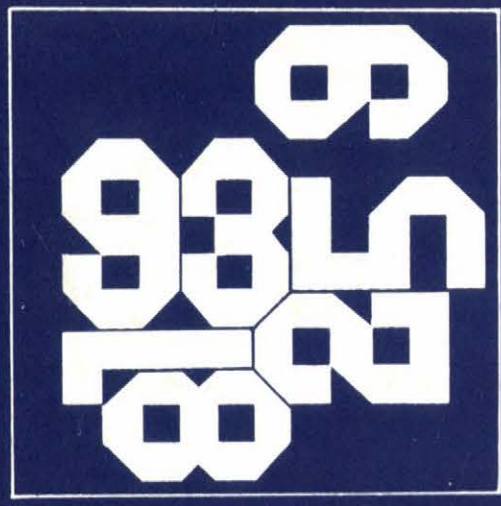
tanulmányok

197/1987

MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest



A TUDÁSÁBRÁZOLÁS TECHNIKÁI ÉS GÉPI ESZKÖZEI

Felmérő tanulmány

HERNÁDI ÁGNES

BODÓ ZOLTÁN

KNUTH ELŐD

Tanulmányok 197/1987

A kiadásért felelős:

KEVICZKY LÁSZLÓ

Készült: az OMFB-vel kötött 700325 (G1-16-038/86) sz. szerződésben kitűzött 4.1 számú feladat teljesítésekent.

ISBN 963 311 227 3

ISSN 0324-2951

Készült az Országos Széchényi Könyvtár
Sokszorosító üzemében, Budapest
a megrendelő által adott szövegeredeti alapján.
Felelős vezető: Rosta Lajosné
Példányszám: 220, terjedelem: 27 (A/5) kiadói ív.
Munkaszám: 87 076

TARTALOMJEGYZÉK

I. BEVEZETÉS

Ezen tanulmány célja annak a későbbi évekre előirányzott szoftver fejlesztési munkának az előkészitése, mely a megismerés folyamatát, a fogalomalkotás iterativ lépéseit, az igy kialakuló tudás rendszerezett karbantartását, dokumentálását, más személyek (vagy számitógéprendszerek) felé való átadását kivánja majd a legkorszerűbb eszközök alkalmazásával támogatni. Ez a rendszer túl kiván lépni azokon a határokon, melyeket a jelenleg rendelkezésre álló technikák nyújtanak. A továbblépés fő eszköze (jelenlegi elképzeléseink szerint) az absztrakciós mechanizmusok és mai alkalmazási módjuk felülbirálata, egyfelől új mechanizmusok bevezetésével, másfelől pedig az interaktiv-absztrakció biztositásával. Ez utóbbi a hagyományos megoldásokkal szemben merőben új eszközökkel valósitható csak meg. Mielőtt azonban ezen új megközelitéseket részleteiben megterveznénk, szükségesnek tartottuk a probléma jelenlegi nemzetközi állását gondosan felmérni. Ezt tartalmazza jelenlegi anyagunk.

Az anyag első részében a mesterséges intelligencia kutatások során eddig kialakult bevált megoldásokat tekintjük át. A második rész a kérdést az adatbázis kezelés oldaláról közeliti meg és az un. fogalmi leiró nyelvek alapjait tárgyalja.

II. TUDÁSÁBRÁZOLÁS

1. A tudásábrázolás helye és szerepe

A mesterséges intelligencia rendszerekben - a hagyományos adatbázis rendszerekkel ellentétben - elengedhetetlen a tárgyakra és folyamatokra, valamint a célokra, a motivációra, az okozati viszonyokra, az időre, a tevékenységekre stb. ismereteket tartalmazó tudásbázis. Tudásra alapozott számítógépes rendszerekhez viszont a hagyományosan a szakértők agyában ill. papíron tárolt tudás géppel kezelhető alakjára van szükségünk. Ehhez pedig nem elég, ha a dokumentumokat a gépek számára olvashatóvá tesszük, hiszen ettől a dokumentumok tartalma még nem válik a gép számára kezelhetővé. A tudást belülről kell strukturálni, és e strukturált tudást azokhoz a dolgokhoz kell kapcsolni, melyekre vonatkozik.

Ilyen meggondolásból mondhatjuk, hogy alapvetően a tudásábrázolás az a "ragasztó", ami összekötí a mesterséges intelligencia kutatások nagy részét, bár, mint ezt a továbbiakban igyekszünk kifejteni, sajátos problémaköre van.

Amikor átfogó tudást próbálunk ábrázolni, olyan kérdésekkel találjuk szemben magunkat, mint

* Hogyan strukturáljuk az explicit tudást egy tudásbázisban?

* Hogyan kódoljuk azokat a szabályokat, melyek egy tudásbázis explicit tudásával manipulálva a tudásbázisbeli implicit tudásra következtetnek?

* Mikor következtessünk és hogyan vezéreljük a következtetéseket?

* Miként írjuk le formálisan egy tudásbázis szemantikáját?

* Mihez kezdjünk a hiányos tudással?

* Hogy válasszuk ki a szakértői tudást a tudásbázis kezdeti "feltöltéséhez"?

* Az idő múlásával miként tegyünk szert automatikusan új tudásra, hogy a tudásbázis időszerű maradjon?

A számítógéptudományban egy jó megoldás gyakran a jó ábrázoláson múlik. A legtöbb mesterséges intelligencia alkalmazásban még a szokásosnál is nehezebb az ábrázolás megválasztása, hiszen a lehetőségek lényegesen nagyobbak és a kritériumok kevésbé tiszták. Az ábrázolás megválasztása azért döntő, mert az ábrázolásnál használható primitívek és az ezeket egyesítő rendszer lényegesen korlátozza azt, hogy e rendszer mit képes érzékelni, tudni vagy megérteni.

2. Tudásábrázolási technikák

Jóllehet a legkorábbi mesterséges intelligencia rendszerek legtöbbjében közvetett módon, szabályokon és adatstrukturákon keresztül testesült meg a tudás, a tudásábrázolás fontosságát nem ismerték fel kifejezetten. Néhány korai kutató közvetlenebbül célozta meg az ábrázolás kérdését. A SIR következtető rendszer [RA'68] például LISP tulajdonságlistákat használt a felhasználóktól szerzett információk ábrázolására, és az ezekből való következtetésekre; a Deacon rendszer [CR'66] gyűrűszerkezetekkel kódolt sokféle tudást, beleértve az időváriáns információt; míg F. Black [BL'68] hangsúlyozta a tárolt tudásból való következtetések és e következtetések vezérlésének szükségességét.

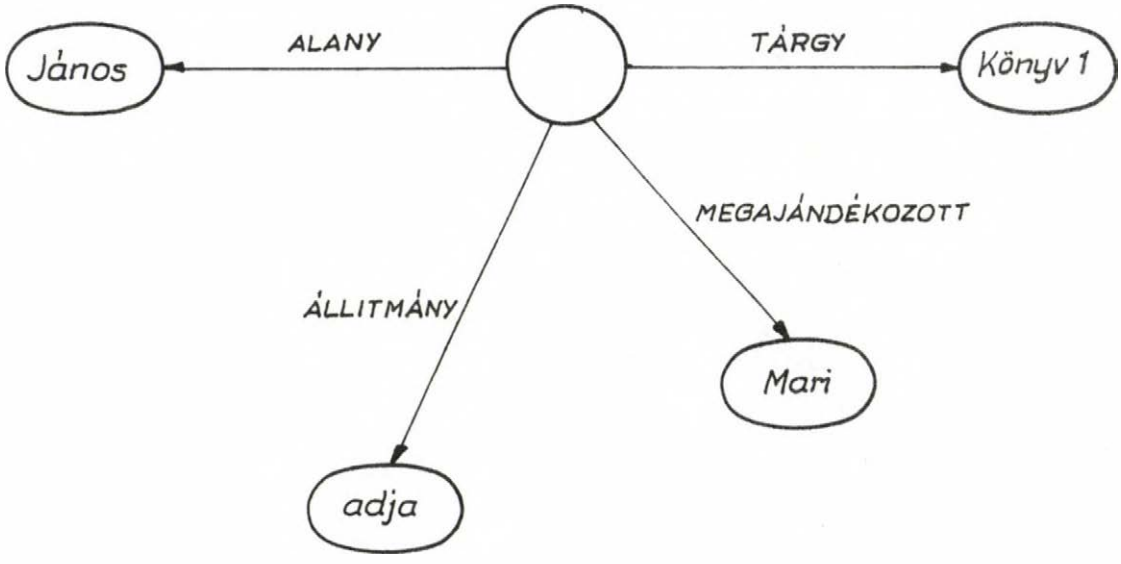
Az 1960-as évek közepén a tudásábrázolás lassan önálló tudományterületté vált. Számos tudásábrázolási megközelítés kezdett kialakulni, melyek eredményei a ma használatos különféle formalizmusok lettek. A legfontosabb jelenlegi megközelítések:

- a szemantikus hálók,
- az elsőrendű logika,
- a frame-ek és
- az un. production vagy szabályokon alapuló rendszerek.

2.1 Szemantikus hálók

2.1.1 A szemantikus háló "fogalma"

Az 1960-as évek közepén-végén M. Quillian [QU'68], S. Shapiro [SW'71] és mások kezdték fejtegetni azt, ami később szemantikus hálózat ábrázolási sémaként (semantic network representation scheme) vált ismertté. Az 1. és 2.ábrán jellegzetes szemantikus hálók láthatók, a szokásos grafikai jelöléssel.

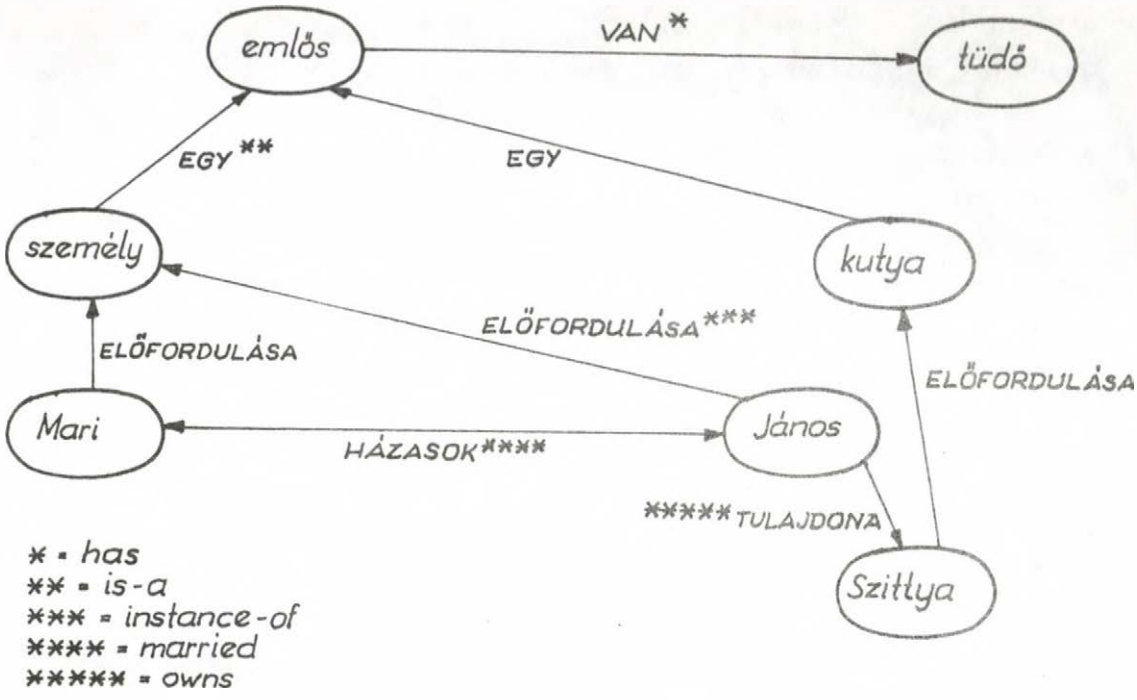


1.ábra. "János a könyvet Marinak adja"

A szemantikus hálózatokat eredetileg az emberi felismerő memória egy meglehetősen durva modelljének szánták. Minden szemantikus hálózat egy irányitott, cimkézett gráfnak tekinthető, melynek szögpontjai entitásokat vagy fogalmakat - pl. Nagy János személyét, a "narancs" szót, vagy az egyetemi hallgató fogalmát -, mig összekötő részei bináris kapcsolatokat ábrázolnak - pl. Nagy János kapcsolatát feleségéhez, a "narancs" és "gyümölcs" szavak közti szemantikus kapcsolatot, vagy az egyetemi hallgató fogalma és a személy fogalma között lévő kapcsolatot. A szemantikus hálózatok fő jellemzője ez az asszociativ nézőpont, ami nyilvánvaló grafikai ábrázolást kinál, és egyben a fogalmak közti hozzáférési utak definiálására is használható. Különböző dimenziók mentén néhány ilyen hozzáférési út szemantikus hálózatok szervezésére használható, és ezt meg is valósitották.

Jóllehet manapság sokféle hálót használnak tudásábrázolásra, csaknem minden séma szögpontok, azaz ábrázolt fogalmak adatstrukturájából és az adatstruktúrán működő speciális következtető eljárások halmazából áll.

Sok tudásábrázolásra szolgáló rendszert tekintenek szemantikus hálózatnak jórészt azért, mert az ábrázolandó világbeli dolgok osztályainak kategorizálásában főszerepet adnak egy kifejezett rendszertani hierarchia, egy fa- vagy rács-struktura fogalmának. A hierarchia gerincét valamilyen, ábrázolt objektumok közötti "öröklődési" kapcsolat képezi. Ez a gyakran "IS-A" néven emlegetett, bár "IS", "SUPERC", "AKO", "SUBSET" stb. néven is ismert kapcsolat tűnik talán a szemantikus hálók legstabilabb elemének, ha végigtekintjük a szemantikus hálók kialakulását.



2.ábra. Egy szemantikus háló (IS-A hierarchia).

Sajnos ez az állandóság illuzórikus. Az IS-A kapcsolatnak csaknem annyi jelentése van, ahány tudásábrázolási rendszer készült. Ezt támasztja alá az a vita is, ami a logika és a szemantikus hálók hivei között kerekedett. A logikusok úgy vélték, hogy a szemantikus hálók csak egy mutatórendszert nyújtanak a formulához, amit ugyanúgy, vagy talán jobban is ki lehetne fejezni az elsőrendű predikátum kalkulus nyelvén [HA'77, HA'79, IB'81]. A vita érdekessége az volt, hogy valahányszor a logikusok megpróbálták leszögezni az IS-A kapcsolat célját, a hálózat hivei azt állitották, hogy nem értik meg a lényegét. Ugyanez lett a hálók közötti összehasonlitások sorsa is. Egy sémát aszerint kritizáltak, hogy a kritikus mit gondolt az IS-A kapcsolatok jelentéséről, mig a szerző azon az alapon védte a sémáját, amit ő gondolt

arról, amit csinálni akart. Az IS-A kapcsolat jelentését gyakran addig süllyesztették, hogy "mit tesz vele a kód" - sem megfelelő szemantikai fogalom, sem hasznos vezető nem állt rendelkezésre a kapcsolat valódi jelentésének kiókumulálásához.

2.1.2 Az IS-A kapcsolat eredete

A szemantikus hálók korai történetének elején a kutatók megfigyelték, hogy sok ábrázolás foglalkozik olyan fogalmi relációkkal, melyek angol nyelvű megfogalmazásában "is a" szerepel. Pl. "John is a bachelor", azaz "János nőtlen", vagy "A dog is a domesticated carnivorous mammal", azaz "a kutya háziasitott, ragadozó emlős". A mesterséges intelligencia alkalmazások tudásábrázolási rendszerei tehát olyan állitásokat kezeltek, melyekben túlsúlyban voltak

- a predikátumok, melyek azt fejezik ki, hogy egy egyed (János) bizonyos tipusú (fenti példánkban nőtlen), és

- az általánosan mennyiségileg minősitett ($\forall$) feltételes állitások, melyek azt fejezik ki, hogy egy tipus (a kutya) egy másik tipus (az emlős) altipusa [BR'83].

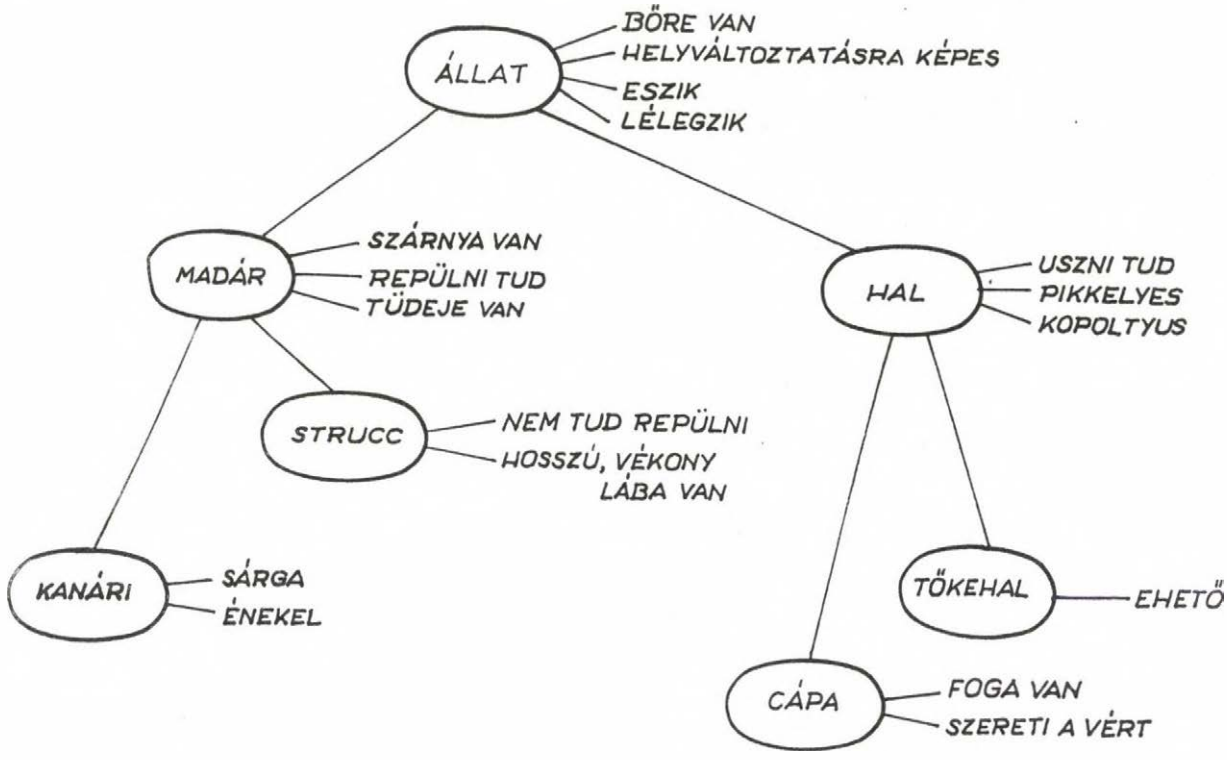
Ilyen állitások szemantikus sémába foglalása akkor a legegyszerűbb, ha van egy olyan kapcsolatunk, ami közvetlenül ábrázolja az ilyen mondatok "is-a" részét. Igy született meg az IS-A kapcsolat ötlete.

Hamarosan felismerték azt is, hogy az IS-A kapcsolatok hierarchiát, vagy néhány esetben rácsot alakitanak ki az összekapcsolt tipusok között, azaz az IS-A reláció durván egy részben-rendezés a tipusok között. A hierarchikus szervezés megkönnyitette a "tulajdonságok" oly módon való szétosztását, hogy a közös tulajdonságok a hierarchia azon pontján helyezkedjenek el, ami lefedi a rajtuk osztozó szógpontok maximális részhalmazát (3.ábra). Ez a szervezés a szemantikus hálót hatékony tárolási sémává tette, mivel a közös tulajdonságok nem ismétlődnek meg mindenütt, ahol igazak. A tulajdonságok öröklésének fogalma alatt tehát azt értjük, hogy a tulajdonságokat "örökli" minden szógpont, ami a tulajdonságokat tároló szógpont alatt van. A tulajdonságok öröklésének fogalmát látszólag mindig az IS-A kapcsolat velejárójaként emlitik.

Ha a tulajdonság-örökitő IS-A kapcsolatok egy hálózatának mintáját egyszer létrehozták, akkor e hálózat kidolgozottabb állitásokhoz, leirásokhoz stb. való felhasználására új sémákat kell kifejleszteni [BR'79].

2.1.3 Mit takar az IS-A kapcsolat valójában?

Mig a szemantikus hálók sikert arattak a tudásábrázolás ágazataként, talpkövük - az IS-A kapcsolat értelmezése - jelentősen megingott. Az IS-A kapcsolatot mindenekelőtt bizonyitható, főleg alapértelmezésként használt mondatok formálására alkalmazták. Azonban az IS-A kapcsolat sok más dolog jelentéseként is szerepelt, csaknem lehetetlenné téve a hálózatok és a többi módszer összehasonlitásait.



3.ábra. Tulajdonságok elosztása egy szemantikus hálóban

[BR'83] megállapitja, hogy az IS-A kapcsolattal alapvetően kétféle dolog fejezhető ki, éspedig azok, melyek

- egy fogalomból egy másik fogalmat képeznek, és azok, melyek

- valamilyen kijelentést tesznek két halmaz közötti relációról vagy két predikátum argumentumairól.

Ez utóbbiaknak négy alkotórészük van:

- a kijelentés bizonyitó ereje (assertional force);
  Az IS-A által ábrázolt kijelentést ténymegállapitásnak tekintjük-e vagy sem?

- az állitás módozata;
  Az IS-A által ábrázolt igazság vajon szükséges-e vagy csupán esetleges, és igy másképp is lehetne szemlélni?

- az állitás mennyiségi minősitője;
  Az állitás tartalmát vajon általánosan igaznak kell-e tekinteni, vagy éppen "igaznak, hacsak kifejezetten nem érvénytelenitik"?

- és az állitás mátrixa vagy tartalma.
  Aszerint, hogy általános/általános, illetve általános/egyedi IS-A kapcsolatról van-e szó, az IS-A kapcsolat tartalma egy részhalmaz ill. eleme reláció, vagy egy IF... THEN... feltételes állitás ill. predikátum.

Bár Hayes [HA'79] felvetette, hogy az IS-A kapcsolattal ábrázolt összekötés valójában az IF... THEN... feltételes állitás, az IS-A kapcsolatot többnyire mégis alapértelmezésként használják. Azaz a "MADÁR REPÜL" állitást rendszerint a madarakra vonatkozó igazságnak tekintik mindaddig, amig kifejezetten vissza nem vonják, vagy nem érvénytelenitik.

E megközelítést egyszerűen az indokolja, hogy tulajdonság-érvénytelenítés nélkül általában a szemantikus hálóban nem lehet kivételeket ábrázolni. A világ pedig olyan, hogy a kivételek a tudásábrázolás fontos szempontját képviselik. Az IS-A kapcsolatot alapértemezésnek tekintve, egy szemantikus háló ugyan foglalkozhat kivételekkel teli világgal, de sajnos ez az értelmezés nem teszi lehetővé néhány fontos dolog kifejezését.

Az IS-A kapcsolat alapértelmezéskénti használatának egyik sajátos következménye, hogy a szógpontokról intuitiv nevük ellenére sem gondolhatjuk, hogy a neveik által sugalmazott fogalmakat ábrázolják. Ehelyett egyszerűen az alapértelmezés szerinti tulajdonságkötegek tartópontjai, az IS-A kapcsolatokkal kifejezett tulajdonságok szigorúan egyirányú jellege és érvénytelenithetősége miatt. Azaz, ha Jumbó elefánt, akkor rendelkezik az elefántok tipikus tulajdonságaival.

Ez a szabály azonban ellenkező irányban nem működik. Ha Jumbónak tipikus elefánt tulajdonságai vannak, akkor nem következtethetünk arra, hogy Jumbó elefánt, mivel akármi lehet. Pl. előfordulhat, hogy olyan zsiráf, ami egyetlen tipikus zsiráf tulajdonsággal sem rendelkezik - ezek mind érvénytelenek -, és pontosan olyan tulajdonságai vannak, melyek az elefántoknál tipikusak. Igy az ELEFANT szógpont valójában nem az elefánt fogalma, hiszen egyetlen tulajdonsága sem foglalja össze tömören az elefánt definicióját. Ehelyett az ELEFANT szógpont a tipikus elefánt tulajdonságok gyűjteménye. Vitathatatlan, hogy nincsenek elefántságot definiáló tulajdonságok. Az elefánt "természetes fajta", mint ahogy a legtöbb, ha nem is minden olyan fogalom, amivel egy mesterséges intelli-

gencia rendszernek foglalkoznia kell. (Eltekintve persze az absztrakt és definiált matematikai fogalmaktól, és nem keveredve filozófiai vitákba.)

