artificial Intelligence Research. *(being published)*

Newell, A., Simon, H.A. (1956). *The logic theory machine*. IRE Transactions on Information Theory, IT-2(3), pp. 61-79

Newell, A., Simon, H.A. (1961). *GPS, a program that simulates human thought*. In Billing H. (ed.), Lernende Automaten, pp. 109-124. R. Oldenbourg, Munich, Germany. Reprinted in Feigenbaum and Feldman (1963).

Papadimitriou, C.H. (1994). *Computational Complexity*. Addison-Wesley, Reading, MA.

Penberthy, J.S., Weld, D.S. (1992). *UCPOP: A sound, complete, partial order planner for ADL*. In Proceedings of KR-92, pp. 103-114

Peot, M., Smith, D. (1992). *Conditional nonlinear planning*. In Hendler J. (ed.), Proceedings of the First International Conference on AI Planning Systems, pp. 189-197. College Park, Maryland. Morgan Kaufmann.

Robinson, J.A. (1965). *A machine-oriented logic based on the resolution principle*. Journal of the Association for Computing Machinery, 12, 23-41.

Rónyai, L., Ivanyos, G., Szabó, R. (1998). *Algoritmusok*. Typotex, Budapest.

Russell, S.J., Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Sacerdoti, E.D. (1974). *Planning in a hierarchy of abstraction spaces*. Artificial Intelligence, 5 (2), 1, pp. 15-135

Sacerdoti, E.D. (1975). *The nonlinear nature of plans*. In Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI75), pp. 206-214. Tbilisi, Georgia. IJCAII.

Steels, L. (1990). *Cooperation between distributed agents through self organization*. In Y. Demazeau and J.-P. Müller, editors, Decentralized AI -- Proceedings of the First European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-89), pp. 175-196. Elsevier/North Holland.

Stefik, M.J. (1981). *Planning and meta-planning.* Artificial Intelligence, 16, pp. 141-169.
Schwefel, H.P. (1995). *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. John Wiley & Sons, Inc., New York.

Sussman, G.J. (1973). *A computational model of skill acquisition*. Massachusetts Institute of Technology, Technical Report No. AI TR-297.

Szeredi P., Benkő T. (2000). *Bevezetés a logikai programozásba*. BME.

Szeredi P. (2002). *Nagyhatékonyságú logikai programozás*. Előadásjegyzet.

Tanenbaum, S. A. (1999). *Számítógép-Hálózatok*. Panem-Prentice Hall.

Tate, A. (1975). *Interacting goals and their use*. In Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75), pp. 215-218, Tbilisi, Georgia. IJCAII.

Tate, A. (1977). *Generating project networks*. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI77), pp. 888-893. Cambridge, Massachusetts. IJCAII.

Van Hentenryck, P. (1989). *Constraint Satisfaction in Logic Programming*, Logic Programming Series. MIT Press.

Warren, D.H.D. (1974). *WARPLAN: a system for generating plans*. Department of computational logic memo 76, University of Edinburgh, Edinburgh, Scotland.

Warren, D.H.D. (1976). *Generating conditional plans and programs*. In Proceedings of the AISB Summer Conference, pp. 344-354

Wilkins, D.E. (1988). *Practical Planning: Extending the AI Planning Paradigm*. Morgan Kaufmann, San Mateo, California.

Wooldridge, M., Jennings, N.R. (1995). *Intelligent Agents: Theory and Practice.* The Knowledge Engineering Review, 10, pp. 115-152

Wooldridge, M. (1999). *Intelligent Agents*, In G.Weiss (ed) Multiagent Systems. MIT Press.

Younes, H.L.S., Simmons, R.G. (2002). *On the role of ground actions in refinement planning*. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso (eds.), Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling Systems, pp. 54-61, Toulouse, France. AAAI Press.

Zhang, B-T., Cho, D-Y. (1999). *Co-evolutionary Fitness Switching: Learning Complex Collective Behaviours Using Genetic Programming*, Advances in Genetic Programming, Vol. 3, 18, pp. 425-445

10. Függelék

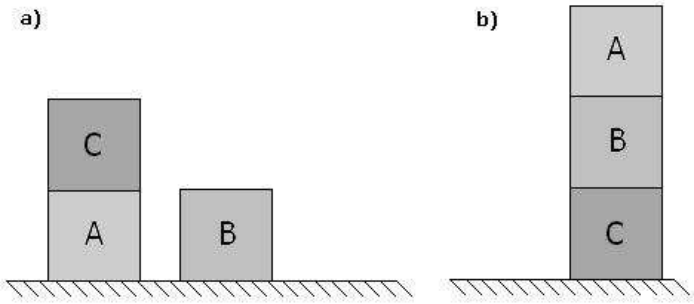
10.1. TERVKÉSZÍTÉSI MÓDSZEREK FELSOROLÁSA