Az IS-A kapcsolat alapértelmezésként használatának másik következménye, hogy még a legegyszerűbb fogalmi összefüggéseket sem tudjuk ábrázolni. Pl. a legtöbb, amit a "kékszemű elefánt" állítással tehetünk az, hogy kijelentjük, hogy egy KÉKSZEMŰ-ELEFÁNT tipikusan kékszemű. Egy mesterséges intelligencia rendszer egy szigorúan alapértelmezésre épülő hálózatot csak olyan tudásbázisként tud használni, ami tartalmazza a felhasználó által célszerűnek tartott osztályozó tényeket, pl. Jumbó EGY ELEFÁNT, de a rendszer önmaga egyetlen ilyen következtetés levonására sem képes. Anélkül, hogy kifejezetten megmondanánk neki, egy ilyen rendszer nem tudná egészen kétségtelenül, hogy egy kékszemű elefánt elefánt.

Bár az intuitiv érzés az, hogy az érvénytelenités a kivételek értelmes eseteinek kezelésére korlátozható, az az igazság, hogy az érvénytelenités egészen könnyen megenged bizarr, nem intuición alapuló struktúrákat is. Könnyen alakithatnánk ki olyan strukturákat, amik azt mondják, hogy

- "az indiai elefánt elefánt",

- "Jumbó indiai elefánt", és

- "Jumbó nem elefánt".

Az önkényes érvénytelenités nem túl konstruktiv.

2.1.4 Mit nem takar az IS-A kapcsolat?

A tulajdonságok öröklése szigorúan megvalósitási kérdés, és a szemantikus hálók kifejezési fölényének egyetlen megvilágitásában sincs semmi súlya.

A hálózatos rendszereknek az a jellegzetessége, hogy a tulajdonságok bármely kifejezése a "legáltalánosabb helyen" van, könnyen lemásolható egy logikai rendszerben is. Ehhez nem is kell mást tenni, mint a tulajdonság axiómákat egyszerűen a legáltalánosabb predikátumokhoz társitani és a szabványos feltételes állitások megteszik a többit.

Az öröklés csupán a tulajdonságok szemantikus hálóban való tárolásához szükséges idő-/hely-igény egy lehetséges csökkentése. Néhány esetben mérhetetlenül könnyebb, ha minden tulajdonságot kifejezetten az alkalmazási helyen tárolunk és igy lerövidítjük a keresési időt.

Noha az IS-A reláció részkomponensekre bontható, szemantikus célból a korábban tárgyaltak, azaz a bizonyitási erő, módozat stb. a fontosak, nem pedig az "add át ezt a tulajdonságot" és a "ne add át amazt". Ámbár e két utóbbi nagyon hasznos lehet egy bizonyos IS-A módszertan megvalósitásakor, nem hozhatók fel érvként abban a vitában, hogy a szemantikus háló sémák tudásábrázolásra megfelelők-e vagy sem.

2.1.5 A szemantikus hálók szerepe a tudásábrázolásban

Összegezve tehát: a szemantikus hálók hagyományosan fő árnyoldala, hogy tervezőik fő támasza a szögpontok és az összekötő részek cimkéiből eredő intuició, nem pedig egy formális szemantika [WO'75].

Az IS-A többé-kevésbé szabványos használata - mint a késedelmi információ egy mutatója - néhány potenciálisan komoly problémát hoz magával. Egy erre alapozott hálózatot nem lehet bonyolult fogalmak ábrázolására használni, és az ebből eredő érvénytelenitési fogalom előre nem látható következményekkel járhat.

Az öröklés és az IS-A közti szoros gondolattársitás csak a dolgok további összezavarásának kedvez. Mivel az öröklés a megvalósitás és nem a kifejező erő kérdése, csak az örökléstől eltekintve tisztázhatjuk, hogy valójában mik az igények az IS-A-val szemben.

A lambda absztrakciót, az IS-A fogalomalkotó alakját is számitásba véve megállapithatjuk, hogy egy szemantikusan jól leirt szemantikus háló számitás ugyanolyan elfogadható és ésszerű gondolkodás, mint bármely más. Az IS-A fogalomalkotó stilusa és az ebből származó hálózat stilusu nyelvek a szabványos predikátum kalkulusbeli számitások valódi alternativái. Hiszen ha kölcsönös kapcsolatban lévő strukturált kifejezésekkel rendelkezünk, ez alapot ad a tárgykör leirására használt terminológia formális számitásához, mig a logika szabványos alakjai nem támogatják a definiált, nem-atomi predikátumokat.

A legfontosabb talán az, hogy a szemantikus háló stílusú ábrázolás olyan kényszerítő mintákat hangsúlyoz a tudásábrázolásban, melyek nem merülnek fel a predikátum logikára alapozott ábrázolásban. Pl. a fogalom/szerep paradigmának legalább egy értelmezése könnyen kifejezhető egy szabványos logikai nyelven [HA'79], de ez a minta csupán egy a végtelen sok közül. A hálózat sémák a beépített alak szintjére emelték a mintát a tudásábrázolásban való széleskörű használata kapcsán. Kényszerítő minta az IS-A kapcsolatnak az "is"-től való megkülönböztetése is. A szemantikus hálók - a típusokra alapozott érvelés uralkodó mivoltát tükrözve - szembetűnően megkülönböztették az "is"-nek a "John is a man" állításbeli jelentését az "is" minden más jelentésétől, az olyanoktól, mint a "John is running scared" (János megrémül) és a "John is extremely tall" állításbeli jelentések.

Mindent összevetve, a hálózat sémák IS-A kapcsolataikkal feltétlenül hozzájárulnak az ábrázolás világához. Sajnos e közreműködés jellege nem mindig tiszta, és nem mindig a kifejező erő a valódi kérdés.

[BR'83] szerint a jövő IS-A sémáiban gondosan meg kellene különböztetnünk a leírás- vagy kifejezés-alkotó operátorokat a mondatalkotó operátoroktól. Mivel a strukturált predikátumok vagy fogalmak fontosak a tudás kifejezésében, az ábrázolásban egy technikai szótárt kellene megőrizni, azaz egy olyan hálózat stílusú ábrázolási sémát, amiben a fő reláció a fogalmi elkülönítés IS-A kapcsolata. Ezt a technikai szótárt meg kellene különböztetni a világ tényeit kifejező hálózattól vagy axiómahalmaztól. Mivel ez a bizonyítási hálózat az,

ahol a világról állitásokat teszünk, e hálózatnak szüksége van a szabványos predikátum kalkulus kifejező és levezető erejére, amiről talán egy olyan szabványos kvantifikációs, vagy valamilyen hálózatszerűbb nyelv gondoskodik, ami felhasználja az IS-A kapcsolat mondatalakitó stilusát. Ezen IS-A háromtagú "prefixe" magában foglalná az állitás bizonyitó erejét, módozatát és mennyiségi minősitőjét (lásd 2.1.3). Ezt a stratégiát ábrázolási rendszerek tervezésére teljes mélységében feltárja [BFL'83].

2.2 Elsőrendű logika

2.2.1 Az elsőrendű logika és a tudásábrázolás

Az, hogy az elsőrendű logika hasznosan alkalmazható a tudásábrázolás területén, az 1960-es évek alatt vált nyilvánvalóvá, elsődlegesen a mechanikus tételbizonyitás kutatásának eredményeként. Sok kutatás irányult arra, hogy a rezolúciós elv következtetési módszerként való használatát tanulmányozza különféle alkalmazásokban, igy pl. a kérdés-megválaszolásban is. Más kutatás számitásra orientáltabb keretek között kisérelte meg újrafogalmazni a logikai formalizmusokat. A példák között meg kell emlitenünk a Planner formalizmust [HEW'72], a Strips tervezési paradigmát, és az újabban egyre szélesebb körben használt Prolog programozási nyelvet. Ha úgy tekintjük, hogy a mesterséges intelligenciát az 1956-os Dartmouth-i konferencián alapitották, akkor jogosan mondhatjuk, hogy a tudományterület kezdete óta vitatták a mesterséges intelligencia kutatók, vajon helyén való-e

ábrázolási formalizmusként olyan fajta nyelvet alkalmazni, amilyet a logikusok konstruálnak, használnak és tanulmányoznak.

2.2.2 Mennyire "logikai" egy tudásábrázolási rendszer?

Dióhéjban: a matematikai logika nyelveit nem általános használatra szánták. Kifejlesztőik nem léptek fel azzal az igénnyel, hogy általános szimbolizmusok legyenek tetszőleges alkalmazások esetén - azaz, hogy minden elképzelhetőt ki lehessen fejezni rajtuk. Valóban, még az sem tartható, hogy minden természetes nyelven mondható dolog kifejezhető egy formalizált logikai nyelven, és ugyanakkor biztos, hogy sok józan észnek nevezett ismeret fejezhető ki angolul. Természetesen az, hogy ezeket a formalizmusokat nem a mesterséges intelligencia problémájának megoldására tervezték, még nem jelenti azt, hogy ily módon nem használhatók.

A vita elég nyilvánvalóan akörül dúl, hogy milyen szerepet játszhat a "logika" a "mesterséges intelligencia problémájának" megoldásában. Minden érdekelt egyetért abban, hogy a kutatás egyik központi célja az, hogy a számítógépek valahogy "megtudják" egy jó részét annak, amit minden emberi lény tud a világról, és a világot lakó természetes és mesterséges organizmusokról. Ez, a határait tekintve kétségtelenül határozatlan tudástömeg az, amit "józan ész" alatt értünk. Azzal a problémával állunk szemben, hogy hogyan adjunk át ilyen tudást egy robotnak. Azaz, hogyan tervezzünk egy olyan robotot, amelynek következtető kapacitása elég erőteljes és eredményes ahhoz, hogy e tudás valamely résztömegét megadva,

a robot képes legyen a hiányzó részből eleget előállítani ahhoz, hogy intelligensen illeszkedjen környezetéhez és hasznositani tudja környezetét. Feltételezhetjük, hogy e józan ész alatt értett tudásnak ha nem is az egésze, de a legnagyobb része olyan általános ismeret, mint az, hogy a tárgyak leesnek, hacsak nincsenek alátámasztva, vagy hogy a fizikai tárgyak nem tűnnek el hirtelen, és az esőben megázhatunk [NIL'82]. Feltételezhetjük továbbá, hogy egy robotnak bizonyos tényekre vonatkozó tudása, pillanatnyi helyzetét tekintve, az érzékelőitől érkező input.

A "logika alapvető" nyelvének, az elsőrendű predikátumkalkulus nyelvének megfelelő, vagy helyénvaló voltával kapcsolatban felmerülő kétségek közül sok támaszkodott valamilyen tudástömeget formálisan ábrázoló, sajátos módszer kritikájára. Ezek a meggondolások leginkább a tárgykör tárgyainak, tulajdonságainak és kapcsolatainak felfogása, vagy ábrázolásmódja elleni kifogásként értelmezhetők, nem pedig a logika nyelve elleni támadásként. Ábrázolási formalizmusként egy logikai nyelv csupán eszköz. Ezen eszköz hatékonysága egy bizonyos feladat kivitelezése során attól függ, hogy miként használják. A szabványos logikai formalizmusokhoz való kötöttség nem jár egy bizonyos metafizikához vagy lételmélethez való kötöttséggel, még kevésbé akkor, ha ez különösen alkalmatlan.

Úgy tűnik, hogy a "logika ellenségei" (Minsky és követői) feltételezik, hogy a matematika és a józan ész közt gyanitott iszonyú szakadék, valamint az a tény, hogy a matematikai logika formalizmusait a matematikához szánták, és erre remek is, valahogy már garantálják az

ilyen nyelveknek a józan ész alatt értett ismeretek kezelésére való alkalmatlanságát. Ezen túlmenően, ez a garancia egyben felmenti a "logika ellenségeit" azon kötelezettség alól, hogy ténylegesen minden részletében kimutassák ezt az alkalmatlanságot.

Bizonyos kifogások valójában nem a logikai nyelvek kifejező képességét vették célba. Ehelyett inkább arra az igényre irányultak - amire a "logika barátai" (McCarthy és követői) feltételezhetően elkötelezettek -, hogy a józan ész szerinti érvelés vagy következtetés megfelelően megfogható egy helytálló tételbizonyítónak egy ilyen nyelv fölötti futtatásával.

Mindkét fél azt vallja, hogy ha ábrázolási nyelvként logikai nyelvet használunk, akkor ez elkerülhetetlenül valamilyen helytálló, algoritmikus, következtető apparátushoz, mint az új tudás, vagy az új meggyőződések központi vagy egyedüli, nem észleleti generátorához való kötöttséggel jár. Az első kötelezettség azonban teljesen független a másodiktól, és a második sokkal vitatottabb, mint az első. Egy formális logikai nyelvet és szemantikáját előírva szabadon előírhatunk bármilyen átalakitási szabályt. A szabályoknak nem kell helytállóaknak lenniük, csak mechanikusan végrehajthatóknak. Azaz "megengedett" alkalmazhatóságuk feltételeit kizárólag a mondatok szintaktikus struktúráinak kiértékelésével lehessen meghatározni. Pl. csinálhatnánk olyan szabályokat, amelyek kifejezik a hihető vagy valószinű érvelés, vagy éppen az analógián alapuló érvelés hasznos alapelveit. Fontos kérdés, hogy ezen szabályok alkalmazhatósága függhet a mondatokban lévő, nem-logikai leiró kifejezések előfordulásaitól csakúgy, mint a logikai konstansok

előfordulásaitól. Azaz, a szabályok egy sajátos tárgykörre irhatók elő, és nem kell helytállóaknak lenniök. Igy nem kell következtetéses apparátust képezniök [IS'83]

Az, hogy a szabályoknak nem kell helytállóaknak lenniük, nem jelenti azt, hogy a nyelv mondataival operáló levezető eljárások meghatározásakor nem vesszük figyelembe a nyelv szemantikáját. Ez egyszerűen kisérletezni enged minket. Egy olyan szabályhalmazt szeretnénk kialakitani, ami együttesen, szintaktikusan kodifikálható alakban fejezi ki az érvelés eredményes és általánosan megbizható módjait. Nem kell, hogy a szabályok a sajátjaink legyenek, és bizonyára nem kell őket önmegfigyeléssel felfedeznünk. Számunkra azonban feltétlenül elfogadhatóaknak kell lenniük. De hogyan lássunk azon gépesithető szabályok megtervezéséhez, melyek kifejezik a meggyőződés megállapitásának és felülvizsgálatának ésszerű elveit, ha nem fogjuk fel teljesen azon mondatok jelentését, melyekre a szabályok vonatkoznak? Ez a kérdés nagyon erős érv egy formális logikai nyelv használata mellett, nevezetesen: egy ilyen nyelvben pontos számitásokat kaphatunk arról, hogy a mondatok mit jelentenek.

2.2.3 A logika és az érvelés

Hogy kiemeljük azt a szabadságot, amivel a szabályok kiválasztásakor rendelkezünk, világosan meg kell különböztetnünk az érvelést és a bizonyitást.

Vegyünk egy egyszerű esetet és tegyük fel, hogy többek

között elfogadunk valamilyen "ha P, akkor Q" alakú mondatot, és ennek előtagját. A kérdés az, hogy elfogadjuk-e, el kell-e fogadnunk a következményt? A válasz: Nem szükségképpen. Lehetnek ugyanis rendkívül jó általános indokaink, hogy nem-Q-t higgyük, és ezek oda vezethetnek, hogy feladjuk vagy a feltételes állitást, vagy az előtagját. Ezen kivül, a bizonyitási szabályok lokálisak, tehát a mondatok egy adott halmazára vonatkoznak, egyedi szintaktikus alakjaik szerint. Az érvelés ezzel szemben gyakran lehet globális, hiszen meg kell próbálnunk az összes lényeges bizonyiték számitásba vétele mellett több bizonyitékot nyernünk, amennyiben a közvetlen bizonyitók elégtelennek bizonyulna.

Ez a döntés, valamint a jelentőségről és a bizonyitók súlyáról való döntések tipikusan az érvelés termékei.

A logikai bizonyitás tehát látszólag szembehelyezkedik az érveléssel. Úgy tünik, a megfelelő nézet az, hogy a logikai bizonyitás egy érvelésben alkalmazott eszköz. Ezek után "logikai érvelésről" beszélni helytelen, különösen akkor, ha ez azt az érvelést vonja maga után, hogy az érvelés "illogikus" vagy nem logikus.

Két további érvet kell még felhoznunk. Ha a tudomány történetét végigtekintjük, láthatjuk, hogy a logikusok, aki előállitják a tudástörzs axiomatikus formalizálásait - ha ilyen formalizálások egyáltalán adódnak - csak azután formalizálhatnak, miután a tudósok már befejezték a munkájukat. Azt azonban nem tételezhetjük fel, hogy a bizonyitásnak, a következtetéses levezetésnek nem, vagy csak lényegtelen része volt a tudományos munkában. Azon felül bár lehet rá okunk, hogy kételkedjünk a logikai

formalizmusoknak a meglévő tudás pontos kodifikálására és rendszerezésére való alkalmasságában, ezek a kételyek kívül esnek azokon az igényeken, hogy a következtetés szerint helytálló bizonyítási szabályok megfelelőek legyenek.

Az, hogy csupán következtetés szerint érvényes bizonyítási szabályokat követelünk meg, különösen erős igény. Azt mondja, hogy minden, amit egy robotnak szükséges tudnia, még ha egy erőltetett, de valós környezetben is, azon dolgok következtetéssel kapott következménye, amit "megmondtunk" neki - az érzékelői által átadott sajátos tényeket is beleértve. Ekkor az egyetlen mód, ahogy egy robot új dolgokat tanulhat - az érzékeléssel tanultakat kivéve - az, hogy következtetéseket von le abból, amit már tud.

2.2.4 A logikai ábrázolásmód és a kivételek

A logikai ábrázolásmód elleni kifogásmód sajátos esete a kivételekkel foglalkozó probléma, vagy a "nem-monoton logika". Minsky ezt különösen szembetűnő fogyatékosságnak tekinti [MIN'82]:

>> A logikai rendszerek nagyon jól működnek a matematikában, de az egy jól definiált világ. Csak akkor mondhatunk valami olyat, hogy "ha a és b egészek, akkor a+b mindig megegyezik b+a-val", ha matematikáról van szó. ... Tekintsünk egy olyan tényt, mint amilyen az, hogy "A madarak repülnek." Ha úgy véljük, hogy a józan ész szerinti érvelés olyan, mint a logikai

érvelés, akkor azt hisszük, hogy vannak olyan általános elvek, amik azt állítják, hogy "Ha Joe madár és a madarak repülnek, akkor Joe repül." Mi van, ha Joe strucc vagy pingvin? Nos rendben, axiomatizálhatjuk, és mondhatjuk azt, hogy ha Joe madár, és Joe nem strucc vagy pingvin, akkor Joe repül. De mi van, ha Joe elpusztult? Vagy ha közben valójában lép? A logikával az a probléma, hogy nem tudunk megszabadulni attól, amire egyszer következtettünk. <<

Ez a logikával kapcsolatos probléma állítólagosan a monotonitásnak köszönhető: ha egy S mondat egy A mondathalmaz logikai következménye, akkor S (még) logikai következménye bármely A-t tartalmazó mondathalmaznak is. Ezért, ha azt gondoljuk A-ról, hogy magába foglalja azt a meggyőződés-halmazt, amivel indultunk, új meggyőződések hozzáadása nem vezethet a régi következmények "logikai" megtagadására. Ahogy McCarthy fogalmaz [MCA'80]:

>> Helyes axiomatizáció az, amiben minden olyan következtetésre van bizonyítás, amit szokásosan levonnak ezekből a tényekből. De a józan észről tudjuk, hogy ez túl nagy kérés. Másféle, nem monoton érvelésre van szükség... Ha tudja, hogy kocsim van, arra következtethet, hogy megkérhet egy kocsikázásra. Ha azt mondom önnek, hogy a kocsi az üzletben van, arra következtethet, hogy nem kérhet meg egy kocsikázásra. Ha azt mondom önnek, hogy két órán belül kinn lesz az üzletből, arra következtethet, hogy megkérhet. (A premisszák hoz-

záadásával változik a következtetés.) <<

[IS'83] szerint a logikának ez az állitólagos hiányossága egyáltalán nem hiányosság, és valójában közvetlenül semmi köze a logikához. A logika nem mondja meg sem azt, hogy mihez ragaszkodjunk, sem azt hogy mit vessünk el. Ez az érvelés dolga, ami biztosan nem-monoton folyamat. Új dolgok kitalálása vagy elhivése gyakran jó indok a régi kedvencek megtagadására. Kezdeti meggyőződéseink elvetése egyáltalán nem illogikus, különösen akkor nem, ha azért vetjük el őket, mert az, amit logikailag maguk után vonnak, ellentétben van azzal, amit elsőprő indokok alapján hiszünk. Ez az egyik oka annak, amiért érvelés esetén nem beszélünk premisszákról. Az érvelés általános jellegéhez valami másnak van köze, és ez a más semmivel sem kevesebb, mint egy teljes elmélet. Biztosan furcsa úgy gondolni egy egész elméletre, mint egy premisszára.

A logika nemcsak azt nem mondja meg, hogy mely meggyőződéseket vessük el vagy tartsuk meg, de még kevésbé mondja meg, hogy mit tegyünk, amikor segitségével felfedezzük, hogy ellentmondó meggyőződéseink vannak. Ebben az esetben csak azt mondja meg, hogy összes meggyőződésünk nem lehet igaz. Az a tény, hogy sok szabványos logikában bármi és minden következik egy ellentmondásból, teljesen jelentéktelen, ha világosan megkülönböztetjük a logikát és az érvelést, és a logikát az érvelésben alkalmazott eszköznek tekintjük. (Vannak olyan tökéletesen szabványos nyelvek fölött definiált logikák, melyekben nem következik minden egy ellentmondásból.)

A Minsky és McCarthy által felvetett probléma mély. A kritikus pont mindazonáltal az, hogy a nem-monoton logi-

károl való vitában egy érv sem szól az ellen, hogy a mesterséges intelligencia területén ábrázolási nyelvként valamilyen szabványos szemantikai számitással rendelkező szabványos logikai nyelvet használjanak. Az egyetlen követelés az, hogy a "logikát" a neki megfelelő helyen tartsuk. Ahogy Minsky mondja:

>> De a "logika" egyszerűen egyáltalán nem érvelési elmélet. Meg sem próbálja leirni, hogy hogyan működhetne egy érvelési folyamat. Csupán annak részelmélete, hogy hogyan korlátozzunk egy ilyen folyamatot... <<

Nem lehet tehát kétséges, hogy a logika formális pontossága és értelmezhetősége hasznos, és egyben olyan kifejező képességet ad, ami más tudásábrázoló sémákból hiányzik.

2.3 Frame-ek

2.3.1 A frame szerepe a tudásábrázolásban

Minsky természetesnek tartotta, hogy egy tudásbázist hasznos úgy szervezni, hogy nagyon moduláris, "majdnem szétszedhető" nagy darabokra, un. frame-ekre tördelik. E néha "schemata"-nak is nevezett frame-ek váltak a tudás-ábrázolás egy másik fő iskolájának alapjává. A tudásbázis frame-ekre osztása sokféle alkalmazásban általános, igy pl. számitógépes felismerő rendszerekben (vision system) és természetes nyelvek megértésénél. A frame-ek különösen alkalmasak bizonyos változatlan fogalmak vagy

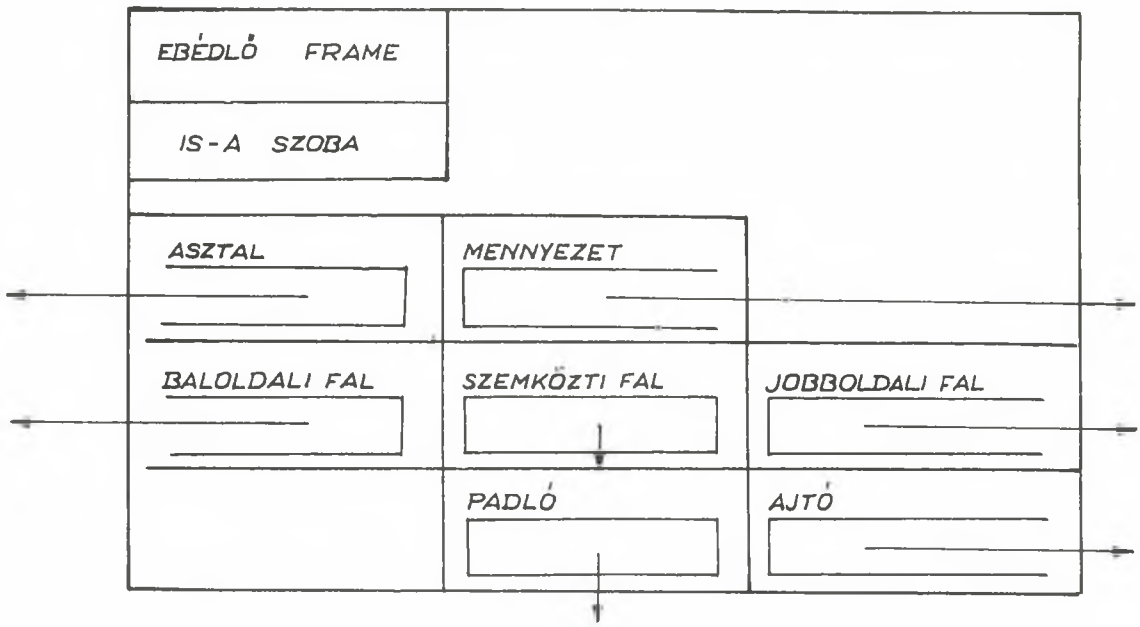
események tudásának ábrázolására (4.ábra).

Egy ilyen szabványos fogalom vagy esemény felismerésekor a megfelelő frame-ben lévő slotokba (frame változókba) a valóságos szereplőket vagy cselekményeket ábrázoló jelek (token) tölthetők. Ezután a lépés után sok "előre összeállitott" tudás böngészhető ki közvetlenül a frame-ből. Természetes, hogy sok tudás valószinüleg csupán következtetéssel is levezethető, és a frame-ek sok megközelitése részletesen foglalkozik azzal, hogy hogyan és mikor következtessünk igy. Gyakran megkülönböztetik a kis következtetési képességü scripteket és az eljárás orientáltabb frame-eket. Ez a megkülönböztetés idézte elő az eljárás és az adatstruktura közti kapcsolatra vonatkozó vitát, az un. eljárási/deklarálási vitát.

2.3.2 A frame jelentése

A frame-eken alapuló (frame-based) nyelvek kifejlesztésénél nagy hangsúlyt fektettek a következő tulajdonságokra:

- a legfontosabb ábrázolási objektumok vagy frame-ek néhány bonyolult fogalom nem-atomi leirásai,
- a frame-ek általánosabb frame-ek specializációiként definiáltak,
- az egyedeket a generikus frame-ek példával való szemléltetései ábrázolják, és
- a frame-ek közt adódó kapcsolatok taxonómiákat alkotnak.



4.ábra. Egy sablonos ebédlő frame-je. A kapcsolatok "slotokat" ábrázolnak és sajátságos mennyezet-, fal-, ajtó-, asztal- és padló-frame-ekre mutatnak egy adott ebédlő felismerésekor.

A tervezésben és a használatban problémát jelent, hogy

- a strukturákat különböző időkben különbözőképpen értelmezik (ez a kétértelműség elsősorban a definiciós és a tényleges értelmezés között szembetűnő);

- az ábrázolás jelentését csak a megvalósitásra használt adatstruktúrák irják le;

- a hálózatok tipikusan öröklèses jellegúek.

Egy tudásábrázoló nyelvben elófordulhat pl. a család következó leirása:

család
IS-A szociális szociális-struktura
apa: férfi pontosan 1
anya: nó pontosan 1
gyerek: személy

Ez az adatstruktúra - még ha nem is értelmezzük - tagadhatatlanul hasznos. A tudásábrázoláshoz azonban az ábrázoló objektumokat értelmezni kell, azaz az ábrázoló objektumoknak jelenteniük kell valamit. Egy frame-taxonómiában az összekótó részeknek és szógpontoknak kivételesen sok értelmezés adható, és a tipikus frame rendszerek a szemantikus munka jó részét az olvasóra hagyják [BR'83].

A sok lehetséges értelmezés ellenére úgy tünik, hogy a frame-ek jelentésének két megközelitése ismétlódik:

- a frame-ek állitások vagy kijelentések a dolgok világi létezéséról [HA'79], illetve

- a frame-ek közvetlen állitási értelem nélküli leirások (lásd KL-One).

Az elsó értelmezés szerint a "család" frame jelenléte

egy frame rendszerben azt állitaná, hogy minden család egy társadalmi struktúra: egy apával, egy anyával és valahány gyerekkel. A második értelmezés szerint viszont a "család" szimbólum a következő leirás rövidítése: "egy szociális struktúra, amiben - többek között - van egy apa, aki férfi, egy anya, aki nő és valahány gyerek, ezek mind személyek". Azért "többek között", mert az, hogy egy frame kifejez-e elégségességi feltételeket vagy sem, lényegében attól függ, hogy strukturálisan vagy állitásként olvassuk-e.

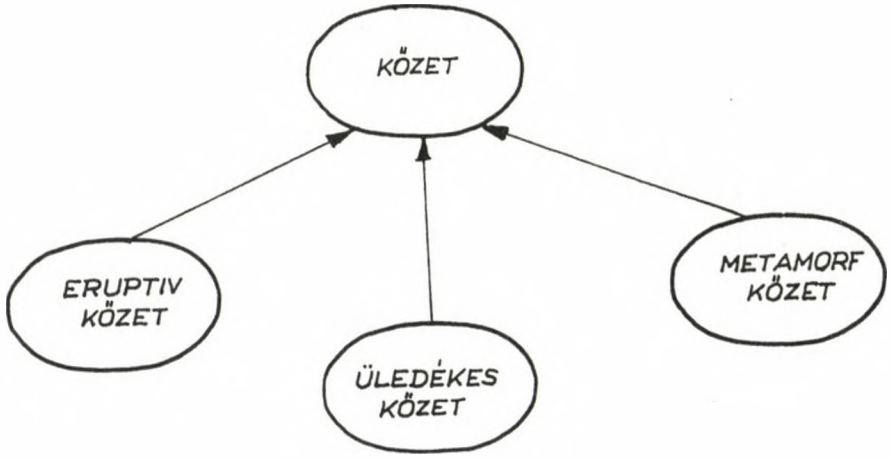
A frame-ek állitásként való értelmezése nagyon erősen korlátozza az ábrázolást [BR'83,WO'75,LE'82].

- Ilyen rendszerekben az állitás alapvető formájának példával való szemléltetése, azaz egy frame slotjainak megtöltése a hiányos tudás kifejezését nehézzé vagy lehetetlenné teszi. Pl. egy tipikus frame rendszerben nem lehet kijelenteni, hogy "vagy a Riska, vagy a Bimbó legel az Új élet Mgtsz legelőjén".

- Igazán összetett leirások sem fejezhetők ki. Pl. ahelyett, hogy ki tudnánk alakitani egy "gyermek nélküli család" leirását, csak egy "gyermektelen-család" frame-t hozhatunk létre, és azt állithatjuk, hogy az ilyen tipusú családokban nincs gyerek - mintha ez egy olyan esetleges tulajdonság volna, mint az, hogy mindkét szülő dolgozik.

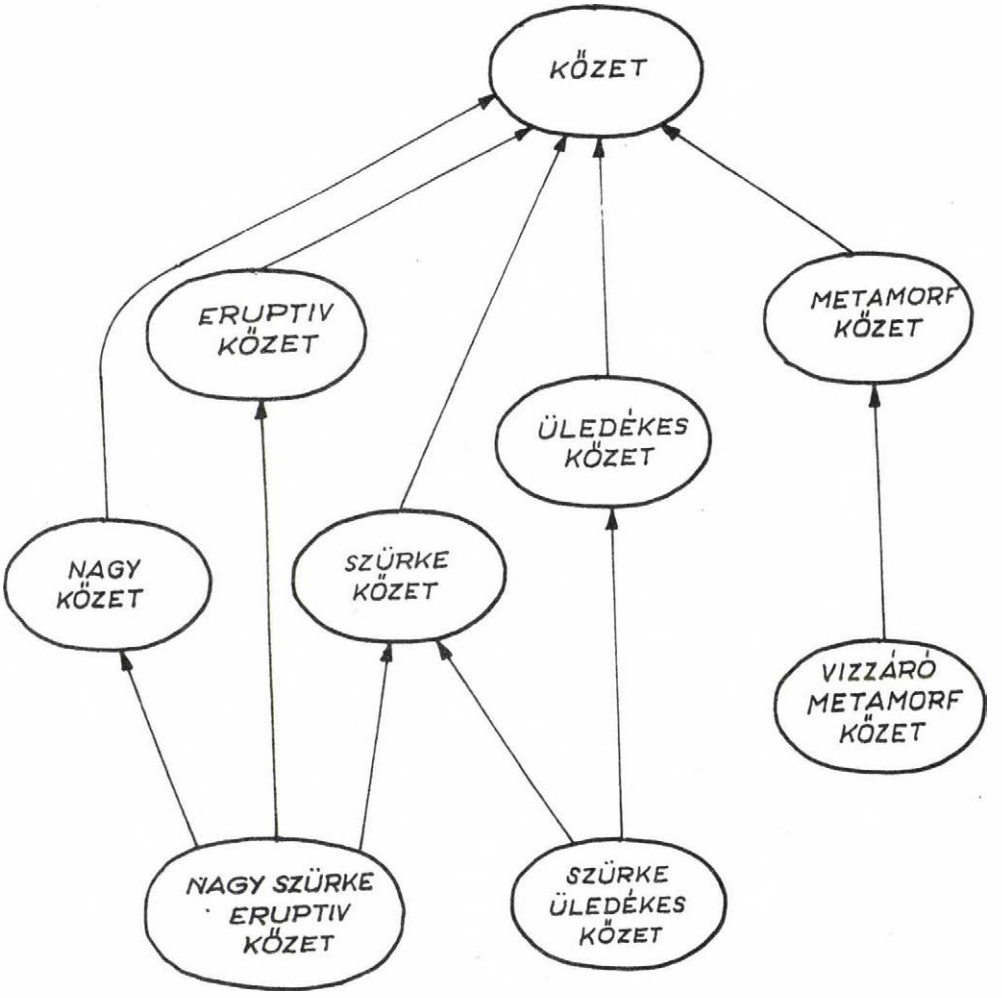
Azok, akik a frame-ek leirásként való értelmezését támogatják, úgy vélik, hogy ez a felfogás világosabb nyelvet

hoz létre, ami nem szenved azoktól a problémáktól, melyek a szigorúan állitásos frame rendszereket sújtják. Sajnos a nem-állitásos megközelités tisztasága sem oszlat el teljesen minden, az összekötő részek és a frameek félreérthetőségével kapcsolatos félelmet ezekben a rendszerekben. Még egy szigorúan értelmezett strukturális frame rendszer is csupán egy, szimbolikus adatstruktúrák kezelésére alkalmas programcsomag. Következésképpen a felhasználó vagy a felhasználói program levonhat nem igazolható következtetéseket. Tekintsük pl. a kőzetfajtákat ábrázoló struktúrát (5.ábra).



5.ábra. Kőzetfajták

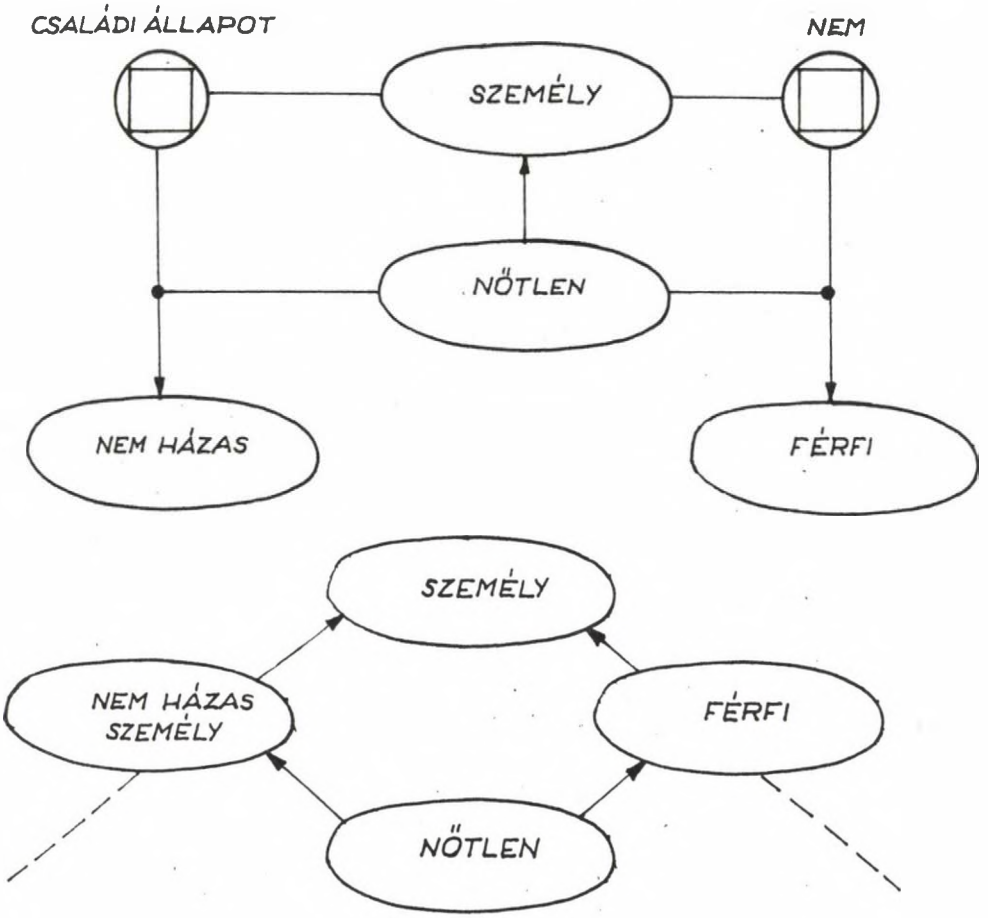
Azt gondolhatnánk, hogy könnyen felelhetünk a "hány féle kőzet van?" kérdésre, csupán meg kell számolnunk a "kőzet" alatti szögpontokat. A fajták megszámlálása azonban struktúrált hálózatban értelmetlen, mivel a nyelv megengedi, hogy számtalan olyan leíró kifejezés legyen az "eruptiv kőzethez" kapcsolva, mint pl. "nagy, szürke, eruptiv kőzet" (6.ábra).



6.ábra. A kőzetfajták egy struktúrált hálózata

Hasonlóan csábító, amikor a felhasználó hozzáfér egy hálózat összekötő részeihez. A 7.ábra a "nőtlen" két különböző ábrázolását mutatja. E két esetben a "nőtlen" és a "személy" közötti távolságok különbözők. A szemantikus hálóbeli feldolgozások elterjedőben lévő aktiválási elméletei ezt a távolságot lényegesnek tekinthetik. A szögpontok közti távolság az alapértelmezés szerinti tulajdonságok öröklésében is lényeges, amikor sor kerül a "legközelebbi" érték keresésére. Egy nem állitásos frame rendszerben azonban az összekötő részek egyszerűen kifejezések, azaz termek képzésére szolgálnak, és nincs esetleges állitási következményük vagy pszichológiai értelmük.

Összegezve az eddigieket, úgy látszik, hogy frame-eket használó tudásábrázoló rendszerekben legalább két komoly problémával állunk szemben. Először is vigyáznunk kell a struktúrák egyértelmű értelmezésére, nehogy egy kapcsolatot egyszer állitásként, másszor pedig egy kifejezés vagy term jelentésének részeként értelmezzünk. Ezen túlmenően foglalkoznunk kell azzal az alapvető problémával, hogy csak adatstruktúrákat kezelünk. Így nem igazolható következtetések áldozataivá válhatunk az adatstruktúrák megszámlálásakor, vagy annak feltételezésekor, hogy bizonyos struktúrák jelenléte/hiánya jelent valamit.



7.ábra. A "nőtlen" két különböző ábrázolása

2.4 Szabályokon alapuló rendszerek

2.4.1 Szabályokon alapuló rendszerek működési elve

A mesterséges intelligencián belül a tudást gyakran ábrázolják feltétel-következmény, ill. minta-tevékenység párokból álló szabályok halmazaként, un. production rendszerként. Ezt a modellt, amit először A. Newell és H. Simon [NS'72] javasolt az emberi felismerési architektúra modelljeként, a mesterséges intelligencia sok területén használták, a nyelvfeldolgozástól a szakértő rendszerekig. Egy production rendszer programja elsődlegesen a feltétel-következmény szabályok, azaz productionok egy halmazából, valamint egy dinamikus munkamemóriából áll. Az új változatok a fogalmak korlátlan memóriájára törekszenek. Egy ilyen rendszer ciklusokban működik. Minden ciklusban minden egyes szabály feltétel részét szembeállitja a munkamemória pillanatnyi állapotával. A sikeresen illeszkedő szabályok közül egyet vagy többet kiválaszt alkalmazásra. Egy szabály alkalmazásakor a szabály következmény része befolyásolja a munkamemória állapotát, új szabályok illesztésére késztetve a rendszert. Ez a ciklus addig ismétlődik, mig nem lesznek illeszkedő szabályok, vagy egy stop parancsot nem talál a rendszer.

2.4.2 Szabályokon alapuló rendszerek és a modularitás

A production rendszerek igéretes jellemzőinek egyike a modularitásuk. Mivel minden szabály viszonylag független minden más szabálytól, elvileg olyan moduláris programo-

kat szerkeszthetünk, melyek új szabályok beszúrása vagy régi szabályok törlése után is tovább futhatnának. Természetesen ilyen rendszereket sokkal nehezebb megvalósitani, mint tárgyalni, és a mesterséges intelligencia és felismerés-szimuláció irodalmában leirt sok production rendszer program közül csak néhány igazán moduláris.

R. Young [YO'76] hosszan folytatódó feladatra alkalmas gyermekfejlődési modellje rendelkezett moduláris jelleggel. A gyerekeknek az egymást követő fejlődési állapotokhoz tartozó modelljeit egy bázis modell új szabályokkal való bővitése állitja elő.

D. Klahr és R. Siegler [KS'78] hasonló fejlődési modellt gondolt ki a Piaget-féle egyensúlyi arány (balance scale) feladatra. Négy állapotú modelljük minden állapota egy vagy két szabály hozzáadásával keletkezik a megelőzőből.

Végül R. Young és T. O'Shea [YOS'81] is modularitást alkalmazott a gyermekek kivonási stratégiáinak modellezésekor. Modelljük tovább fut, ha bizonyos szabályokat elhagynak vagy hozzátesznek, de az eredményül kapott viselkedés hibákra vezet, és a szerzők modelljüknek ezt az oldalát használták a gyermekek kivonási viselkedésében lévő hibák megmagyarázására.

E kutatók mindegyike a rendszerükben lévő szabályok közötti kölcsönhatások minimalizálásával érte el a modularitást. Azáltal, hogy minden szabályt olyan nagy résszé tesznek, ami felelős a viselkedés néhány szempontjáért, minden szabály komoly ártalom nélkül hozzáférhetővé vagy elmozdithatóvá válik, noha a rendszer

viselkedése természetesen megváltozik.

A mesterséges intelligencia egyik központi kérdése a tanulás, azaz a viselkedésnek tudásszerzés által történő megváltoztatása. A modularitás és a tanulás közti kapcsolat világos: a tanulásba beleértjük a tudás bővitését, és a tudást úgy kell bővíteni, hogy az új tudás jó kölcsönhatásban legyen a meglévő tudással. Bár a production rendszerrel történő megközelités valószinűleg nem az egyetlen modularitást adó formalizmus, minden esetre azon kevés ábrázolási sémák egyike, melyet e vonatkozásban sikeresen kipróbáltak (lásd Sage tanuló program).

2.4.3 Az alkalmazandó szabály meghatározása

A viselkedés feltétel-következmény párok halmazával való jellemzése sok alkalmazásban nagyon természetes módnak bizonyult a szabályokra alapozott tudás kivonatolására és kódolására [WHR'78]. Manapság a production rendszereket elterjedten használják speciális célú, tudásra alapozott, un. szakértői rendszerek szerkesztésére [NA'83]. E feladat megkönnyitésére különféle absztrakt production rendszereket dolgoztak ki, mint pl. az Emycin és társai. A 8.ábra egy tipikus szakértő rendszer szabályaira mutat példát.

P_1 IF the category is a form or a colouring or a texture
THEN the object has an appearance

P_2 IF the category is an appearance or an odor or a tactile-quality
THEN the object has an external quality

P_3 IF the category is a size or an external quality or a mass or a substance
THEN the object has a physical_quality

...

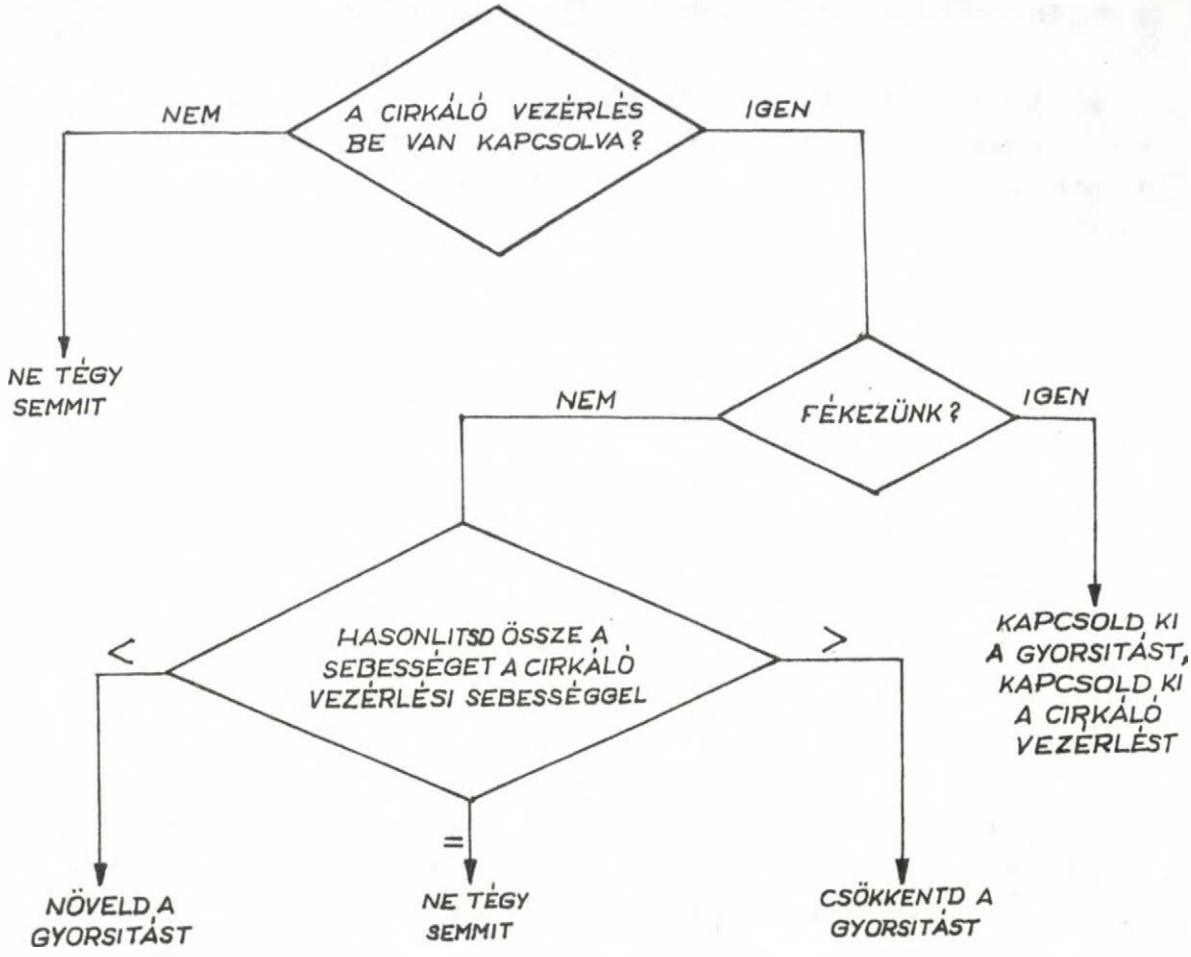
P_4 IF ...
THEN ...

8.ábra. Egy tipikus szakértő rendszer szabályai [MCC'83]

Bár a szabályokon alapuló rendszerek elvileg úgy működnek, hogy a rendszerszabályok mindegyikét megvizsgálják, a szabályok számának növekedésével módszereket keresnek arra, hogy elkerüljék minden szabály vizsgálatát (9.ábra).

Az alkalmazandó szabály meghatározásához feltételezik, hogy az összes ilyen szabály feltétel része mindazon helyzetek és tárgyak strukturált rendszertanába szervezett, melyekről a rendszer tud valamit. Rendszertan alatt általánositási IS-A relációkkal olyan módon összekapcsolt fogalmakat értenek, melyben az általánosabb

fogalmak a kevésbé általános fogalmakból elérhetők.



9.ábra. A megkülönböztető háló a faktorizált tudás-struktúra egy fajtája [WO'83]

Struktúrált rendszertanról pedig akkor beszélnek, ha a fogalom-leirásoknak van egy olyan, a számitógéprendszer

számára hozzáférhető belső struktúrája, amelyben pl. a fogalomnak a rendszertanon belüli elhelyezkedése számitással meghatározható. Az ilyen rendszertan egyik jellemzője, hogy az információ alkalmazhatóságának legáltalánosabb szintjén tárolható és a specifikusabb fogalmak szintjén indirekt módon érhető el. Örökithető az információ.