TÍPUS / MÓDSZER	Hierarchikus		Nem hierarchikus		Feltételes tervkészítő	Kiterjesztett operátorok	Logika	
	Teljesen rendezett	Részben rendezett	Teljesen rendezett	Részben rendezett			0-rendű	1-rendű
LT							■	
GTPM							■	
GPS							■	
QA3							■	
STRIPS			■				■	
ABSTRIPS	■						■	
SIPE				■				
HACKER				■				
WARPLAN				■				
WARPLAN-C				■	■			
INTERPLAN				■				
TWEAK				■				
NOAH		■						
NONLIN		■						
MOLGEN		■						
SNLP		■						■
O-PLAN		■				■		■
UCPOP		■				■		■
UWL		■			■			■
CNLP		■			■			■
C-BURDIAN		■			■	■		■
UMCP		■						■
GRAPHPLAN		■					■	
SATPLAN		■					■	
BLACKBOX		■					■	
SHOP	■							■
MIPS			■		■	■		■
TLPLAN			■			■		■
FF	■				■	■		■
VHPOP		■			■	■		■
SHOP2		■						■

10.1.táblázat. Tervkészítési módszerek osztályozása

10.2. SUSSMAN ANOMÁLIA

A probléma a következő: adott egy asztal, és rajta kockák. Célunk az, hogy a 10.1. (a) ábrán látható kiindulási állapotból a 10.1. (b) ábrán látható cél-állapotot állítsuk elő pusztán csak a kockák rakosgatásával. Egyszerre csak egy kockát mozgathatunk. Az elmozdított kockán más kocka nem helyezkedhet el, továbbá a mozgatott kockát – az asztalon kívül – csak olyan kockára rakhatjuk rá, amelyiken más kocka nincsen.



10.1.ábra. Sussman anomália

A problémára egy időben nem tudtak megoldást adni az akkori tervkészítő módszerek, mivel a rész-célok (A van B-n, B pedig C-n) összefüggnek egymással.

10.3. ÁGENS-ÁGENS TESZTEK A WUMPUS-VILÁGON

Paraméter \ Teszt sorszáma	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
Runs	3	3	3	3	3	3	3	3	3	3	3	3	3	3
WorldSizeX	4	4	4	4	4	4	4	4	4	4	4	4	4	4
WorldSizeY	4	4	4	4	4	4	4	4	4	4	4	4	4	4
AgentNum	1	2	3	1	3	3	3	3	3	3	3	3	3	3
WumpusNum	1	1	1	3	3	3	3	3	3	3	3	3	3	3
GoldNum	1	1	1	3	3	3	3	3	3	3	3	3	3	3
AbyssNum	1	1	1	3	3	3	3	3	3	3	3	3	3	3
Steps	50	50	50	50	50	30	30	30	30	30	30	30	30	30
Fantanum	7	7	7	7	7	15	15	15	15	15	15	15	15	15
ActionL	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Popnum	30	30	30	30	30	30	50	50	50	50	50	50	50	50
Popsize	31	31	31	31	31	31	31	51	51	51	51	51	51	51
GeneNum	3	3	3	3	3	3	3	3	7	7	7	7	7	7
HeadLength	63	63	63	63	63	63	63	63	63	127	127	127	127	127
P1cx	850	850	850	850	850	850	850	850	850	850	850	850	850	850
P2cx	200	200	200	200	200	200	200	200	200	200	200	200	200	200
Pgtx	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Pisx	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Prisx	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Pgcx	200	200	200	200	200	200	200	200	200	200	200	200	200	200
Pmutx	10	10	10	10	10	10	10	10	10	10	100	100	100	100
Nevals	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FGWPGoldsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FGWPResx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	10000	10000	10000
FMEMPMovesx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPLeftsx	800	800	800	800	800	800	800	800	800	800	800	800	800	800
FMEMPRightsx	800	800	800	800	800	800	800	800	800	800	800	800	800	800
FMEMPShootingsx	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FMEMPPicksx	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FMEMPClimbsx	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FMEMPStopsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPSmellsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPWindsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPShinesx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPPushesx	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
FMEMPScreamsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FPFGWx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	100
FPFMEMx	100	100	100	100	100	100	100	100	100	100	100	100	1000	1000

10.2. táblázat. Részlet egy ágens-ágens teszt paramétereinek alakulásából a Wumpus világban

A teszt-paraméterek magyarázata a 10.4. függelékben található.