Ha ez a rendszertani szerkezet rendelkezésre áll, a rendszerszabályok következmény része a megfelelő struktúrabeli szögpontokhoz rendelhető. Igy egy adott helyzetre alkalmazható szabályok meghatározásának feladata a helyzetnek a rendszertanon belüli osztályozásából és a következmények örökitéséből áll. Egy ilyen fogalmi rendszertan a rendszerszabályok feltétel részeit olyan hatékony struktúrába szervezheti, ami megkönnyiti a felismerést (lásd KL-One).

2.5 Egyéb kutatások és irányzatok

Egy olyan kiterjedt, állandóan mozgásban lévő kutatási területet, mint amilyen a tudásábrázolás, túlegyszerüsités a szemantikus hálók, az elsőrendü logika, a frame-ek és a production rendszerek négy elegánsan független ideológiájára bontani. Az ábrázolás néhány megközelitése e területek egyikéhez sem kapcsolódik valójában. Az analóg ábrázolások (analogical representation) [FU'76] például természetesnek tartják, hogy közvetlen szerkezeti kapcsolat legyen a való világ tárgyai és ezek ábrázolásai között, azaz, hogy legyen egy olyan térképezés, ami arra törekszik, hogy egyéni beállitottságú, alkalmazás-függő megközelitésekhez vezessen. Egy másik példa

S. Fahlmannek a nagymértékben párhuzamos, hálózat stílusú, következtető hardver szerkesztésével kapcsolatos munkája [FA'79], amit nehéz szigorúan kategorizálni. Ez az egyszerű négy területre való osztás figyelmen kívül hagyja a területek közti gyakori kereszteződéseket is, nem beszélve a nagyszámú, valamennyi megközelítést érintő általános kérdésről. Valójában e közös kérdések egyre határozottabb felismerése teszi a tudásábrázolást a mesterséges intelligencia különálló és növekvő fontosságú területévé.

A kutatások mindmáig erőteljesen foglalkoznak azzal a kérdéssel, hogy hogyan definiáljuk a tudásábrázolás pontos dimenzióit és formális alátámasztásait. Sok kutató kísérelte meg körvonalazni a tudásábrázolás problémájának alapvető dimenzióit [WO'75, SGC'79]. Számos fronton folyik a kísérlet, hogy pontos szemantikát adjanak a tudásábrázoló formalizmusoknak. Igy a nem-monoton logikával kapcsolatos munkák úgy próbálják meg kiterjeszteni a predikátum kalkulust, hogy a jelenségek egy szélesebb választékát kezelje, megtartva formális szemantikáját. Ugyanakkor a szemantikus hálókkal kapcsolatos néhány kutatás [FI'79] megpróbálja az elegáns strukturáltságot és következtető képességet megtartva formalizálni a szemantikus hálókat. A tudásábrázoló nyelvek - pl. KRL [BWI'76], Microplanner [HEW'72] - egy értelmező pontosságával próbálják definiálni, hogy mit jelentenek a különféle tudásábrázoló szerkezetek. A legtöbb ilyen nyelv a tudásnak frame-szerű objektumok hálóiban történő ábrázolását támogatja, és már számos nyelv megjelent a piacon is [RU'86].

A tudásábrázolás egyik sajátos problémája, hogy hiányos

pontosságú információ ábrázolására van szükség. Az egyik megközelités a fuzzy logika, ami látszólagos pontatlanságot kezel azáltal, hogy különféle numerikus hihetőségi mértéket kapcsol a javaslatokhoz. Egy ettől eltérő megközelitésben G. Hendrix [HEN'79] arra összpontosit, hogy a szövegösszefüggés eltolódásával hogyan válik bizonyos tudás láthatóvá vagy láthatatlanná. A nem-monoton logikával kapcsolatos néhány új munka formális mechanizmusokat próbál adni a befejezetlenül előirt tudás kezelésére.

A legtöbb kutatás a tudásszerzés témakörével kapcsolódik. A mellőzöttség évei után úgy tűnik, hogy a tudásábrázolás területén dolgozó kutatók eléggé magabiztosak az ábrázolási formalizmusok alapvető alakjaiban ahhoz, hogy olyan rendszerek épitésével kisérletezzenek, melyek automatikusan naprakész állapotra hozzák ezeket a formalizmusokat. P. Winstonnak a strukturális leirások tanulására [WIN'75] és P. Langley-nek a fizikai törvények felfedezésére vonatkozó kisérlete [LA'83] a probléma különféle oldalaira összpontositott.

Sok mesterséges intelligencia kutatót érint olyan gyakorlati rendszerek épitése, ami valós világi visszacsatolást ad a tudásábrázolás törekvéseinek [NA'83]. E rendszerek közül sok, de nem mind alapszik production rendszer felépitésen és kisszámú szakértői körben megfelel az emberi teljesitménynek, vagy felülmúlja azt.

3. Tudásábrázolási módszerek értékelési szempontjai

[RU'86]-tal szemben nem az a célunk, hogy a tudáskezelő rendszereket, illetve a tudásábrázolási módszereket mint eszközöket értékeljük. Minket elsősorban az érdekel, hogy mi alkot egy jó ábrázolási rendszert, és mi az ábrázolási primitiveknek az a jó halmaza, amivel kezelhető a tudáskörök nyilt végű tartománya. "Ábrázolási primitivek" alatt nemcsak primitiv fogalmakat értünk, hanem főként azokat a primitiv elemeket és operátorokat, melyekből megszerkeszthető a tanult fogalmak nyilt végű tartománya.

A fontos kérdések olyan problémákat érintenek, melyek akkor keletkeznek, amikor megkisérlünk olyan intelligens számitógépes programokat szerkeszteni, melyek valamely feladat végrehajtásához tudást használnak. Példaképpen néhány ilyen probléma:

- hogyan strukturáljunk egy olyan ábrázolási rendszert, ami elvileg minden fontos megkülönböztetésre képes: néhány esetben ugyanis az állandónak vélt attributumok idővel változhatnak.

- hogyan maradjunk semlegesek a fel nem oldható részletekben;

- hogyan ismerjük fel hatékonyan, hogy mely tudás lényeges a rendszer számára egy adott helyzetben;

- hogyan sajátitsunk el tudást dinamikusan a rendszer élettartama alatt, és

- hogyan asszimiláljuk a tudás részleteit: a felmerülés vagy inkább az átadás különleges, megkívánt sorrendjében.

3.1 A kifejező erő és a hatékony jelölés

A tudásábrázolás két aspektusát kell kiemelten figyelembe venni. Az első az ábrázolás kifejező ereje, vagyis az, amit az ábrázolás mondhat. A kifejező erő két komponensét azok a megkülönböztetések alkotják, melyeket egy reprezentáció tehet, illetve melyeket nem határoz meg pontosabban a részletes tudás kifejezéséhez. A második a hatékony jelölés, ami mind az ábrázolás tényleges alakjával és szerkezetével, mind e szerkezetnek a rendszer működésére gyakorolt hatásával kapcsolatos. A hatékony jelölés alkotórészei a különféle következtetések számítási hatékonysága, az ábrázolás tömörsége és a módosíthatóság problémái.

A kifejező erőt azért fontos megkülönböztetni a hatékony jelöléstől, mert ezen a területen már sok vitát mérgesített el az, hogy nem tisztázták, e két kérdés melyikéről is van szó. Az az érv például, hogy elsőrendű predikátum kalkulust kellene használni, mert jól megértett a szemantikája, nem említi világosan a hatékony jelölés kérdését. Az érv úgy is érthető, hogy a logikusok által használt jelölések alkalmazását támogatja, és ez néhány esetben az is lehet, amit jelent. Azonban kitalálhatunk sok olyan különböző jelölési rendszert, melyek mindegyikének szemantikája megegyezik ugyan az elsőrendű logika szemantikájával, de mindegyik másként viselkedik a hatékony jelölés különféle alkotórészeit tekintve.

A tudásábrázolás kutatásának - hogy ésszerű alapokat adhasson a tudás gyakorlati felhasználásához az érvelésben, érzékelésben és tanulásban - olyan jelölési konvenciókat kell keresnie, melyek egyidejűleg szolgálják a kifejező erőt és a hatékony jelölést. Egy ábrázolási rendszertől megkívánjuk, hogy feleljen meg a különféle következtetések átfogó tartományának és adjon számítási előnyöket a gyakran és gyorsan végrehajtandó következtetéseknek. A gyorsan és hatékonyan végrehajtandó következtetések egyik osztálya lehet pl. valami vagy valaki legújabb helyzetének jellemzése egy rendszertanilag szervezett tudáshálózatra vonatkoztatva.

4. Tudásábrázoló módszerek alkalmazásai, tudásábrázolást támogató rendszerek

A tudásábrázolással kapcsolatos kutatások

- tanulmányozhatják, hogy hogyan ábrázolhatunk bizonyos szemantikai fogalmakat, pl. az időt, az okságot, a hiteket és az ajánlásokat;

- a nyelvtervezés alakját ölthetik, ahol a nyelvet tudásábrázolás támogatására szánják: az ezen a nyelven irt "programok" tudásbázisok, melyek a szakértelem valamely tartományára vonatkozó tudást tárolnak;

- magukba foglalhatják egy programozási környezet kifejlesztését, tudásbázisok épitéséhez és használatához.

A tudásábrázoló módszerek alkalmazásait is a fenti tárgykörök szerinti csoportositásban tekintjük át.

4.1 Sajátos szemantikai fogalmak ábrázolása

4.1.1 Kérdés-felelet folyamatokban keletkező következtetési problémák speciális kezelése [SPT'83]

A kérdés-felelet folyamatokban keletkező következtetési problémák speciális kezelés nélkül nagy számitási erőforrásokat köthetnek le. E következtetési problémák egy részében azt kell meghatározni, hogy milyen kapcsolat

van a dolgok két tipusa között, pl. a "személy" magában foglalja-e a "lányt", vagy a "lány" összeegyeztethető-e a "számitógéppel". A többiben pedig a tárgyak részei, a szinek vagy idők közti ilyen vagy ehhez hasonló kapcsolatok meghatározására van szükség.

A sajátos következtetési tárgykörőknek ez a gyűjteménye nem olyan véletlen, mint amilyennek tűnik. Bizonyos érzékszervekkel ellátott térbeli és időbeli teremtmények ként sajátosan modellezzük pl. a szinérzékelésünket. Sajátosan kategorizáljuk és hozzuk kapcsolatba azokat az entitásokat, melyeknek térbeli elhelyezkedése és időbeli állandósága felismerhetően összetartozóvá teszi őket (tipus rendszertanok). Sajátosan elemezzük a térbeli tulajdonságokat, úgy mint a részek struktúráját és ezen entitások világi viselkedését. Hogy a mesterséges intelligencia rendszerek illőek legyenek felismerő készségünkhöz, hasonló speciális módszerekre lesz szükségük.

Az itt leirt módszereket egy működő, deduktiv, kérdésekre válaszoló algoritmus kiegészitésére tervezték. Az algoritmus szemantikus hálóba szervezett logikai állitásokat kezel. A háló szelektiv hozzáférést enged az egyes "gondolatvilágok" és elbeszélések tartalmához, valamely előirt tipus entitásainak halmazaihoz, valamint valamely előirt entitást bevonó, és valamely előirt téma alatt osztályozott logikai állitásokhoz. Bizonyos tulajdonság-öröklési mechanizmusok könnyitik az információ átvitelt a generikus entitásokból és részeikből a sajátos entitásokba és részeikbe.

A felvetett kérdések előbb vagy utóbb elkerülhetetlenül

felmerülnek bármely általános tudásábrázolásban akár szemantikus hálókon, akár frame-ken, akár scripteken, akár production rendszereken vagy bármi máson alapszik is. Ezekben a névlegesen nem összeillő formalizmusokban ugyanis sok a közös vonás: pl. mind magában foglal egy predikátum kalkulus-szerű állitási nyelvet (még ha megreformált nyelvbe rejtve vagy kódba hálózva is), mindben csoportositható úgy az információ, hogy egy adott feladatra adott időben vonatkoztatott információ élesen korlátozható, és mindben vannak tulajdonság öröklési mechanizmusok (vagy rendelkezniük kell ilyenekkel).

4.1.1.1 Tipuskapcsolatok felismerése

Két problémakört vizsgálunk:

- tipusfogalmak párjaira vonatkozó gyors összeegyeztethetőségi vizsgálatokat, és
- tipus kifejezetten tipussal kapcsolatba hozó logikai állitások közvetlen kiértékelését.

A tipusfogalmak hierarchiájában minden állitás logikailag $[T \subseteq T_1,\ldots,T_k]$ alakú, ami azt fejezi ki, hogy a T tipusfogalom a páronként összeegyeztethetetlen $T_1,\ldots,T_k$ résztipusokra van osztva úgy, hogy ez a felosztás teljes. Minden fogalom zárójelbe tett számpárral cimkézett, ami preorder számából és az utódai közti legnagyobb preorder számból áll. Ha egy szögpont egy másik őse, akkor a hozzárendelt zárójelbe tett számpár - intervallumként tekintve - tartalmazza a másikhoz tartozó zárójelbe tett számpárt. Ha egyik sem a másik őse, akkor

a zárójelbe tett számpárok diszjunkt intervallumok [AHU'83]. Ez nyilvánvalóan megengedi a kompatibilitás és alárendeltség állandó idő alatt történő ellenőrzését.

A módszer valamelyest általánositható. A nem teljes felosztások maradék-kategóriákkal teljessé tehetők, igy a módszer ezekre is alkalmazható.

Lényegesebb, hogy átfedő hierarchiákra is kiterjeszthető, ha minden fogalmat külön zárójelbe tett számpárral látunk el minden olyan hierarchiára, amihez tartozik, természetesen egy megfelelő hierarchia-azonositóval együtt. Ekkor két fogalomra az összeegyeztethetőség és alárendeltség vizsgálata közös hierarchia azonositó keresésével kezdődik, és ha ilyen akad, akkor a zárójelbe tett számpárok összehasonlitásával folytatódik. Ez az ellenőrzés még mindig elég gyors, feltéve hogy nincs olyan fogalom, ami néhány - mondjuk két vagy három - hierarchiánál többnek része lenne. Széleskörű fogalom rendszertanok kidolgozására irányuló kisérleteink ezt a feltevést látszanak alátámasztani.

Az általánositott módszer azonban logikailag nem teljes, ahogy ezt a rész rendszertanok következő tárgyalásánál látni fogjuk.

4.1.1.2 A "part-of" kapcsolatok felismerése

Egy objektum "part-of" strukturája lényegében ugyanúgy ábrázolható, mint a tipusfogalmak rendszertana. Egy P objektum-felosztó relációt [x $\underline{P}$ $x_1,\ldots,x_k$] jelöléssel vezetünk be, kifejezve, hogy az x objektum - maradékta-

lanul - az $x_1,\ldots,x_k$ részekre oszlik. Ha egyszerűen azt akarjuk állitani, hogy x-nek van egy y része, akkor ezt [x P y z] irásával tehetjük meg, ahol z az esetleg üres fennmaradó rész.

A 10.ábra részleges emberi anatómiát mutat P-gráf formájában. Egy fő hierarchiából és három, a csontváznak, a végtagoknak, valamint a nyak és a törzs y-nal jelölt egyesitésének megfelelő az előbbire helyezett kiegészitő hierarchiából áll. (Elkendőzünk néhány logikai árnyalatot ezen gráfok értelmezését illetően, melyek szögpontjai generikus entitások, úgymint "medence" és "bal láb".)

A tipus párokra felvázolt algoritmust itt olyan állitás párok összeegyeztethetetlenségének megállapitására használhatnánk, mint pl.

x János medencéje; x János bal lába,

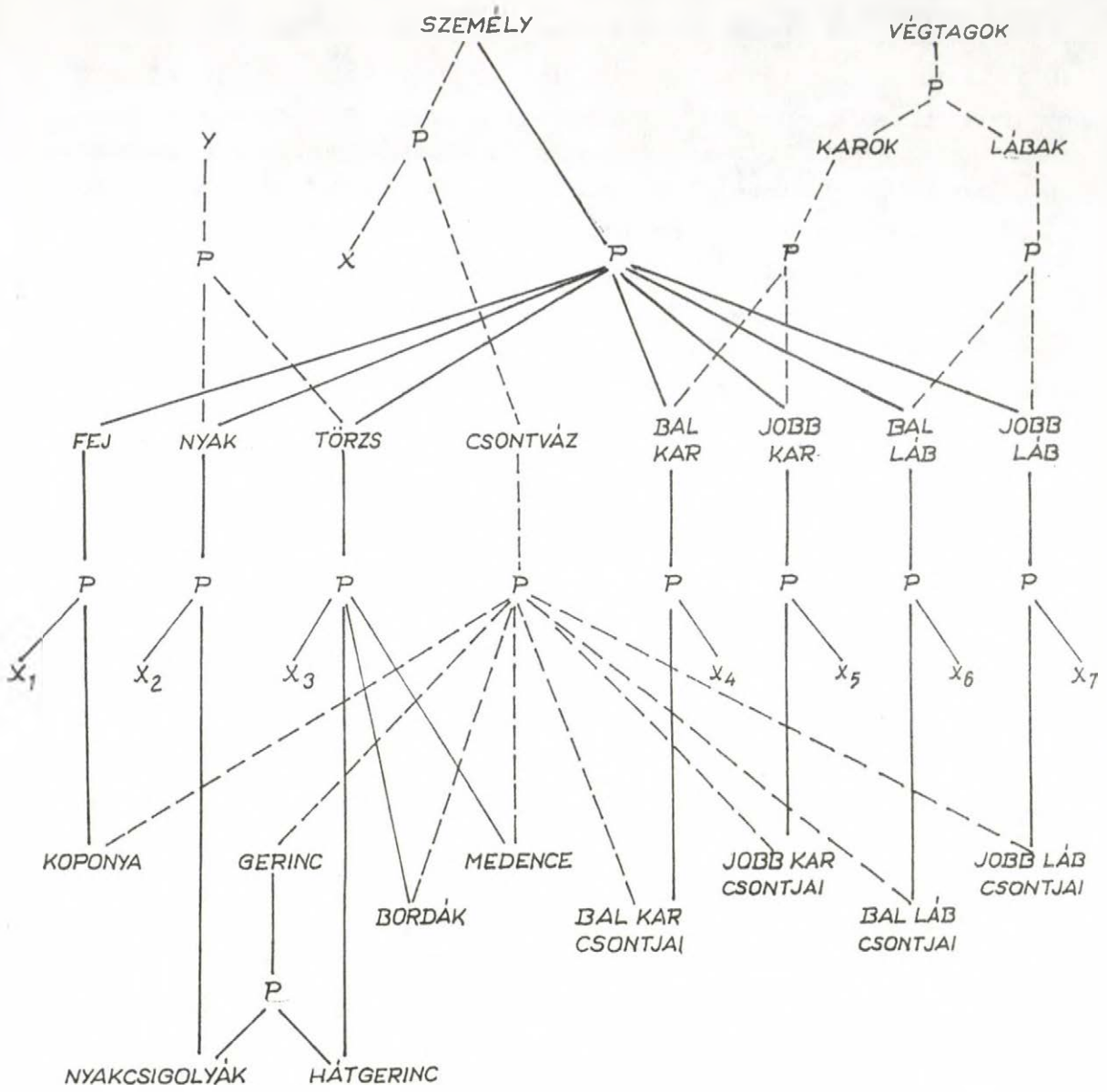
vagy olyan állitások igazságértékeinek meghatározására, mint

ha x János koponyája és y János csontváza,
akkor x része y-nak,

felhasználva az ábrabeli átfedő hierarchiák preorder számozásán alapuló zárójelezett számpárokat.

Tekintsük a következő kérdést:

A gerinc része y-nak?



10.ábra. Az emberi test felosztó gráfjának felső szintjei. Minden P-token egy felosztó állitást ábrázol, a fentről hozzákapcsolt szögpontot az alatta elhelyezkedő szögpontokra osztva. A folytonos vonalak a fő hierarchiát, a szaggatott vonalak a kiegészitő hierarchiákat definiálják.

A válasz "ismeretlen" lenne, mivel a "gerinc" és y nincsenek közös hierarchiában. A gráf mégis biztosan megengedi azt a következtetést, hogy a "gerinc" y része, mert egy P-állítás a gerincet maradéktalanul a nyakcsigolyákra és a hátgerincre osztja és ezekből van y-hoz felfelé vezető út. Hasonlóképp megkérdezhetjük, hogy a "gerinc" része-e a "végtagoknak" és a válasz ismét "ismeretlen" lenne, mivel a "gerinc" és a "végtagok" nincsenek közös hierarchiában. Ugyanakkor a gráfból negatív válasz vezethető le. A módszerek tehát nem teljesek és ez erőteljesebb módszerek keresésére ösztönzött.

Az [SCU'79]-ben ismertetett módszer teljes és hatékony a "zárt" P-gráfok osztályára. A 10.ábra gráfja zárttá tehető a logikailag redundáns [x $\underline{P}$ $x_1,\ldots,x_7$] P-állítással. A "félig zárt" P-gráfok még nagyobb osztályára [PS'81] egy módszerről bizonyították, hogy helyes és teljes [PAS'82].

Bár a zárt és félig zárt P-gráfok megadják a part-of és tipus struktúrák ábrázolásához szükséges hajlékonyság nagy részét, de még mindig az átfedő hierarchiák bizonyos fajtáira korlátoznak minket.

Sajnos csekély esélyünk van arra, hogy olyan módszert találjunk, amely egy P-gráf méretéhez viszonyítva csak lineáris tárat és időt igényel, mivel ehhez a híres $P \overset{?}{=} NP$ problémát kellene igenlően megoldanunk. Így valószínűleg meg kell elégednünk nem teljes speciális módszerekkel a tipusok és az alkotórészek taxonómiáira. Ez egyáltalán nem végzetes feltétel, mert a speciális módszereket az általános következtető algoritmusok

támogatására, nem pedig ezek helyettesitésére tervezik.

4.1.1.3 Szin kapcsolatok felismerése

Egy pontosabb jellemzéshez az alapszinek árnyalatait megjelölő kifejezéseket egybeesö és nem egybeesö részekre kell osztani. Az alapszinek határán terpeszkedő árnyalatok (türkiz) felosztásai csak tovább növelik e felosztások burjánzását. Ha olyan "kibúvó" szinkapcsolatokkal próbálunk foglalkozni, hogy a türkiz egyfajta kék és ugyanakkor egyfajta zöld is, akkor a szinfelosztó gráfnak legalább a szomszédos és/vagy nem szomszédos relációkkal is gyarapodnia kellene. Még e bővitések mellett sem lenne eszközünk az olyan tulajdonságok és kapcsolatok kezeléséhez, mint a halványság, tisztaság, telitettség, kiegészitő szinek és a meleg-hideg szinek megkülönböztetése.

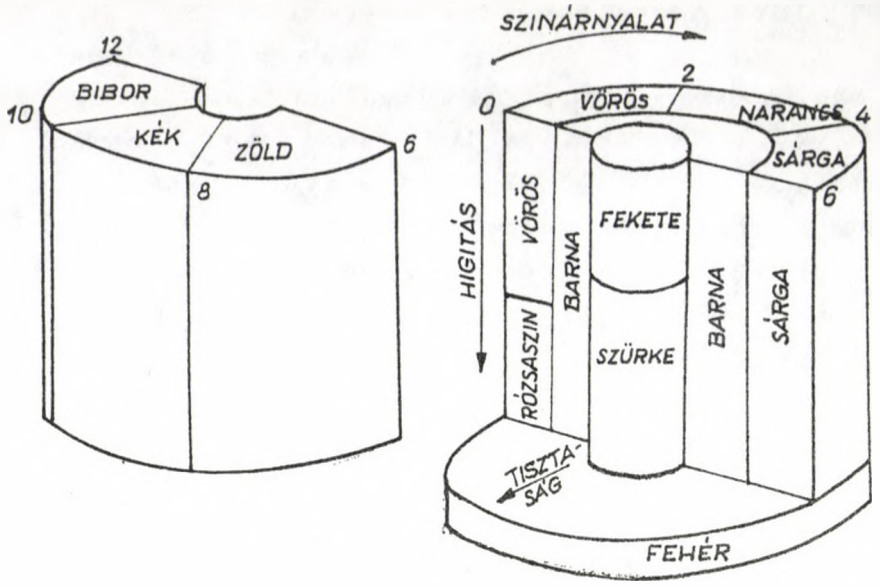
A választott megoldás hengerábrázolás: bármely szin egy tiszta, egyszerű szin valamely mennyiségéből, plusz bizonyos mennyiségű feketéből és fehérből tevődik össze. Az egyik dimenzió átfutja a szivárvány körben elrendezett, megszámlálhatatlanul sok árnyalatát, mig a további dimenziók a jelenlévő fekete és fehér mennyiségét paraméterezik (11.ábra).

A modell lefedi a felfogható árnyalatok teljes skáláját. Annyiban egyedi, hogy minden angol szin-kifejezést egyszerűen egy olyan régióként ad vissza, melyet - három, alsó-felsö paraméter párral definiált - hat koordináta felület határol. Az ábra egyszerű régióinak igazitásához tapasztalati vizsgálatokat terveznek.

E szin-geometriában kevés, rögzitett számú összehasonlitással ellenőrizhető a szin-régió párok között bármely kivánt kapcsolat, úgymint belefoglalás, átfedés és szomszédosság. Továbbá az olyan nem alapvető kifejezéseket, mint türkiz, vöröses barna, beige, skarlát stb., könnyű - az alapszinek régióihoz hasonlóan - koordináta felületekkel határolt régiókként definiálni. Az olyan tulajdonságok, mint a halványság és tisztaság, és az olyan kapcsolatok, mint a kiegészités, elég nyilvánvaló módon kiszámithatók.

Az előző tárgyalás során hallgatólagosan feltételeztük, hogy a szinhenger tartalmazza minden olyan szin kifejezett ábrázolását, melyre összeegyeztethetőségi ellenőrzés valaha is szükséges lehet. Ezen enyhithetünk.

Megengedhetjük, hogy bizonyos nem alapvető szinek csak az alapszinekhez való minőségi kapcsolatukkal legyenek definiálva. Pl. a türkiz axiomatizálható olyan szinként, ami egyszerre kék és zöld, és a "lime" olyan szinként, ami egyszerre zöld és sárga. Ezt egy olyan módszer támogatja, ami táblázat alapján összeegyeztethetetlennek minősiti az olyan jellemzést, mint pl. kék és sárga (és igy türkiz és lime). A táblázat az ilyen tipusu tagadott leirásokat is lefedi, a "sort-of" minősitővel és anélkül.



	SZINARNYALAT		TISZTASAG		HIGITAS	
FEHÉR	(0	12)	(0	1)	(.9	1)
SZÜRKE	(0	12)	(0	.2)	(.4	.9)
FEKETE	(0	12)	(0	.2)	(0	.4)
BARNA	(0	6)	(.2	.6)	(0	.9)
RÓZSASZÍN	(0	2)	(.6	1)	(.5	.9)
VÖRÖS	(0	2)	(.6	1)	(0	.5)
NARANCS	(2	4)	(.6	1)	(0	.9)
SÁRGA	(4	6)	(.6	1)	(0	.9)
ZÖLD	(6	8)	(.2	1)	(0	.9)
KÉK	(8	10)	(.2	1)	(0	.9)
BÍBOR	(10	12)	(.2	1)	(0	.9)

11.ábra. A 11 alapvető szín színárnyalat/tisztaság/ /higitás szintérben

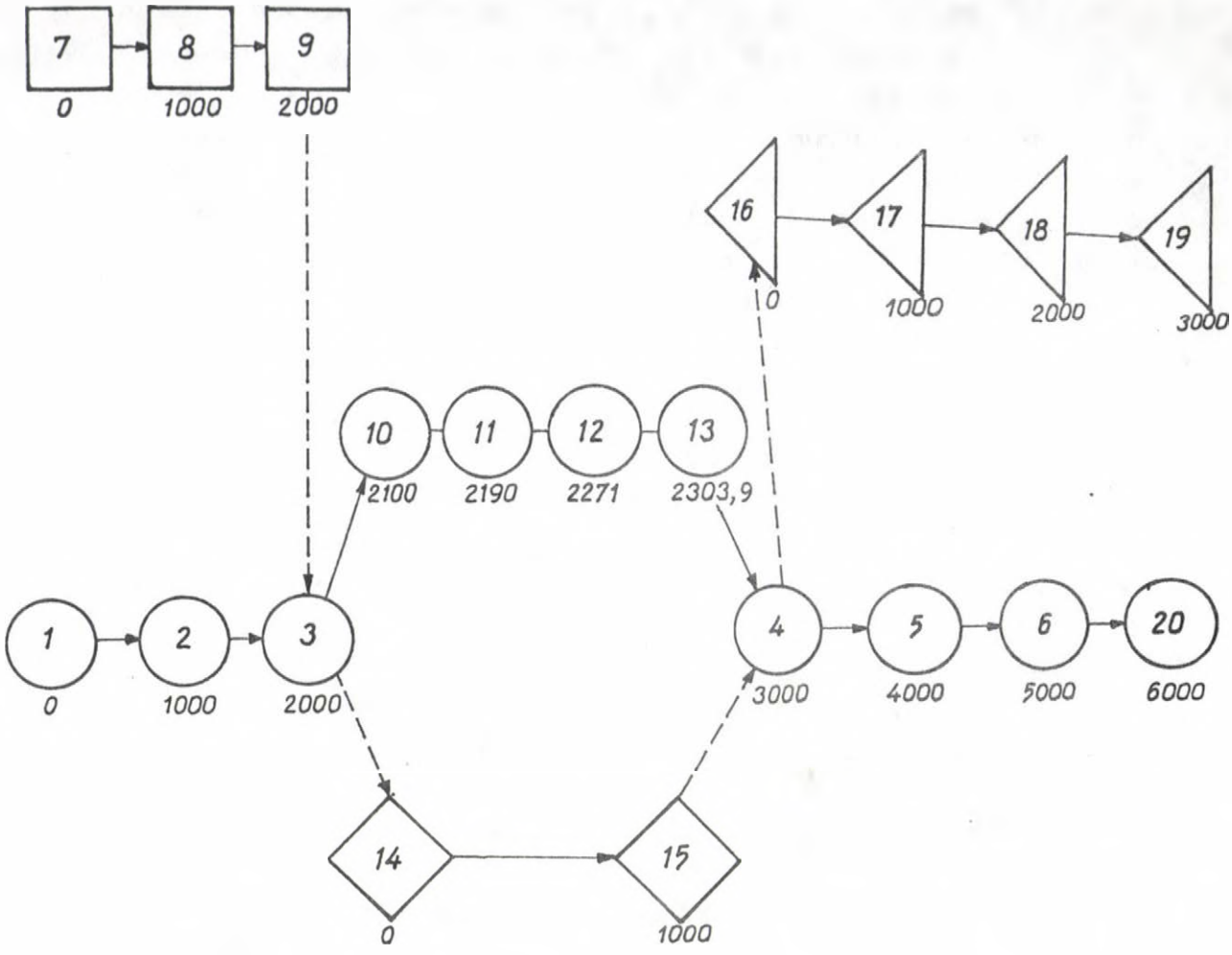
4.1.1.4 Idő kapcsolatok felismerése

Az időábrázolás az időintervallumokat végpontjukkal kódolja, és irányitott ív köt össze minden olyan időpont párt, melyeknek sorrendje kifejezetten ismert. Egy elbeszéléssel előállitott gráf tehát egy nem-periodikus digráf - kivéve, ha újrabelépéses időutazást mesélnek el a történetek. Bármely, a gráfban lévő implicit rendező reláció meglelhető, ha nyomon követünk egy olyan művelet, ami a gráf éleinek számához viszonyitva lineáris idejű.

Mint a többi tekintetbe vett következtetési tartomány esetében, most is a lineáris kereséssel szemben előnyben részesitenénk egy állandó idejű ellenőrzést.

Az ötlet a következő: az időpontokhoz időrend szerint numerikus értékeket, un. pszeudoidőket rendelünk, amikor a sorrend ismert. Amig ez a hozzárendelés lehetséges, két időpont időbeli sorrendje pszeudoidejük összehasonlitásával, állandó idő alatt ellenőrizhető.

A 12.ábrán egy elbeszéléssel meghatározott időgráf látható. A szögpontok időpillanatokat jelölnek és gráfbakerülésük sorrendjében számozottak.



12.ábra. Elbeszéléssel meghatározott időgráf

A pszeudoidők - ezerrel növelve, ha nincs felső korlát, ill. a pszeudoidő intervallum tizedrészével növelve különben - a szögpontok alatt láthatók. Az elbeszélés eseményei olyan időszögpont pároknak felelnek meg, mint 1,2 és 3,4. A 3 szögponttól a 4 szögpontig vezető

áthúzott összekötés a 10,...,13 szögpontok beszúrásakor törlődik. A gráf saját pszeudoidő sorrenddel rendelkező időláncok gyűjteménye. A különböző időláncokban különböző alakú szögpontok szerepelnek. Ez a megkülönböztetés valójában az egyes láncok numerikus tipizálásával történik. A tipusok közti kapcsolatokra egy különálló strukturát tartanak karban (az ábrán szaggatott vonalak jelzik). A gráfot felhasználhatjuk pl. a 2 és 18 szögpontok időrendjének ellenőrzésére. Miután meghatároztuk, hogy a kérdéses szögpontok különböző láncokhoz tartoznak, a láncok ill. tipusok közti kapcsolatokat vizsgáljuk és megkapjuk a 4-16 keresztösszekötést az első láncból a másodikba. Mivel a 2 szögpont pszeudoideje kisebb, mint a 4 szögponté, és a 16 szögpont pszeudoideje is kisebb a 18 szögponténál, a válasz "2 előbb van, mint 18".

Nyilvánvaló, hogy az egy láncra korlátozott időellenőrzéshez csak egy összehasonlitás kell; a legrosszabb esetben a láncokon keresztül történő időellenőrzések számitási ideje a láncok közti kapcsolatok számával arányos. Ez a szám tipikusan kisebb, mint az időgráfbeli összekötőrészek teljes száma. Ezt sokhasábos, újságban megjelent elbeszéléseknek, egy mesének, valamint Hemingway "Az öreg halász és a tenger" cimű kisregényének és egy európai történelemkönyv szemelvényeinek megfelelő időgráfok példáin ellenőrizték.

4.1.2 Reprezentációs kérdések tanuló rendszerekben [LA'83]

A tudásábrázolás módja befolyásolja a tanulás módját és meghatározhatja, hogy szóba jöhet-e egyáltalán tanulás. Ha egy rendszer pl. kiértékelő függvényeket használ kutatásának irányitására, akkor a tanulás jobb kiértékelő függvények felfedezését foglalja magában. Ha azonban egy rendszer feltétel-következmény szabályokat használ a viselkedés meghatározására, akkor a tanulás olyan új szabályok előállitását foglalja magában, ami megváltoztatja ezt a viselkedést.

Az a rendszer, amivel először foglalkozunk, valójában egyetlen új szabályt sem tanult; az új szabályokat a programozók szúrták be. A Sage [LA'82] olyan production rendszer, ami keresési heurisztikákat tanul tapasztalatból. A program megkapja a kezdeti- és cél-állitásokat, valamint egy szabályhalmazt, hogy törvényesen átléphessen néhány problématerületet. Kezdetben csak azokat a jogos feltételeket ismeri, amelyek alatt lépésjavasló szabályai alkalmazhatók. Miután kezdetben ezen szabályok mindegyike egyenlő súlyú, a Sage-nak addig kell keresnie, amig valami megoldást nem talál a problémára. Ha van már elérhető megoldási út, a rendszer ezt a megoldási úton lévő jó lépéseknek az útról letérítő rossz lépésektől való megkülönböztetésére használja. A nemkivánatos következményeket javasló szabályok súlyát csökkenti; mivel a magasabb súlyú szabályok előnyt élveznek az alacsonyabb súlyúakkal szemben, ezért ezek használatára a következőkben kevésbé kerül sor. A rendszernek azonban azokat a heurisztikus feltételeket is meg kell határoznia, melyek alatt egy adott következ-

ményt végre kell hajtani. A Sage megkülönböztetéses folyamat segitségével találja meg egy szabály jó és rossz előfordulásai közti különbségeket. E különbségek extra feltételekként adódnak az eredeti szabályok változataihoz. Egy variáns első létrehozatalakor alacsony súlyt kap, de a súly növekszik, valahányszor a variánst újratanulja a rendszer. Ez az "erősitő" folyamat szolgál a hasznos heurisztikáknak a hamisaktól való megkülönböztetésére, ilyenformán irányitva a keresést a lehetséges szabályok terén át. Egy elég sokszor megtanult variáns súlya meghaladja azét a szabályét, amiből előállt, és ekkor a variáns kezdi irányitani a keresési folyamatot. Az ilyen variánsok is igényelhetnek keresést egy megoldás eléréséhez. Ez esetben a megkülönböztetéses folyamat rekurziv módon kerül alkalmazásra, hogy további feltételekkel rendelkező, még specifikusabb szabályokat állitson elő. Végső fokon a Sage egy olyan heurisztikus halmazhoz ér, ami tökéletesen kiküszöböli a megoldás keresését.

A Sage közeli rokonságban van a Lex-rendszerrel [MIT'81], ami szintén keresési heurisztikákat tanul megoldási utakból. Mig a Lex a változat-tér módszert (version space method) használja fel azon feltételek meghatározására, melyek mellett egy operátort alkalmazni kell, addig a Sage a megkülönböztetéses-erősitési stratégiát használja. A változat-tér megközelités olyan jellemzőket talál, melyek közösek egy szabály minden jó alkalmazásában, a módszer ezért nehézségekbe ütközik, ha egy nemkivánatos következményt véletlenül kivánatkosként rangsorol. Ezzel ellentétben a megkülönböztetéses módszer statisztikát gyűjt minden javasolt szabályváltozatról; következésképpen több előfordulásra van

szükség a megfelelő feltételek meghatározásához, de a hibás besorolásoknak kevesebb hatásuk van a rendszerre.

A Sage természetesen olyan heurisztikákat tanul, melyek a próbával és tévedéssel talált megoldási lépéseket reprodukálják. Így, ha ezek nem optimálisak, akkor a rendszer sem fog optimális keresési stratégiákat találni. Ez a korlátozás azonban a kezdő keresési stratégiában van, és nem a rendszer tanulási mechanizmusaiban.

4.1.2.1 Adat-modularitás

A Sage a modularitás érdekében egyrészt megköveteli, hogy minden egyes szabálya teljesen önálló legyen. Így egy szabály hozzáadása/törlése, vagy egy súlycsere, megváltoztatja ugyan a keresési folyamatot, de a rendszer tovább fut. Hogy a rendszer tanulhasson, a tanulási folyamatban használt adatnak is modulárisnak kell lennie. Miután a Sage további feltételekkel rendelkező új szabályok előállításával tanul, és ezek a feltételek a munkamemóriában lévő elemeken alapulnak, e memória struktúrájával is foglalkoznunk kell.

A Sage által megtanult szabályok az információábrázolásra használt darabok (chunk) méretétől is függenek. Ha ez a darab túl sok információt tartalmaz, akkor a megtanult szabályok szükségtelenül specifikusak lehetnek, és néhány viselkedés megtanulására esetleg egyáltalán nem leszünk képesek. Másrészt, ha az információ túl kis darabokra van tördelve, akkor a helyes feltételek meghatározására irányuló keresés túl nagy lehet.

Egy közismert türelemjátékban, a "Hanoi tornyaiban" azzal a döntéssel, hogy az állapot-átmenetekbe bevont mozgások és objektumok sorrendjére vonatkozó információt elkülönitették, a Sage-nak elég rugalmas ábrázolást adtak. Amikor egy bizonyos mozgás fontos volt, a rendszer megkülönböztető algoritmusa megengedte, hogy a Sage felfedezze ezt a tényt, és új feltételként bevonja. Azonban, ha az információ jelentéktelen volt, nem került a Sage által szerkesztett szabályváltozatok közé. Úgy tűnik, hogy ez az ábrázolási szint a "Hanoi tornyai" feladaton túlnyúlik. A rendszer további öt tárgykörben - más türelemjátékoktól az egyszerű algebráig - sikeresen tanult keresési heurisztikákat, és minden esetben nagyon hasonló felbontást, dekompoziciót alkalmaztak. Így, túl azon a nyilvánvaló következtetésen, hogy az adatmodularitás fontos a tanuláshoz, arra következtethetünk, hogy ez a sajátos ábrázolási szint hasznos a keresési heurisztikák tanulásában.

4.1.2.2 Szemantikusan ekvivalens szabályok

A tanuló rendszerek egyik problémája a szintaktikusan különböző, de szemantikailag ekvivalens szabályok szerkesztése. Két szabály szemantikailag ekvivalens, ha az egyik garantáltan illeszkedik valahányszor a másik illeszkedik, és fordítva.

A depth-first keresést végző tanuló rendszerekben az ilyen ekvivalenciák kevés nehézséget okoznak, mert az elsőként megtalált sikeres szabályt adják vissza. Azonban sok tanuló program - köztük a Sage is - broadth-first keresést végez a szabályok terében. Pl. a Sage, ha

sok különbséget fedez fel valamely szabály pozitiv és negativ esete között, akkor minden egyes különbségre variánsokat szerkeszt. Bármely eléggé sokszor megtanult szabály alkalmazásra kerül, ha illeszkedik, befolyásolva a keresést. Igy az ekvivalens változatokat mindig együtt tanulná meg a rendszer, ezért mindkettőt kiválasztaná - mindig azonos következményt javasló - alkalmazásra. Sajnos mindkét szabály javasolhat túlságosan általános és nemkivánatos következményeket, előidézve ezzel azt, hogy a megkülönböztetéses folyamat mindegyikükről duplikált változatokat készitsen. Ha ezen változatokban lévő új feltételelemek is vannak szemantikus ekvivalensei, kombinatorikus robbanásra kerül sor. Szerencsére ez a robbanás nem fordult elő azon tárgykörökben, melyekben a Sage-t vizsgálták, de az ilyen szükségtelen keresés lehetősége figyelmet érdemel.

E probléma nyilvánvaló megoldása, ha a tanuló rendszert információval látjuk el a szemantikusan ekvivalens szabályok alakjairól. Ekkor a vizsgálatok szerkesztésekor a Sage megjegyezné, hogy két szabálynak mindig ugyanakkor kell illeszkednie és csak egyikűket hozná létre. Azonban a tanulás tanulmányozásának egyik oka éppen az általánosságok elérése, hogy ne kelljen minden tárgykörben, amiben a rendszer fut, újra és újra információt adnunk.

Egy vonzóbb megközelités szerint a Sage az operátorok és az operátorok hatásainak ismeretében következtetne az ekvivalenciára. Igy a rendszer megkisérelhetné "bizonyitani" két szabály ekvivalenciáját azáltal, hogy egy általános következtetőt (reasoning component) tárgykör-specifikus információval egyesit.

A végső megoldás az lenne, ha hagynánk, hogy a Sage tapasztalati úton maga fedezze fel az ilyen ekvivalenciákat. A rendszer pl. nyomon követhetné, hogy különböző változatokat hányszor szerkesztett meg együtt. Ha két szabályt mindig együtt szerkesztett meg, akkor eltávolitaná az egyiket, és ezt az információt későbbi használatra megőrizné. Ha később ugyanilyen két feltételhalmazzal rendelkező más variánsokat hozna létre, csupán az egyik feltételhalmazt tartalmazókat tartaná meg. Ezt a megközelitést nagyon általános módon meg lehetne valósitani, és bármely tárgykörre alkalmazni lehetne. Az emlitett módszerek közül az utóbbi kettőt vizsgálják.

4.1.2.3 Ábrázolás és általánosság - a Bacon rendszer

A kutatók között teljes az egyetértés, hogy az általánosság bármely tanulórendszer esetén fontos kritérium, de kevés javaslatot tettek az általánosság vizsgálatának módjára. Az egyik nyilvánvaló eljárás az, hogy a tanulórendszert számos különböző tárgykörben futtatják. Ha a rendszer minden egyes feladat esetén tud tanulni, és nem igényel további, pontosabban meghatározott komponenseket e feladatokhoz, akkor általános voltára következtetnek. A mesterséges intelligencia kezdeti napjaihoz képest tekintélyes fejlődést értünk el, és ma már sok olyan tanulórendszer létezik, amit sikeresen vizsgáltak ily módon. Azonban e tanulórendszerek közül soknak rejtett szabadsági foka van, mert a programozó bármilyen módon ábrázolhatja a rendszernek adott adatokat. Ezért, bár a tanulási mechanizmusok nagyon általánosan vannak megfogalmazva, gondosan és fortélyosan előkészitett inputot igényelhetnek ahhoz, hogy sikeresek legyenek.

A tapasztalati törvényeket felfedező Bacon rendszer [LBS'83] példája segithet megvilágitani ezt a kérdést. Ha adott a megfigyelések halmaza, akkor a rendszer megállapitja a numerikus változók közti kapcsolatokat, és olyan lényeges tulajdonságokat javasol, mint a tömeg és a fajhő. Végül is valamilyen törvényhez ér, ami összegzi a beadott adatokat. A Bacon-t eredetileg ugyan fizikai törvényekre gondolva tervezték, de általános mivoltát számos törvény újrafelfedezésével megmutatta. Az újrafelfedezett törvények között van az ideális gáztörvény, a Snellius-Descartes-féle fénytörési törvény, és Black fajhő törvénye is. A Bacon általánosságának további vizsgálataként a rendszert a korai kémikusok számára hozzáférhető adatokkal futtatták. A kémia történetének ebben a szakaszában fontos szerepet játszott a közös osztó fogalma, és ezért a rendszert kibővitették egy közös osztó megtalálására szolgáló heurisztikával. Ez a módositás csupán néhány napot vett igénybe, és csak három új szabállyal kellett bővíteni az eredetileg 86 szabályból álló halmazt.

A történelmi adatok megvizsgálásakor azonban egyáltalán nem volt világos a helyénvaló ábrázolás. A kémiai reakciók ugyanis gyakran foglalnak magukban inputként két elemet és outputként egyetlen vegyületet. A Bacon-t viszont olyan törvények felfedezésére tervezték, amelyek csupán az egyik input tulajdonságait hozzák összefüggésbe az outputtal. Hosszú vita után a rendszernek csak egy input elemet adtak meg és természetesen az output vegyületet. A többi input elemet egyszerűen nem vették figyelembe. A viz-reakció esetében pl. input elemként vagy a hidrogént, vagy az oxigént lehetett megadni, de mindkettőt nem. Ez az ábrázolás egészen jól működött, és

a Bacon elérte a korai kémikusok által felfedezett törvények analóg változatait. Mégis, van valami kinos abban, hogy ilyen "kézművességet" kell latba vetni az ábrázoláshoz, és felvetődhet, hogy a Bacon mégsem olyan általános, ha mindent jól a szájába kell rágni.

Nem a Bacon az egyetlen rendszer, amihez gondosan hozzá kell szabni az adatokat. Az input gondos fortéllyal történő előkészitése viszont esetenként meglehetősen erőltetett ábrázolásokhoz vezet. Nem tudunk egyetlen olyan tanulórendszerről sem, melyet ugyanazon adatok többféle ábrázolása esetén kifejezetten vizsgáltak volna. Kevés rendszer végezne jól ilyen teszt esetén, de számos ismert rendszer ilyen teszttel való futtatása hasznos lehetne ahhoz, hogy meghatározzuk az általános tanuló-mechanizmusok fejlesztésében elért haladásunkat.

Az, hogy funkcionálisan ekvivalens szabályokat tanuljunk, amikor adott két szintaktikusan különböző, de funkcionálisan ekvivalens ábrázolás, bizonyára kivánatos. Ugyanakkor ésszerűtlen lenne azt várni, hogy egy tanulórendszer akármilyen ábrázolással foglalkozzon. Az információtárolásra használt darab méretének korábbi tárgyalása azt sugallja, hogy bizonyos ábrázolások egyszerűen nem illenek néhány tanulási feladathoz. Másik megközelités lenne, ha az olyan rendszereknek, mint a Sage és a Bacon megengednénk, hogy bizonyos alaku inputokra támaszkodjanak, de megkövetelnénk, hogy ezt az inputot ne emberi lény, hanem egy másik program állitsa elő. Képzeljünk el pl. egy természetes nyelvű front-end-et, ami elfogadná egy probléma angol nyelvű leirását, és azt legális operátorok halmazává alakitaná át a kezdeti- és cél állitásokkal együtt. Ez a rendszer nemcsak hasz-

nos operátorokat és állitás-leirásokat állitana elő különböző problémákhoz, hanem meg tudná magyarázni pl. a Sage inputjának eredetét, és lényegesen növelné a rendszer általánosságát.

A Bacon hasonlóképpen kölcsönhatásban lehetne más felfedező rendszerekkel, melyek közül néhány javasolná azokat a változókat, amikre a Bacon támaszkodna. A tudománytörténetből tudjuk, hogy a minőségi törvényeket rendszerint a mennyiségiek előtt fedezik fel. Igy a minőségi felfedező folyamat tanulmányozása hasznos lehet annak indokolásához, hogy honnan erednek a Bacon inputjai. Információ azonban a másik irányban is folyhat és a Bacon outputjait - a numerikus törvényeket és a lényeges tulajdonságokat - felhasználhatnák egy integrált felfedező rendszer más részei. Az pl., hogy a Bacon észrevette a közös osztókat oda vezethetne, hogy egy strukturális modelleket kigondoló komponens atom-hipotézist javasoljon a kémiai reakciók szabályszerűségeinek megmagyarázására. Bár egy ilyen integrált felfedező rendszer megszerkesztése több erőfeszitést igényelne, mint ami a Sage természetes nyelvű front-end-del való ellátásához kellene, mindkét esetben ugyanazok az elvek. Egy olyan integrált rendszer létrehozásával, ami külső adatokból állitja elő saját belső ábrázolásait, nagyrészt eltávolithatjuk a szinről a programozót. Eredményképpen rendszereink igazán általánosak lesznek, és nem az input gondos, kézzel történő előkészitésére támaszkodva tanulnak heurisztikákat, vagy fedeznek fel szabályszerűségeket.