10.4. EMBER-ÁGENS TESZTEK A WUMPUS-VILÁGON

Teszt sorszáma / Paraméter	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29
Runs	3	3	3	3	3	3	3	3	3	3
WorldSizeX	4	5	5	3	7	3	7	5	15	4
WorldSizeY	4	3	3	3	7	10	4	5	5	3
AgentNum	1	1	1	1	1	1	1	1	1	1
WumpusNum	3	2	1	4	4	10	5	3	13	3
GoldNum	3	1	3	4	4	7	3	1	1	1
AbyssNum	3	3	2	3	3	5	5	5	7	2
Steps	50	50	50	50	50	50	50	50	50	50
Fantanum	7	7	7	7	7	7	7	7	7	7
ActionL	1	1	1	1	1	1	1	1	1	1
Popnum	30	30	30	30	30	30	30	30	30	30
Popsize	31	31	31	31	31	31	31	31	31	31
GeneNum	3	3	3	3	3	3	3	3	3	3
HeadLength	63	63	63	63	63	63	63	63	63	63
P1cx	850	850	850	850	850	850	850	850	850	850
P2cx	200	200	200	200	200	200	200	200	200	200
Pgtx	100	100	100	100	100	100	100	100	100	100
Pisx	100	100	100	100	100	100	100	100	100	100
Prisx	10	10	10	10	10	10	10	10	10	10
Pgcx	200	200	200	200	200	200	200	200	200	200
Pmutx	10	10	10	10	10	10	10	10	10	10
Nevals	1	1	1	1	1	1	1	1	1	1
FGWPGoldsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FGWPResx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPMovesx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPLeftsx	800	800	800	800	800	800	800	800	800	800
FMEMPRightsx	800	800	800	800	800	800	800	800	800	800
FMEMPShootingsx	0	0	0	0	0	0	0	0	0	0
FMEMPPicksx	0	0	0	0	0	0	0	0	0	0
FMEMPClimbsx	0	0	0	0	0	0	0	0	0	0
FMEMPStopsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPSmellsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPWindsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPShinesx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FMEMPPushesx	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
FMEMPScreamsx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FPFGWx	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
FPFMEMx	100	100	100	100	100	100	100	100	100	100

10.3. táblázat. Részlet egy ember-ágens teszt paramétereinek alakulásából a Wumpus világban

38 paraméterrel rendelkezett egy-egy wumpus-világot megvalósító ágens-környezet a tesztek során. Egyrészt adott volt *Runs,* avagy, hogy hányszor futtassuk ne a rendszert egy-egy paraméter-kombinációra. A *WorldSize* paraméterek a wumpus-világ maximális méretét definiálták. Az *AgentNum* paraméter az ágensek, a *WumpusNum* a Wumpus-ok, a *GoldNum* az aranyak, míg az *AbyssNum* az árkok számát jelölte. Mindet véletlenszerűen helyeztük el a pályán. *Steps* definiálta, hogy egy-egy ágens legfeljebb hány lépést tehet meg a világban. *Fantanum* szabta meg az egyes ágensek által egy-egy gondolkodási ciklusban előállított fantáziák maximális számát. *ActionL* a végrehajtott tervlépések száma. *PopNum* és *PopSize* az ágensek benső evolúcióinak paraméterei, melyek a generációk számát, illetve az egyes generációk méretét határozzák meg. *GeneNum* az

ágensek által érlelt tervek, avagy kromoszómák hossza, azaz lépéseik, génjeik száma. *HeadLength* az egyes gének fejhosszát adták meg (lásd. 2.3.3), s így a gén által reprezentált program-fa mélységét is egyben. A *P*-betűs paraméterek a GEP algoritmus genetikus operátorainak valószínűségeinek 1000-szeresei. Az *F*-betűs paraméterek pedig az egyes ágensek jósági függvényének bemeneti paraméterezése, amely az egyes alteregók által előállított fantáziák különböző jellemzőinek figyelembevételi arányát súlyozza.

Az említett paramétereket természetesen a felhasználónak nem kell mind megadni, elég csupán az első 8 paraméter megadása, melyek valójában csak a globális környezetet definiálják. Minden mást az ágensek maguk hoznak létre adaptíve.