4.1.3 A természetes nyelvek megértését segitő tudásábrázolás [HA'84]

A frame-szerű struktúrákkal megvalósitható a tárgyhoz tartozás vagy szempont fogalma. A szemantikus háló bármely adott felhasználásában csak néhány frame aktiv, a többiben tartalmazott információt nem veszik figyelembe. Igy a hálóban látszólag tartalmazott információ a hálót elérő folyamat szempontjától függ. Ez az alaptechnika igen hatékonyan használható számos különböző célra, beleértve a mennyiségi minősités, a különböző szintű részletek, az ellentmondó feltevés, és ugyanazon tárgy különböző aspektusainak ábrázolását.

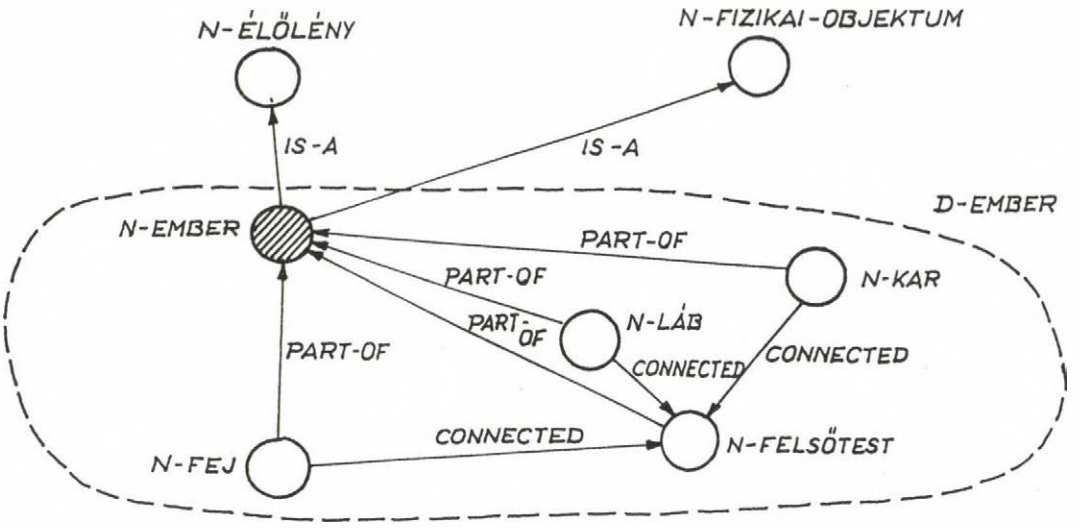
A bemutatott adatábrázolási rendszert kifejezetten olyan természetes nyelvet kezelő rendszerhez tervezték, ami a szövegkörnyezetben lévő kétértelmű szavak megfelelő értelmének megtalálására összpontosit. A rendszer által nyújtott asszociativ struktúra nagyban elősegiti az olyan szavak jelentéseinek megkeresését, melyek jelentése a megelőző és környező szöveggel kapcsolatos. Sok esetben ad hivatkozásokat azon kétséges (vagy kétségtelen) szavakra, amelyek az előző szöveg ábrázolásával kapcsolatosak. A rendszer újszerű módszereket alkalmaz a többszörös részek és a csaknem, de mégsem teljesen azonos objektumok ábrázolására.

4.1.3.1 Az ábrázolás eszközei

A magasabb szintű egységek elnevezése "frame" helyett képmás (depiction), ez szuggesztivebben fejezi ki ezen entitások szerepét. A képmás szógpontok és összekötő részek olyan összessége, amely egy nagy szemantikus háló részhálója. A képmásbeli szógpontok közül egy az un. megfestendő (depictee). A képmások megfestendőiket többi szógpontjuk, az un. lefestők (depicter) segitségével irják le.

Az ember nagyon egyszerű fizikai képmása látható a 13.ábrán. A beszinezett N-EMBER szógpont a megfestendő, mig az üres szógpontok a lefestők. A D-EMBER képmás a zárt szaggatott vonalon belüli szógpontokból, ezen szógpontok közötti összekötő részekből, valamint a bezárt területről kivezető két összekötő részből áll. A szaggatott vonalon kivüli két szógpont nincs a képmásban, noha a megfestendő leirásának részei.

A D-EMBER képmás szógpontjai mind generikus szógpontok abban az értelemben, hogy őstipus testeket, lábakat, törzseket stb. ábrázolnak. Nem lehetnek mind univerzálisan mennyiségileg minősitettek, mert akkor az ábrázolás azt mondaná, hogy minden kar minden testhez kapcsolódik. Ehelyett csak a megfestendő, univerzálisan mennyiséginek minősitett, és a lefestők mindegyike egzisztenciálisan mennyiségileg minősitett, az univerzális mennyiségi minősitő érvényességi tartományán belül.

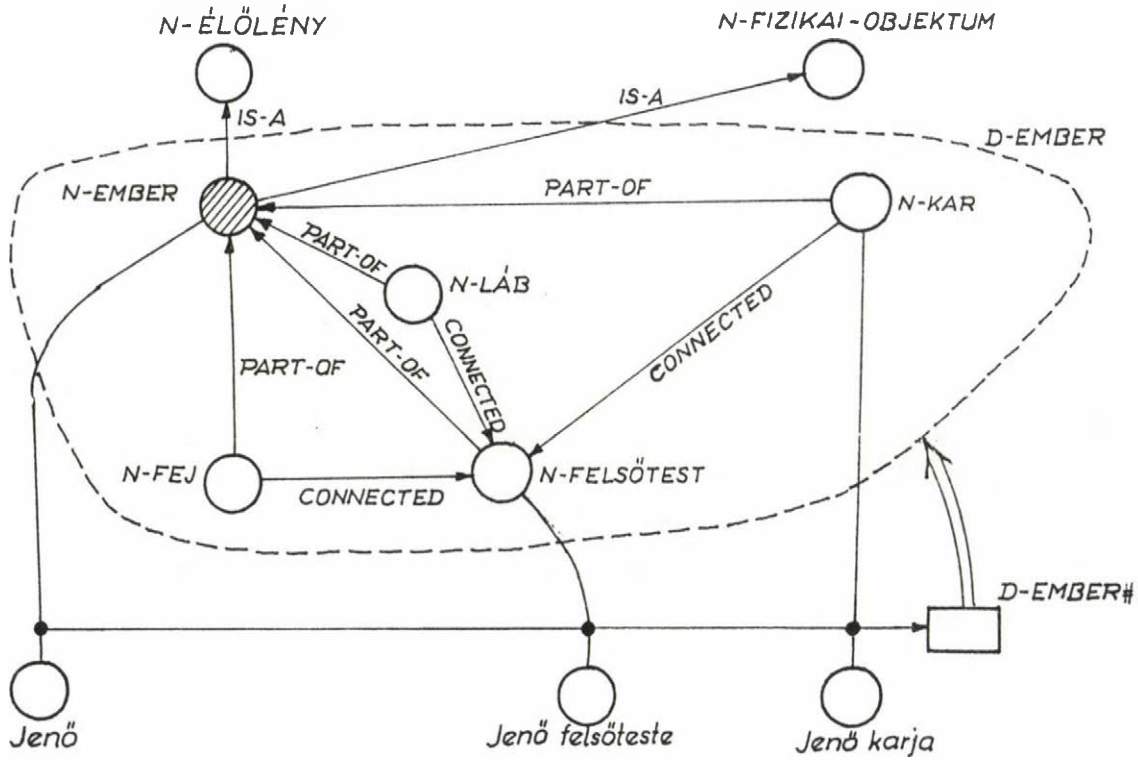


13.ábra. A D-EMBER képmás

Miután a képmások generikusak, példával való szemléltetéseik adják a sajátos tárgyak leírásait. A képmásokat mint egészeket, mindig a lekötők (binder) szemléltetik példákkal. A lekötő egy olyan adatstruktúra, ami többek között tartalmazza annak a képmásnak a nevét, amit példával szemléltet, valamint e képmásbeli generikus szögpontok és a példa szögpontok közötti megfeleltetések listáját. A példa szögpontok inkább specifikus, mint generikus tárgyakat ábrázolnak, és ahogy nevük is mutatja, ezek a specifikus tárgyak azon generikus tárgyak példái, melyekkel lekötőjük párosítja őket.

A lekötők használatát szemlélteti a 14.ábra. A D-EMBER#1

lekötő a négyszögletes doboz, és az ebből a D-EMBER-be mutató kettős nyil jelzi, hogy a D-EMBER lekötője. Jenő, Jenő törzse és Jenő karja mind példa szögpontok, melyek a D-EMBER#1-ben a hozzájuk kapcsolt D-EMBER-beli generikus szögpontokhoz társulnak. A generikus és példa szögpontok összeköttetéseit pontokban metsző, és D-EMBER#1-be mutató vonal e párositások D-EMBER#1-beli jelenlétét jelzi.



14.ábra. A D-EMBER példával való szemléltetése

A példa szógpontok generikus szógpontjaiktól tulajdonságokat örökölnek. Nevezetesen, öröklik generikus szógpontjaiknak más szógpontokkal való összeköttetéseit. Természetesen az örökölt összeköttetések nem más generikus szógpontokkal kötik össze az öröklő példa szógpontot. Így a 14.ábrán Jenő karja a D-EMBER-től nem azt a relációt örökli, hogy a generikus test része (PART-OF), hanem azt, hogy Jenő része. Ha egy képmást egynél többször szemléltetünk példával, a tulajdonság öröklés e módszere megengedi, hogy nyomonkövessük, mely példa szógpontok vannak egymással összekötve. Pl., ha Bélára és Béla karjára szeretnénk létrehozni példa szógpontokat, akkor a D-EMBER új lekötőjével láncolnánk a fenti példa szógpontokat a megfelelő generikus szógpontokhoz. Ez biztosítaná, hogy Béla karja része lesz Bélának, de nem lesz része Jenőnek.

Mivel a lekötők a képmásokat, mint egészeket szemléltetik példákkal, egyetlen adott lekötőben sem kell minden lefestőre példa szógpontot adni. D-EMBER#1 sem szemléltet példával minden D-EMBER-beli szógpontot (14.ábra). Ha azonban N-FEJ példájával bővítenénk a D-EMBER#1-et, akkor az Jenő fejét ábrázolná, és mind Jenő, mind Jenő törzse örökölne vele való kapcsolatot.

4.1.3.2 A CSAW rendszer

A CSAW (Choosing the Senses of Ambiguous Words) olyan működő számítógépes rendszer, melynek feladata szövegkörnyezetben lévő többértelmű/azonos alakú szavak értelmének kiválasztása, természetesen angol nyelvű

szöveg esetén.

Nézzünk egy egyszerű példát:

> Fred walked into the room; his arm was covered in bandages.

Az "arm" szónak sok jelentése van, de ebben a példában egy ember, pontosabban Fred karját jelenti. E mondat ábrázolásának megszerkesztése során a CSAW előállítja a D-EMBER egy lekötőjét Fred ábrázolására. A "his arm" helyes ábrázolása egy olyan példa szögpont, amit ugyanaz a lekötő kapcsol N-ARM-hoz. A CSAW ezen példa szögpont kiválasztásához azt a tényt használja, hogy az "arm" egyik értelmét ábrázoló szögpont olyxan képmásban van, amit éppen most szemléltetett példával. Általánosabb szabály a következő:

(SZ1) Egy szögpont azzal az őt tartalmazó képmással kapcsolatos, amelyik a leírásban van.

A CSAW-ban a szövegkörnyezetet lekötött képmások halmaza definiálja és a szavak értelmezéseit szögpontok ábrázolják. Ezért (SZ1) a szóértelmezések és a szövegkörnyezet közötti kapcsolat definiciójára tett első kisérletnek is tekinthető.

Ha szögpontnak képmással való kapcsolatát használjuk egy nem egyértelmű szó egyértelművé tételére, akkor természetesen a szögpont példával való szemléltetésének kell maga után vonnia azt a lekötőt, ami a képmást a szövegkörnyezetbe hozza. Az (SZ1) esetén a szögpont egyszerűen ugyanazon lekötőben van példával

szemléltetve; ez világosan megadja a "his arm"-hoz a helyes tárgyat a fenti példában.

A szavak értelme és a szövegkörnyezet közti kapcsolat nem adhat mindig egyedülálló tárgyat, sőt néha még helyes értelmezést sem. Vegyük pl. a következő mondatot:

> When I handed the hammer to the man, the (my,his) head fell of.

Mindazonáltal (SZ1) és további finomitásai különösen hasznosnak bizonyultak a CSAW más, egyértelmüvé tevő módszereivel kombinálva.

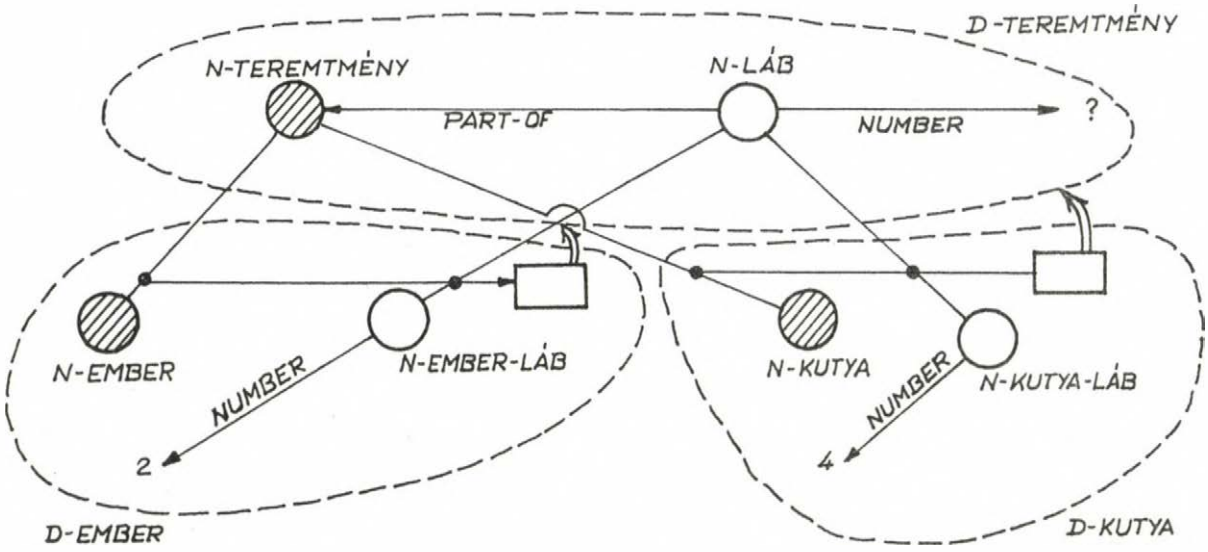
4.1.3.3 Általánositó hierarchiák ábrázolása. Az egyszerepü és a többszerepü szögpontok technikája

Tegyük fel, hogy ábrázolni akarjuk, hány lába van az egyes állattipusoknak. Olyan módszerre van szükségünk, hogy a specializált láb szögpontok örököljék az általánosabb láb szögpontok összes természetesnek tartott kapcsolatát. Ez a probléma része annak, amit Fahlman szimbólumtérképezési problémának nevezett [FA'79].

Kézenfekvő a lekötők ötletéhez folyamodni. Ehhez az állatok lábaira vonatkozó információt egy D-TEREMTMÉNY képmásnak kell tartalmaznia, és a különféle állat alosztályok képmásai a D-TEREMTMÉNY lekötőit tartalmazzák (15.ábra).

A jelölés lényegében az eddigi. A D-TEREMTMÉNY két

lekötője külön-külön a D-EMBER-ben és a D-KUTYA-ban van.



15. ábra. Képmások közötti információöröklés - egyszerepű szögpontok technikája (single-role node technique).

Ezeket eltérő szerepük miatt belső lekötőknek (internal binder) nevezzük. Az N-EMBER és az N-EMBER-LAB között nincs PART-OF kapcsolat. Ez és minden más N-TEREMTMÉNY és N-LAB közti összefüggés pontosan ugyanolyan

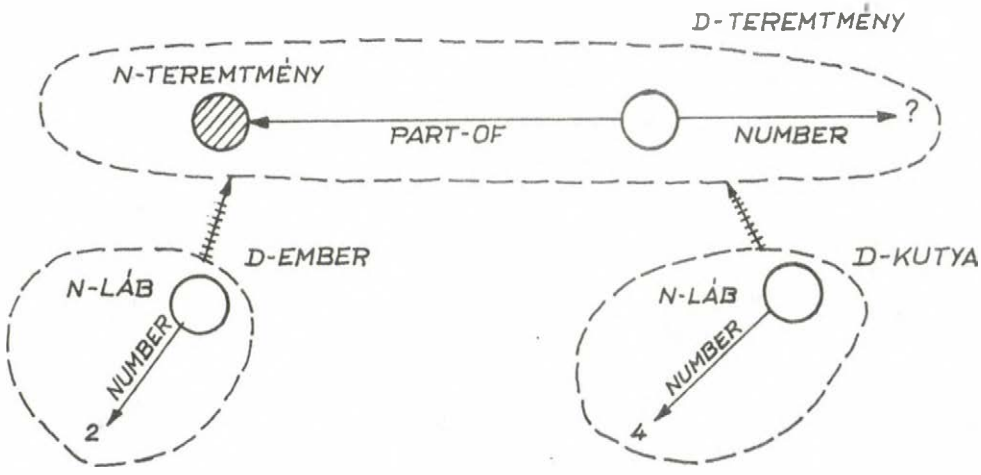
mechanizmus szerint öröklődik, mint amit a példa szögpontokra láttunk.

Számos különböző, de hasonló tárgy képmása sokkal hatékonyabban osztozhat a közös információn, ha a képmásokat szempontként használjuk. Ha az, hogy egy szögponthoz kapcsolódó összekötő részek közül melyek láthatók attól függ, hogy a szögpontot mely képmásból látjuk, akkor ugyanazon szögponttal számos különböző entitás ábrázolható. Így lehetővé válik, hogy egyetlen rögzitett szögpontot használjunk az ember, a kutya, vagy bármely más teremtmény, illetve ismét másikat ezek lábának ábrázolására.

A szögpontok e többszörös használata a CSAW-ban a képmások általánositó hierarchiájának segitségével valósul meg. Ebben a hierarchiában az utód képmás automatikusan örökli a szülő összes összekötő részét és szögpontját. Ha valamely összekötő rész vagy szögpont alkalmatlan az utód képmás számára, akkor ezt specifikusan ki lehet, és ki is kell zárni. Az utód képmásnak ezen kivül lehetnek saját szögpontjai és összekötő részei, melyek a szülő képmásából még akkor sem láthatók, ha azok a a szülő képmásbeli szögpontokat magukba foglaló összekötő részek.

Hogy világosabbá tegyük ezt a finomitást, nézzük meg, hogyan alakulna eszerint a 15.ábránk (16.ábra). Az áthúzott vonalak mutatják, hogy D-TEREMTMÉNY a D-EMBER és a D-KUTYA általánositója. A D-TEREMTMÉNY-beli PART-OF összekötő rész nem szerepel a D-EMBER-ben és a D-KUTYA-ban, mert automatikusan öröklődik. Ez a helyzet N-TEREMTMÉNY-nyel is. N-LÁB szintén automatikusan

öröklődik, és csupán azért irtuk le ismételten mind D-EMBER-ben, mind D-KUTYA-ban, mert mindkét képmásban új összekötő részeket von maga után. A három képmásban is megjelenő N-LAB szögpont valójában azonos, csak tipográfiailag egyszerűbb ez a megoldás. Az egyetlen dolog, amit az ábra hibásan mutat az, hogy a D-TEREMTMÉNY-beli NUMBER összekötő rész nem jelenik meg D-EMBER-ben vagy D-KUTYA-ban, mert egy különleges direktiva szolgál az egyes képmásokhoz társitott hatásra.



16.ábra. Öröklés a képmások hierarchiájában - többszerepű szögpontok technikája (multi-role node technique)

A két technika ábrázolásai nyilvánvaló módon egymásba alakithatók, és ilyen értelemben a két megközelités ekvivalens. A többszerepü megközelités néhány elönye:

- Kevesebb a helyigénye, mert tekintélyesen csökken a szükséges szögpontok száma és nincs szükség a szögpontok közti párositások tárolására.

- Gyorsabban eldönthetö, vajon egy adott reláció összekapcsol-e két szögpontot egy adott képmásban, vagy sem.

- A szövegfeldolgozásra kihat, hogy pl. a "láb" jelentése, mint egy teremtmény lába, pontosan egy szögpontnak felel meg, és nem annyinak, ahány teremtménytipust ábrázolunk.

A sokszerepü megközelitésben kis bonyodalmat okoz, hogy a jelentéseket egy képmásnak és egy szögpontnak kell ábrázolnia. A képmásra azért van szükség, mert nélküle nem tudnánk megmondani, hogy milyen összekötö részek vannak a szögponthoz csatolva, és ezért nem tudnánk, hogy a szögpont mit ábrázol. A szögpontnak és annak a képmásnak a kombinációja, amiböl a szögpontot nézzük: a VNODE (Viewed NODE); és pl. a D-KUTYA-ból nézett N-TEREMTMÉNY VNODE-ot jelölje

$<N-TEREMTMÉNY, D-KUTYA.

Kapcsolat keresésekor, miután a VNODE szögpont részét a szövegkörnyezet lekötöjének képmásában megtaláltuk, ellenörizni kell, hogy ez a képmás kompatibilis-e a VNODE képmásával. Akkor kompatibilisek, ha az egyik

közvetlen vagy közvetett általánositója a másiknak. Tekintsük a következő példát:

> Fred was in the room. His leg was covered in bandages.

Az első mondat feldolgozása után a szövegkörnyezet a D-EMBER lekötőjét tartalmazza az N-TEREMTMÉNY példa szögpontjával, ami Fred reprezentálására szolgál. A "leg" megfelelő jelentésére a szótári bejegyzés: $<N-LAB,D-TEREMTMÉNY. Miután N-LAB D-EMBER-ben van, és D-TEREMTMÉNY a D-EMBER általánositója, kapcsolat van a kérdéses VNODE és a D-EMBER között. Ezért a "leg" ábrázolása az N-LAB példával való szemléltetése lesz a D-EMBER-nek abban a lekötőjében, amit az első mondathoz szerkesztettünk. Forditott irányú általánositó kapcsolat adódik a következő mondat ábrázolásakor:

> I called the dog; the poodle refused to come.

Ez a példa kissé erőltetettnek tűnik, és valóban, azok a példák, amelyekben az általánosabb képmás követi a kevésbé általánost, sokkal hétköznapibbak, mint forditva.

(SZ1) VNODE-ra való kiterjesztése a következőképpen összegezhető:

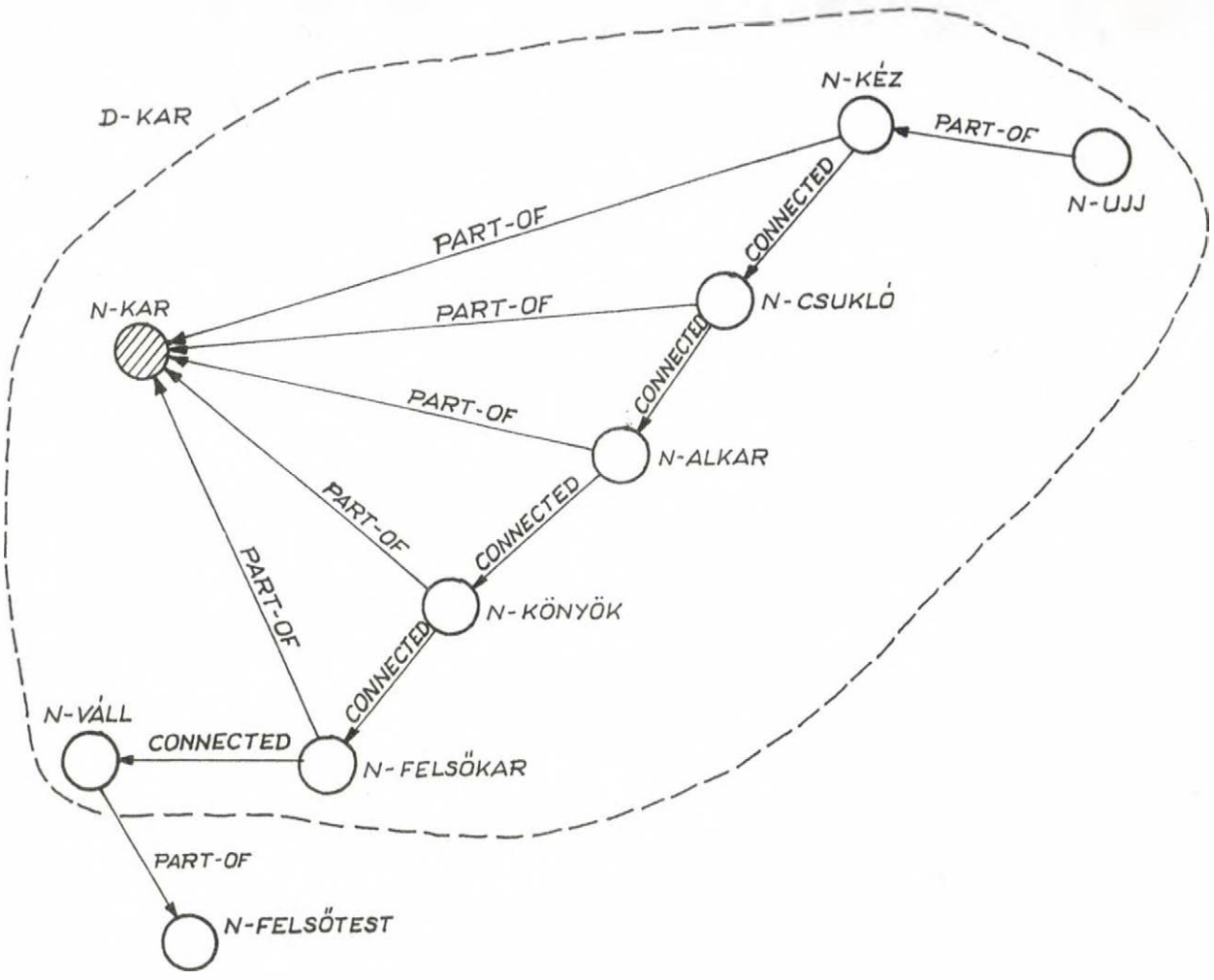
(SZ2) Egy VNODE kapcsolatban van egy képmással, ha szögpontja a képmásban van, valamint képmása, és az adott képmás közül az egyik a másik általánositója. Ha a VNODE képmása az általánosabb, akkor a szögpontot az adott képmás lekötőjében kell

példával szemléltetni. Ha az adott képmás az általánosabb, akkor lekötőjét a VNODE képmásának lekötőjévé kell alakitani, és ebben kell a szógpontot példával szemléltetni.

Ez az eljárás lényegesen kevesebb munkával jár, mert azt ellenőrizve, hogy a VNODE szógpontja része-e a szövegkörnyezet leirásnak, egyben "kiszűrjük" azt is, hogy mely általánositó hierarchiákon kell keresni. Az egyszerepű megközelitésben viszont potenciálisan, belső összekötők láncán keresztül, bármely képmás bármely szógpontja kapcsolatban lehet egy adott szógponttal. Ezért egyszerepű megközelités esetén az összes belső összekötőkkel összekapcsolt szógpontot meg kell vizsgálni.

4.1.3.4 Rész-egész relációk ábrázolása

Az érthetőség kedvéért a jelentések, átmenetileg, ismét egyszerű, egyszerepű szógpontok lesznek. A képmásokat továbbra is szempontként használjuk. Az egyszerű szógpontok most egyszerű entitásokra vonatkozó információ különböző részeit teszik függetlenül elérhetőkké (17.ábra). N-KAR a D-KAR megfestendője, mig D-EMBER-ben lefestő volt. A 17.ábra nem mutatja a 13.ábra összekötő részeit, mert ezek nem láthatók, amikor N-KAR-t D-KAR-ból nézzük. A 17.ábra összekötő részei hasonlóképpen láthatatlanok a 13.ábrán.



17.ábra. A D-KAR képmás

Nem célszerű az N-KAR-ra vonatkozó összes információt D-EMBER-be tenni, mert akkor nem tudnánk kialakitani a részletek szintjeit, és problémát jelentene a többszörös részek ábrázolása. Ha az információ a "természetes" testrészeknek megfelelően választott képmások gyűjteményében oszlik el, akkor csak az adott szövegkörnyezethez

szigorúan hozzátartozó információt kell aktiválni. Ha D-KAR külön képmás, akkor N-KAR kettős szerepet játszhat: egy általános kar ábrázolása mellett a test összes karjának generikus halmazát is ábrázolhatja. Így a szógpont vagy egy egyszerű képmásban lévő, hasonló entitások halmazát, vagy ezen halmaz tipikus elemét ábrázolja aszerint, hogy melyik összekötő részben használják. A 13.ábrán az N-KAR-t más szógponttal összekötő részek a tipikus elem szerepében használják N-KAR-t. A D-EMBER-t ki kellene egészíteni (NUMBER N-KAR 2) összekötő résszel is. Ez az összekötő rész az N-KAR által ábrázolt halmaz méretét mutatja, és így korlátozza az N-KAR-ral ugyanazon lekötőben párosítható példa szógpontok számát. NUMBER összekötő rész nélküli generikus szógpont egy adott lekötőben csak egyszer szemléltethető példával (a megfestendő mindig ilyen).

Ha külön generikus szógpontot használnánk a többszörös rész minden egyes generikus előfordulására, akkor

- a sok előfordulás pazarló ismétlést vonna maga után, pl. egy százlábú esetén;

- a testrészt nem egy egyszerű szógpont ábrázolná, így a karra vonatkozó "arm" szótári bejegyzésnek a két generikus kar szógpontot az "arm" két különböző jelentéseként kellene kezelnie. Ezt a problémát megoldaná a test karjainak halmazát ábrázoló további generikus szógpont bevezetése;

- "egy kar" ábrázolásához a bal és a jobb kar között tetszőlegesen és szükségtelenül választani kellene;

- változó számú rész esetén a maximális számot kellene ábrázolni. Ha ez a szám nagy, akkor az első problémával állunk szemben, de még ennél is rosszabb, amikor nincs felső határ, mint pl. az asztal lábai esetében.

Néha szükség van a többszörös részek megkülönböztetésére, mert különböző generikus információ vonatkozhat rájuk (pl. a bal és jobb kar valóban különböző). Ez a megkülönböztetők (distinguisher) használatával valósítható meg. Olyan cimkehalmazról van szó, ami lehetővé teszi, hogy minden egyes generikus rész egyedülállóan azonosítható legyen egy megkülönböztetővel vagy megkülönböztető gyűjteménnyel. N-KAR esetében éppen két ilyen cimke van: L-BAL és L-JOBB, a bal és a jobb megfelelőjeként. Az L-BAL-lal megkülönböztetett N-KAR jelölése irásban: *!N-KAR(L-BAL). Egyetlen részt vagy a részek részhalmazát magába foglaló generikus információ kezeléséhez olyan összekötő részeket engednek meg, melyek csak akkor vonatkoznak a szögpontra, ha az megvan cimkézve ezen megkülönböztetők egyikével. Egy példa szögpont sajátságossá válik, ha a megfelelő lekötőben a generikus szögpontjával való párositáshoz alkalmas megkülönböztetőt csatolunk.

Ha egy képmás megfestendője más entitás része, vagy valamilyen egyéb módon más entitás létezésére utal, akkor ez a képmás lényegében nem teljes. Teljessé tételéhez szükség van annak a másik entitásnak a képmására, melyben saját megfestendője feltehetően lefestő. Igy D-KAR-nak nincs teljes értelme D-EMBER nélkül. Azt mondjuk, hogy D-EMBER a D-KAR SQN képmása (Sine Qua Non = elengedhetetlen feltétel).

SQN strukturának nevezzük a képmások olyan fáját, melyben a szülő képmások gyermekeik SQN képmásai. Az ugyanazon bonyolult entitást ábrázoló képmásokat rendszerint egy SQN struktúrába szervezik. Nyilvánvalóan nincs minden képmásnak SQN hierarchiabeli felettese. Ezek a képmások lesznek az SQN hierarchiák gyökerei. Igy D-EMBER lesz D-KAR, D-FEJ, D-LAB stb. szülője, és ugyanakkor D-KAR lesz D-UJJ szülője.

Az SQN és az általánositó struktúrák merőlegesek egymásra abban az értelemben, hogy az SQN struktúra független az általánositó struktúrától, és az általánositó struktúra kiegészitője. A két struktúra-tipus annyiban hasonlit egymásra, hogy mindkettő számos olyan különálló fát hoz létre, melyek sok, egymással kapcsolatban lévő képmást csoportositanak.

Valahányszor egy képmást példával szemléltetünk, minden leirásnak, mely egy SQN hierarchiában e képmás szülő leirása, megfelelő példával való szemléltetése kell legyen. A két leirásban közös szögpontok közül egy vagy több SQNNODE-ként jelölhető meg; ezeket a szögpontokat ugyanazon példa szögpontok szemléltetik mindkét lekötőben. Igy D-KAR példával való szemléltetéséhez szükség van a D-EMBER példával való szemléltetésére; ezen kivül N-VALL-nak és N-KAR-nak - melyek a D-KAR SQNNODE-jai - mindkét lekötőben ugyanazon példa szögpontokkal párositottaknak kell lenniük.

Az SQN struktúrák létezése megköveteli az (SZ1) szabály felülvizsgálatát.

(SZ3) Egy szögpont egy adott képmással kapcsolatban van,

ha vagy az adott képmásban van, vagy egy olyan képmásban, melynek az adott képmás közvetlen vagy közvetett őse egy SQN hierarchiában.

(SZ3) kétségtelen kapcsolatot ad pl. a

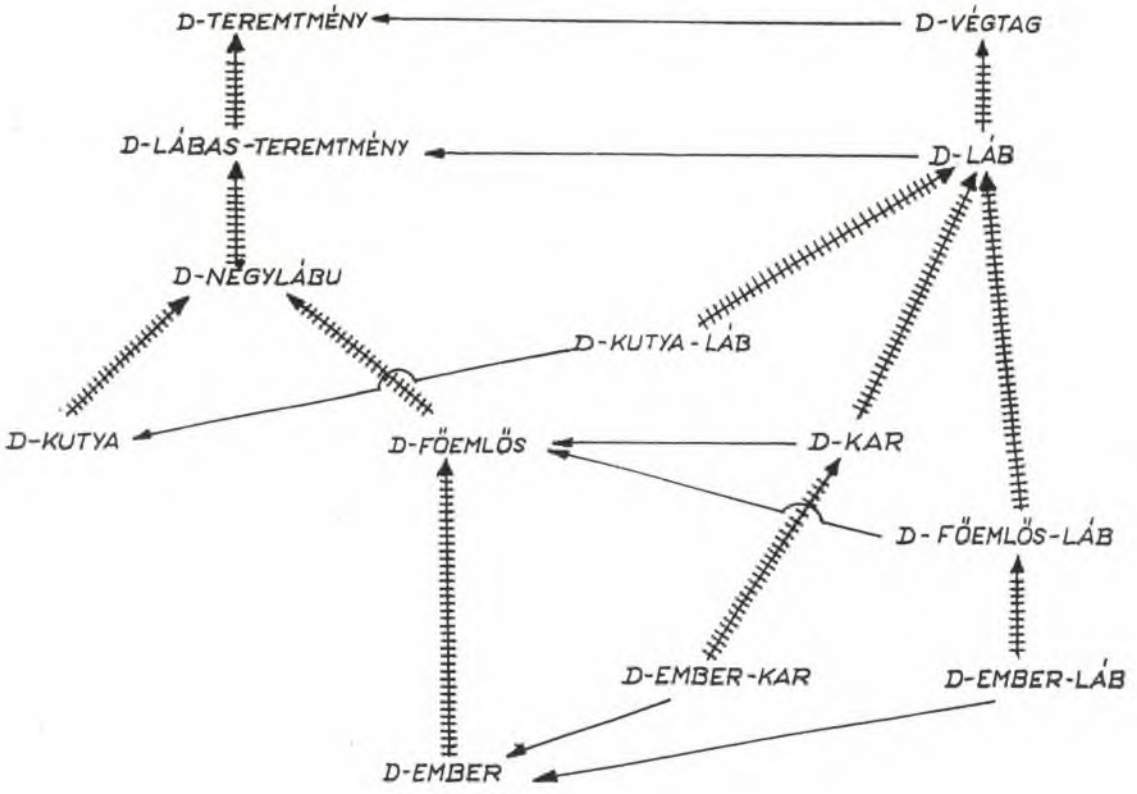
Fred walked into the room; his nails were cracked and dirty.

mondatban "nails"-re. Miután egy képmás lekötésekor az SQN hierarchiabeli szülő képmások megfelelő lekötőit is létre kell hoznunk, (SZ3)-nak nem kell kitérnie az SQN hierarchiában lévő utód vagy testvér képmásokra.

4.1.3.5 Képmáslàncok

Ha általánositó hierarchiabeli entitások az SQN hierarchiák gyökerei, akkor ezek SQN utódai hasonlóak, igy külön általánositó hierarchiába csoportosithatók, ami - megközelitőleg - párhuzamos a felső szintű általánositó hierarchiával (18.ábra). Az ábra két általánositó hierarchiát ábrázol: egyet a felső szintű entitásokra, a teremtményekre, és egyet a teremtmények végtagjaira. A D-NÉGYLABÚ képmás kivételével minden teremtmény képmás egy SQN hierarchia (⟶) gyökere. A legtöbb esetben csak egyetlen utód szógpont van ezekben a hierarchiákban, mely a kérdéses teremtmény egy tipikus lábát ábrázolja. D-NÉGYLABÚ-nak azért nincs egyetlen SQN utóda sem, mert a lábbal rendelkező teretmény tipikus lábát leiró D-LÁB teljesen megfelel a D-NÉGYLABÚ céljainak. Ugyanakkor mind D-FŐEMLŐS, mind D-EMBER két-két SQN utóddal rendelkezik: az egyik mellső

végtagjaikat vagy karjaikat, a másik hátsó végtagjaikat vagy lábaikat írja le. Az hogy D-KUTYA-nak csupán egyetlen SQN utóda van, arra utal, hogy ebben az ábrázolásban hiányzik a kutya mellső és hátsó lábait megkülönböztető információ.



18.ábra. Az általánosító és SQN hierarchiák

Az SQN utódokban való eltérések arra utalnak, hogy a végtagok általánosító hierarchiája nem teljesen párhuzamos a teremtmények általánosító hierarchiájával.

Képmáslánc nak (depiction chain) nevezik egy vagy több

képmás olyan sorozatát, melyben az egymásra következő képmásokat a rokonság-tipusnak (kin-type) nevezett utak egyike kapcsolja össze. Eddig négy rokonság-tipust ismertünk meg:

SQN	a második képmás az elsőnek SQN szülője;
SQN-1	az első képmás a másodiknak az SQN szülője;
GEN	a második képmás az elsőnek közvetlen vagy közvetett általánositó őse;
GEN-1	az első képmás a másodiknak közvetlen vagy közvetett általánositó őse.

Pl. a 18.ábrán D-KUTYA és D-LAB a

(D-KUTYA D-LABAS-TEREMTMÉNY D-LAB)

képmásláncca1 van összekapcsolva, melyben a rokonság-tipusok GEN és SQN-1. A kapcsolatra vonatkozó szabály új változata pedig a következő:

(SZ4) Egy VNODE kapcsolatban van egy képmással, ha van érvényes képmáslánc az adott képmás és a VNODE képmása között.

Egy pillanatra feltételezve, hogy bármely képmáslánc érvényes, könnyen látható, hogy a fenti szabály (SZ2) és (SZ3) összesitése.

A példával való szemléltetés megfelelő szabálya a következő:

(SZ5) Adott egy képmáslánc̈ban két egymást követő képmás, és az első képmás lekötője. A második képmáshoz alkalmasan kapcsolt lekötő létrehozása a rokonság-típus szerint történik:

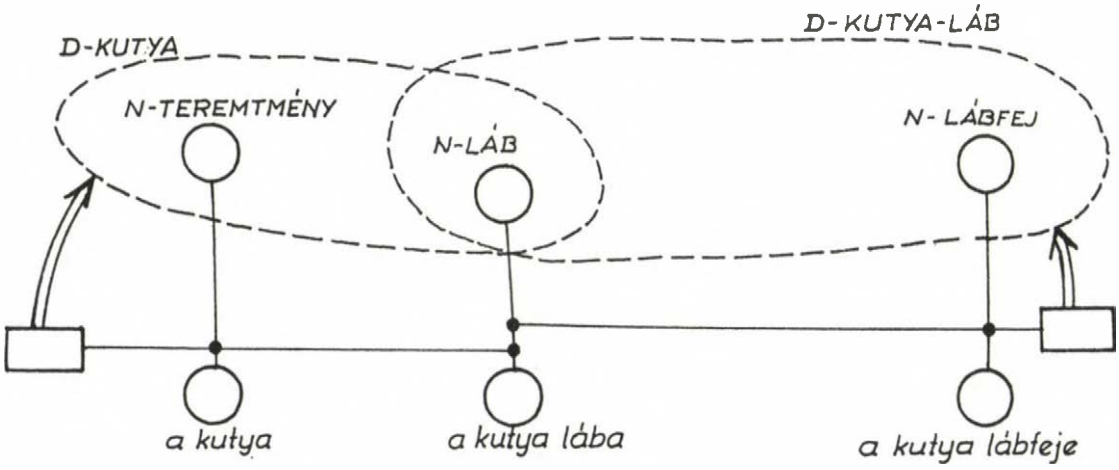
GEN	az adott lekötőt használjuk;
GEN-1	az adott lekötőt használjuk, miután a második képmás lekötőjévé alakítottuk;
SQN	a második képmásra olyan új lekötőt hozunk létre, melyben az első képmás SQNNODE-jainak példa szögpontjai ugyanazok, mint az adott lekötőben;
SQN-1	a második képmásra olyan új lekötőt hozunk létre, melyben a második képmás SQNNODE-jainak példa szögpontjai ugyanazok, mint az adott lekötőben.

(SZ5) egymás utáni alkalmazásai példával szemléltetnek bármely képmáslánc̈ot, melyben az első elemnek van lekötője. Egy VNODE és egy szövegkörnyezet-képmás közötti kapcsolat példával való szemléltetése (SZ) szerint a képmáslánc̈ példával való szemléltetéséből - kiindulásul a szövegkörnyezet-képmás lekötőjét használva -, majd a befejezésképpen előállított lekötőben a VNODE szögpontjának példával való szemléltetéséből áll. Hogy gyakorlatban is lássuk e szabály működését, nézzük a következő példát:

I looked at the dog; its foot was swollen.

Ebben szükség van a "foot"-ra vonatkozó $<N-LABFEJ,D-LAB

VNODE és a D-KUTYA képmás közötti kapcsolatra. (SZ4) szerint képmásláncot kell találnunk D-KUTYA-tól D-LAB-ig. A 18.ábrán nyilvánvalóan ilyen (D-KUTYA D-KUTYA-LAB D-LAB), melyben a rokonság-tipusok SQN-1 és GEN. E képmáslánc példával való szemléltetése (SZ5) alapján először lekötőt állitana elő D-KUTYA-LAB-ra, melyben a D-KUTYA-LAB SQNNODE-jainak - legyen ez csak N-LAB - ugyanazok a példaszögpontjai, mint a D-KUTYA adott lekötőjében. A második lépésben a D-KUTYA-LAB most szerkesztett lekötőjét változatlanul használná fel D-LAB lekötőjeként. Végül N-LABFEJ, a VNODE szögpontja, ugyanezen lekötőjében lenne példával szemléltetve, előállitva a 19.ábrát. és éppen erre van szükség.



19.ábra. A már ábrázolt kutya lábfejének ábrázolása

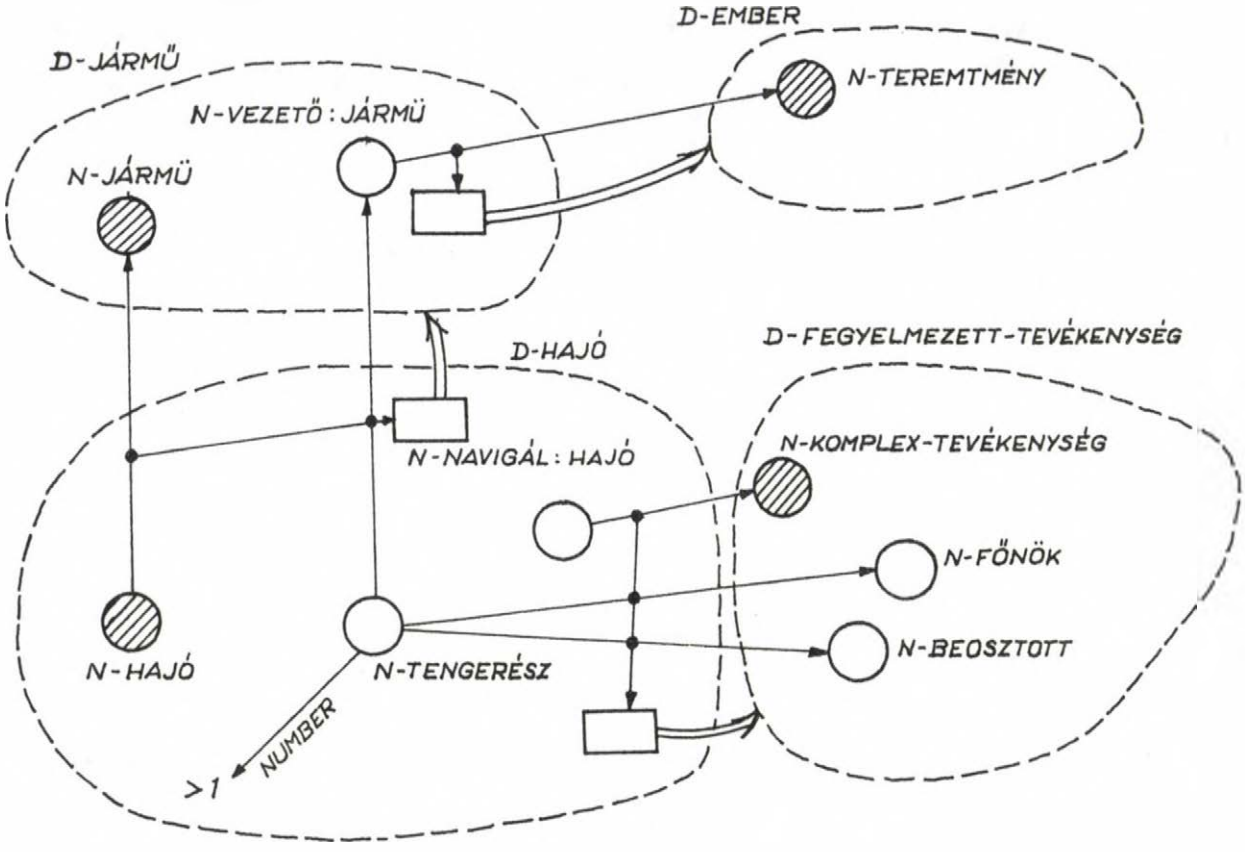
Természetesen vannak más, D-KUTYA-ból D-LAB-ba vezető képmásláncok is, pl. (D-KUTYA D-NÉGYLÁBÚ D-EMBER D-

EMBER-LAB D-LAB), melyek példával való szemléltetése kevésbé kivánatos eredményt adna. Az ilyen láncok kizárhatók, ha megtiltjuk, hogy a képmáslláncokban GEN és GEN-1 illetve SQN és SQN-1 mindkettője előforduljon. Körmönfontabb hibatipust okozna a következő képmáslánc: (D-KUTYA D-LABAS-TEREMTMÉNY D-LAB). Ez éppen a 19.ábrával egyező diagramot eredményezne, kivéve hogy a specifikusabb és alkalmasabb D-KUTYA-LAB helyett D-LAB-at használná. Ez elkerülhető, ha ragaszkodunk ahhoz, hogy az SQN és SQN-1 rokonság-tipusok előzzék meg a GEN és GEN-1 rokonság-tipusokat, ha lehetséges. Ha megpróbálunk kapcsolatot találni D-NÉGYLABÚ és D-LAB között, akkor láthatjuk, hogy az előbbi feltétel nem mindig teljesithető, de ha nem lehetséges, akkor nem is eredményez hibát.

4.1.3.6 A belső lekötők csoportositó funkciói

Általánositó struktúrával nem fejezhető ki minden halmazba tartozás. A belső lekötők viszont a képmás megfestendőjére, sőt a lefestőkre és a lefestők közötti relációkra vonatkozó halmazba sorolásokat is ábrázolhatják. A 20.ábrán az egyik D-HAJÓ-beli belső lekötő mutatja, hogy a hajó jármű, és egyben olyan vonatkozásba hozza a tengerészeket a hajóval, mint amilyen a járművezetők és a járművek között van általában. A másik D-HAJÓ-beli belső lekötő mutatja, hogy a hajónavigálás mint tevékenység a fegyelmezett tevékenység tengerészekre vonatkozó formája. A 20.ábra N-FŐNÖK és N-BEOSZTOTT mindkettőjét az N-TENGERÉSZ-szel párositja; valójában ezek megkülönböztetett párositások, melyek rendre tisztet és beosztott tengerészt mutatnak.

A D-JARMU belsô kötôje mutatja, hogy N-VEZETÔ:JARMU ember.



20.ábra. A belsô lekötôk csoportosító funkciói

Az ilyen belsô lekötôk fontos kapcsolatokat adhatnak, ahogy ez a következô példa kapcsán is kitûnik:

> When I visited the ship, the hands were out on deck.

Itt "hands" a hajó legénységét jelenti. Ha a "hand" földmûvesként (farm hand), matrózként (deck hand), gyári munkásként (factory hand) stb. való értelmezéseinek ábrázolására $<N-BEOSZTOTT,D-FEGYELMEZETT-TEVéKENYSéG-et használjuk, akkor az N-HAJÓ és az N-KOMPLEX-TEVéKENYSéG közti kapcsolat éppen helyes kapcsolatot ad a fenti mondat pontos egyértelmûvé tételéhez. A belsô kötôk ezért érvényes BIND rokonság-tipusként használhatók (SZ4) képmásláncaiban. A fenti kapcsolat kifejezhetô a (D-HAJÓ D-FEGYELMEZETT-TEVéKENYSéG) képmásláncal, melyben a rokonság-tipus BIND. (SZ5) majdnem nyilvánvaló kiegészitése:

(SZ5') BIND új lekötôt szerkesztünk a második képmásra, melyben a belsô lekötô által párositott szögpontoknak ugyanazok a példa szögpontjaik, mint elsô képmásbeli párjuknak.

A BIND rokonság-tipus használatát is némileg korlátozni kell. Ha egy képmáslánc olyan BIND rokonság-tipust tartalmaz, ami csupán egyetlen párositásból álló belsô lekötônek felel meg, a lánc érvénytelen, hacsak a párositott szögpontok példa szögpontja nem olyasmit ábrázol, ami kifejezetten említésre került a szövegben.

4.2 Tudásábrázolást támogató nyelvek

A tudásábrázolás támogatására szánt nyelveket hagyományosan az eljárásos vagy a deklarációs nyelvek osztályába sorolják. Az eljárásos ábrázoló nyelvek különösen jól illenek a heurisztikus tudáshoz, és szakértô rendszerben való használatuk hatékony

keresséshez vezethet. A production rendszerek által ajánlott eljárásos ábrázoló nyelvek általános vázát sikeresen használták számos szakértő rendszer tervezésében. A deklarációs ábrázoló nyelvek, mint a Prolog és a KL-One, elősegitik az áttekinthetőséget és támogatják a tudásbázis karbantarthatóságát.

[MIN'75] óta sokszor kisérelték meg integrálni a deklarációs és eljárásos ábrázoló nyelvek jellemzőit, és ez az irányzat rendületlenül megmaradt a folyó kutatásokban is (lásd PSN).

4.2.1 KL-One [WO'83]

A KL-One célja nem egy részletes számitógépes rendszer előállitása, hanem inkább az, hogy a tudásszervezés és -szerkezet általános elveinek felfedezésére és világos kimondására kényszeritsen. A kifejező erő fontos vezérlő elv a KL-One kutatásban. A kifejező erő hangsúlyozza az ábrázolás szemantikájának fontosságát, és a kifejező erőtől függ, hogy az ábrázolás képes-e olyan hajszálnyi megkülönböztetésekre, mint a bonyolult eszméket felfogó ember. Ezért a KL-One-t érezhetőbben hatja át az alkalmazott filozófiai vizsgálatok szelleme, mint bármely más tudásábrázolással és adatstruktúrákkal foglalkozó munkát.

4.2.1.1 A KL-One fogalmak struktúrája

A helyzetek belső leirásának felépitéséhez szükség van a tárgyak, anyagok, idők, helyek, események, feltételek, állitások, funkciók, egyedek stb. fogalmainak használatára. Az egyes fogalmak olyan attributumok vagy részek konfigurációiként jellemezhetők, melyek kielégitenek bizonyos megszoritásokat és egymással meghatározott kapcsolatban állnak. Ez a jellemzés a KL-One tudásábrázoló rendszer alapvető eleme.

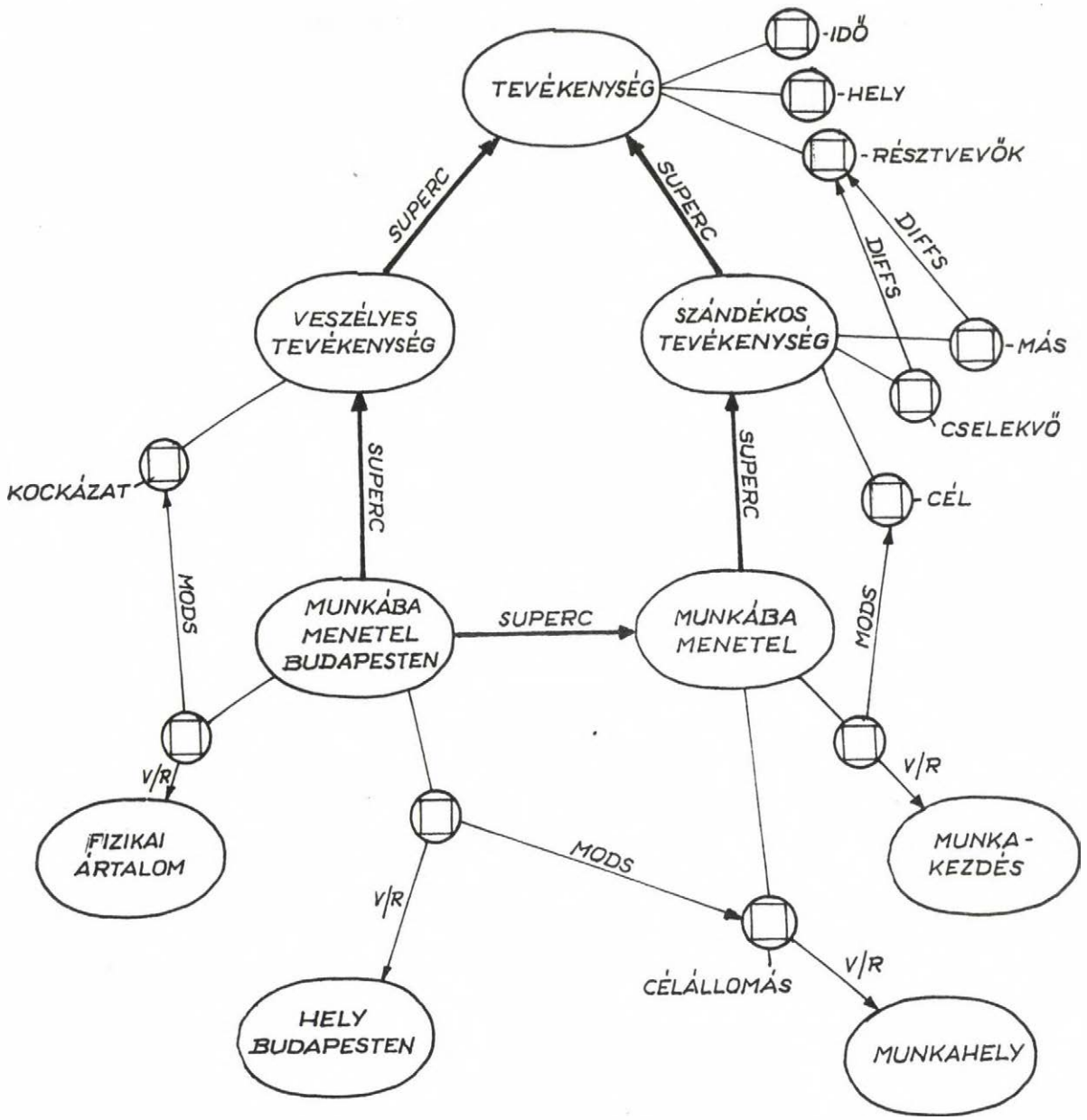
A KL-One-ban egy fogalom-szögpont szerephalmazból - az attributum, rész, alkotóelem, jellegzetesség stb. fogalmainak általánositásaiból - és e szerepek közti kapcsolatokat kifejező strukturális feltételek halmazából áll. A fogalmak un. SUPERC relációval kapcsolódnak az általánosabb fogalmakhoz. Egy ilyen kapcsolatban a legáltalánosabb fogalom a szuperfogalom (superconcept), ami magába foglalja a specifikusabb fogalmakat (subconcept). Egy fogalom szerepei és strukturális feltételei közül néhány közvetlenül a fogalomhoz kapcsolódik, mig a többit közvetve örökli az általánosabb fogalmaktól.

A KL-One fogalmai és szerepei strukturájukat tekintve a rekord és mező általános adatszerkezeti fogalmaihoz, vagy a mesterséges intelligencia terminológiájában szokásos "frame"-ekhez és "slot"-okhoz hasonlóak. Természetesen számos különbség is van, mint pl. a fogalmakhoz csatolt strukturális feltételek jelenléte, a különböző általánossági szintű szerepek közti kifejezett kapcsolatok és az az általános szándék, hogy a KL-One fogalmak ne csupán egy számitógépes megvalósitás

adatstruktúráinak szerepét játsszák, hanem modellezzék a fogalmak absztrakt világának szemantikus és elvi struktúráját.

A KL-One rendszertani szerkezetét a 21.ábra mutatja, ahol a fogalmakat ellipszisek és a szerepeket bekarikázott négyzetek ábrázolják. Az ábra tetején helyezkedik el a Tevékenység magas szintű fogalma. Ennek szerepei az Idő, a Hely és a Résztvevők, melyeket minden alatta lévő fogalom örököl. A Tevékenység alatt - jobbra - van a Szándékos Tevékenység fogalma, ami a Résztvevők általános szerepében megkülönbözteti (DIFFS) a Cselekvőt, azaz azt a résztvevőt, akinek szándéka a tevékenység Más Résztvevőktől. A Szándékos Tevékenység új szerepet vezet be, a Célt, hogy ábrázolni tudja a tevékenység szándékát. A Szándékos Tevékenység alatt található a Munkába Menetel eléggé specifikus, de még mindig általános fogalma. Ez a fogalom módositja (MODS) a Szándékos Tevékenység Célját azáltal, hogy a Munkakezdéssel mint értékmegszoritással (V/R) kibővíti, mutatva, hogy bármi tölti meg a Célt, annak a munkakezdés előfordulásának kell lennie. Egy új szerepet is bevezet, a Célállomást, Munkahely értékmegszoritással. A fogalomhoz kapcsolt nem látható strukturális feltétel előirná, hogy miként függ össze a Munkahely a Munkakezdés céllal - azaz, hogy ez a Munkakezdés célállomása. A Munkába Menetel Budapesten a Munkába Menetel olyan specializálása, melynek Célállomása egy Budapesten lévő Helyre korlátozott. A Munkába Menetel Budapesten ugyanakkor specializálása a veszélyes Tevékenységnek is, melynek Kockázata Fizikai Ártalom.

Elvárnánk, hogy egy intelligens számitógépes rendszerben olyan rendszertan legyen, mint amilyet a 21.ábra mutat,



21.ábra. Példa KL-One hálózatra

beleértve a nagyon magas szintű absztrakciókat és az egészen specifikus fogalmakat is. Ezen felül mindig legyen hely új absztrakciós szinteknek a már meglévők közé szúrására. Valójában a KL-One rendszerben megvalósitott jól-definiált osztályozó eljárás automatikusan új leirást helyezhet egy rendszertanba, SUPERC kapcsolatokkal kötve azokhoz a fogalmakhoz, amelyek őt a legjellemzőbben magukba foglalják, és azokhoz, melyeket ő rendre magába foglal.

4.2.1.2 A rendszertani szervezés szükségessége

A legtöbb szakértő rendszer alkalmazásban gyakran egyszerre teljesül számos szabály feltétel része, melyek közül egy sem magyarázza az egész feladatot, vagy egy sem szoritja ki a többi fontosságát. Ilyenkor figyelembe kell venni mindegyik szabály következmény részét. Előfordulhat, hogy egy helyzetleirás magába foglal egy másik, specifikusabb leirást, és ezek következmény részei kiegészithetik vagy tagadhatják egymást. Ezért megállapodásra van szükség, hogy meghatározhassuk, melyik szabályt kell előnyben részesiteni konfliktus esetén.

Egy klasszikus production rendszerben független szabályok esetén csak akkor derülnek ki a konfliktusok, amikor konfliktusos helyzet az input. Rendszertani osztályozó struktúrában viszont már a szabály rendszertanba történő asszimilásakor kiderülhet mindez. A specifikusabb szabályhoz kapcsolt következmény ekkor kifejezetten magában foglalhatja az információt az általánosabb szabály felülirására vagy kiegészitésére.

A szabályok rendszertani tudás struktúrába asszimilálása nemcsak az egymásrahatások inputkor történő felfedezését könnyiti meg, hanem - és ez már a hatékony jelölés eleme - elősegiti a szabályok tömör leirását is. Arra a tényre támaszkodva, hogy a fogalmak információt örökölnek az általánosabb fogalmaktól egy új szabály minta részéhez, rendszerint úgy hozható létre a fogalom, hogy csupán kisebb megszoritást adunk egy már meglévő fogalomra.

A KL-One esetén, amikor egy adott helyzetnél specifikusabb helyzet leirását akarjuk létrehozni, csak a módositandó vagy a további attributumokat kell megemliteni; az általános helyzet összes attributumát nem kell lemásolni. A memória-takarékosság mellett ez egyben megkönnyiti az adatbázis ellentmondás mentességének napra kész állapotba hozását és karbantartását, mivel az információról nem készit olyan másolatokat, amiket függetlenül kellene módositani, és amelyek véletlenül ellentmondásossá válhatnának. Ez is a hatékony jelölés eleme.

Az a lehetőség, hogy egy meglévő rendszertanba bármely szinten asszimilálhatók új leirások, megengedi az evolúciós rendszertervezést, ami ugyanolyan szigorú szabványokat használ, mint a top-down tervezés, csak anélkül, hogy megkövetelné a fogalmak előre meghatározott sorrendben történő definiálását. A legtöbb alkalmazásban még ha a kezdeti tervet gondosan, szigorúan top-down módon kidolgozva kapnánk is meg, a rákövetkező változások (pl. új adózási törvények következtében szükséges változások az elszámolásban) megkivánják, hogy a rendszert rugalmasabban lehessen módositani. Egy rendszer felismerhető helyzeteinek rendszertanát olyan

fejlődő tudásszerkezetként kell tekinteni, ami az emberi agyhoz hasonlóan folyamatosan finomítható és fejleszthető a rendszer élettartama alatt.

Rendszertani struktúra használata tekintélyes előnnyel járhat azon felismerési folyamat esetén is, melyben néhány jelenleg érzékelt elem egy ismert helyzet előfordulásait alkotja. Ez a folyamat durván annak feltételezéséből áll, hogy ezek az elemek a rendszer által ismert leírásokban lévő szerepek értékeként is értelmezhetők. Rendszerint azonban nem elegendő egy helyzetet egy meglévő leírás egyszerű előfordulásaként jellemezni. Egy helyzet leírásának általában olyan összetett objektumnak kell lennie, melynek különféle részei más, formálisan megengedett módokon összeállított fogalmak előfordulásai.

A helyzetfelismerés a mondatelemzéshez hasonlít, bár jelentősen bonyolultabb. Míg egy mondat részei között a nyelvtani kapcsolatok rögzítettek, egy helyzet "alkotóelemei" közötti kapcsolatok tetszőlegesek lehetnek. Ezek a kapcsolatok magukban foglalják az egymást időben megelőző eseményeket; az egymással különféle térbeli kapcsolatban lévő embereket, helyzeteket és fizikai objektumokat; az objektumokat fizikailag vagy jogosan birtokló embereket; a másokkal hatalmi viszonyban lévő embereket; és a bizonyos célokkal vagy célpontokkal rendelkező embereket.

A helyzetfelismerés hatékonyságának javítására szolgáló módszerek egyike a "faktorizált" tudásszerkezet (factored knowledge structure) [WO'80], amikor a különböző szabályok közös részeit úgy fésüljük össze,

hogy csak egyszer kelljen őket megvizsgálni. Ilyen szerkezetekkel hatékonyan vizsgálható egy nagy szabályhalmaz anélkül, hogy egyenként tekintenénk a szabályokat. A KL-One-ban megtestesitett rendszertani struktúrák faktorizált ábrázolást adnak a helyzetelemzésekhez. Az input helyzetet magukba foglaló legspecifikusabb fogalmak meghatározása úgy történik, hogy a helyzet elemeiből kiindulva a kapcsolatláncokon haladunk azon magasabb szintű fogalmak szerepeihez, melyekben résztvevők lehetnek - a mondatelemzéshez használt algoritmusok általánositásait és kiterjesztéseit használva.

Egy reprezentációnak ilyen algoritmusok támogatására való alkalmassága a hatékony jelölés fontos oldala. Ennek a technikának egy KL-One-t használó változatát sikeresen alkalmazták a PSI-Klone rendszerben. Ez egy ATN átalakitó, amihez egy KL-One rendszertan párosul, megszervezve egy természetes nyelvet megértő rendszer szemantikus értelmezési szabályait [BWE'80].

4.2.1.3 Kifejező erő

Bármilyen hatékony is néhány célra egy ábrázolás, mit sem ér, ha nem tudja kifejezni a szükséges megkülönböztetéseket. Ábrázolás keresésekor kerülendő olyan primitiv halmaz választása, mely a "sétál","fut", "poroszkál", "hajt" és "repül" közti árnyalatnyi különbségeket elmossa, vagy nem veszi tudomásul e sajátos fogalmak és a "mozog" általános fogalma közti közös vonást. A KL-One által megengedett strukturált, öröklèses hálózat mindkettőt elősegiti. Ahogy fontossá

válnak, új megkülönböztetések vezethetők be a meglévő fogalmak finomitásával vagy módositásával, és mindig lehetséges általánosabb fogalmak bevezetése is, melyek specifikusabb fogalmakból vonnak el részleteket.

A hagyományos predikátum kalkulus jelöléseiben egy tárgykör axiomatizálásakor általános probléma a predikátumok halmazának megválasztása, és annak eldöntése, hogy mik legyenek az argumentumaik. Ezek a döntések elkerülhetetlenül elhagyják azokat a megkülönböztetéseket, melyek más célból fontosak lehetnek, mint pl. az időváltozókat, helyzetváltozókat, a határozó módositás módjának kikötéseit, és a közbülső lépéseket és ágenseket. Ilyen döntések felülvizsgálata egy bonyolult rendszerben az axiomatizálás újrakészitésével egyenlő. A KL-One egyik célja - mely irányban némi haladás történt -, hogy ilyen axiomatizációkhoz a KL-One fogalmak adják a predikátumokat úgy, hogy pl. a tevékenységek idő szerepe virtuálisan mellőzhető olyan axióma-kifejezésben, melyben az idő nem szerepel ugyan kiemelkedően (21.ábra), de implicit módon mégis jelen van, esetleg később hozzáadódik, amikor olyan helyzettel kerül szembe, amiben az idő fontos.

Bár a kifejező erő minimális követelmény egy tudásábrázoló rendszerben, végülis olyan vázat akarunk, amiben a tetszőleges új információ asszimilálása nemcsak lehetséges, de valamilyen értelemben természetes is. Pl. kicsit akarjuk változtatni a tudást, hogy a tudásbázisban kis változásokra legyen szükség, de úgy, hogy a tanulási folyamat, vagy még a növekedési hibakeresés is várhatóan véglegesen konvergáljon.

Továbbá olyan operátoroknak kell létezniük, melyek finom igazitásokat végeznek, ahogy a helyes állapothoz közeledünk. Igy a hatékony jelölés másik oldala a kompaktság matematikai tulajdonságának analógiája - azaz a tudás lehetséges állapotainak terében olyan pontoknak kell lenniük, melyek tetszőlegesen közel vannak a tudás elérni kivánt állapotához.

A rendszertani osztályozási struktúrák kifejező erőt és hatékony jelölést előlegezhetnek az intelligens rendszereknek. Az ilyen módszerek végülis a számitógéptudomány egész területén alkalmazhatók lesznek. Az ábrázolás kifejező erejének hangsúlyozása - az adatstruktúrák számitási hatékonyságával szemben - előkészíti az ábrázolás általános módszertanának útját, ami majd felülemelkedik a különböző alkalmazásokon és a különböző megvalósitási módszereken. Ez az irányzat vezet majd a számitási viselkedés olyan magas szintű fogalmi operátorokkal történő specifikálásához, mely megfelel a programozó ember fogalmi struktúráinak, a megvalósitás hatékonyságának kérdését külön megfontolás, vagy még inkább automatikus fordítás tárgyává téve.

Ezen irányzat kezdetét az absztrakt adattipusok fokozott hangsúlyozása és az "objektum-orientált" programozás [RO'81] jelzi. A következő logikai lépés az absztrakt adattipusok fogalmának általánositása kifinomult tudásábrázoló rendszerben jelen lévő absztrakciós, öröklési és kifejező erővel rendelkező szintre. Ez a lépés új programozási stilust állithat elő, amit Goodwin [GO'79] "rendszertani programozásnak" nevezett. Ez a programozási stilus roppant előnyökkel birhat rugalmasság, kiterjeszthetőség és karbantarthatóság

tekintetében csakúgy, mint a dokumentálásban, felhasználó-oktatásban, hiba-csökkentésben és szoftver-termelékenységben.

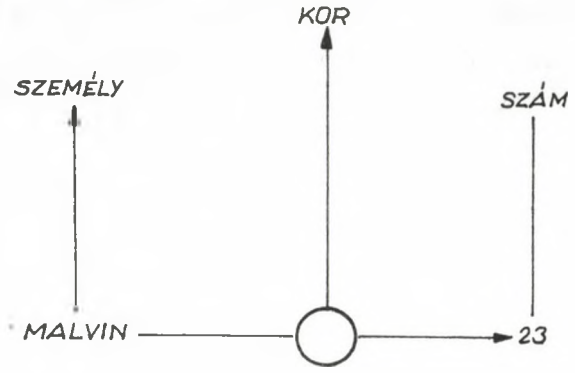
4.2.2 PSN (Procedural Semantic Network)[MST'83]

A PSN valójában az absztrakt adattipusok szemléletét ötvözi a frame javaslat néhány ötletével.

4.2.2.1 A PSN alapvető jellemzői

Egy PSN tudásbázis objektumokból áll, melyek lehetnek token-ek, osztályok (class), összekötő részek (link), relációk és programok. A token-ek sajátos entitásokat ábrázolnak, pl. Rózsa Sándor személyét, a "7" számot, vagy a "mese" karakterláncot, és az entitások közti bináris kapcsolatokat ábrázoló összekötő részeken keresztül kölcsönös kapcsolatban vannak. Az osztályok generikus fogalmakat ábrázolnak, pl. személy, szám vagy karakterlánc. A relációk pedig olyan generikus kapcsolatokat ábrázolnak, mint a "fivére" reláció egy férfi és egy másik személy között. Minden egyes token példával való szemléltetése legalább egy, de rendszerint több osztálynak, míg minden összekötő rész példával való szemléltetése legalább egy relációnak.

A 22.ábrán két token (Malvin, 23), két osztály (Személy, Szám), egy reláció (Kor) és egy összekötő rész (Malvint köti össze 23-mal) látható, INSTANCE-OF kapcsolatokkal Malvin, 23 és a Malvin-23 összekötő rész, valamint a Személy, a Szám és a Kor között.



22.ábra. Egy egyszerű konfiguráció

Az egyes osztályok "jelentését" négy-négy program adja, rendre leírva, hogyan kell beszúrni, eltávolítani és elérni (fetch) az osztály példával való szemléltetéseit, valamint hogyan kell megvizsgálni: vajon egy objektum az osztály példával való szemléltetése-e. Hasonlóan minden relációhoz négy program tartozik, melyek előírják, hogy hogyan kell beszúrni vagy elmozditani példa összekötő részeket, hogyan kell a tárgykör egy sajátos példájával kapcsolatban lévő reláció értelmezési tartományának minden objektumát elérni, és hogyan kell megvizsgálni, hogy két objektumot összekapcsol-e a reláció egy példával való szemléltetése.

Egy nagy tudásbázist éppúgy, mint egy nagy programot, intuitiv szervezési elvekkel kell struktúrálni, hogy tervezői és felhasználói megértsék. Sőt, erről a tudásábrázoló nyelvnek kell gondoskodnia. A PSN három ilyen elvet ad: az INSTANCE-OF, PART-OF és IS-A primitiv relációkban.

INSTANCE-OF: az objektumokat kapcsolja a tipusukat alkotó osztályokhoz. A tudásbázis minden objektumának legalább egy osztály vagy reláció példájának kell lennie. Ezt számos beépitett meta-osztály teszi lehetővé:

- "object" példája minden objektum önmagát is beleértve;

- "class" példája minden osztály önmagát és "object"-et is beleértve;

- "relation" példája minden reláció;

- "program" példája minden program.

Ezek az osztályok alkotják az INSTANCE-OF dimenzió tetejét. A dimenzió mélysége az alkalmazástól függ, és két token-tól, osztálytól vagy beépitett meta-osztálytól bármely mélységig terjedhet.

PART-OF: minden fogalom egyszerűbb fogalmak sajátosan rendezett aggregátumának tekinthető. Pl. egy ember tekinthető fizikai vagy szociális jellemzőinek (név, cim, személyi szám) aggregátumaként. Ez a szervezési elv slot-ok segitségével valósul meg. A slotok egy osztályhoz kapcsolódnak, és az osztály által ábrázolt fogalom részeit irják le. Minden slotnak van egy tipusa, és olyan egyéb korlátai, melyek behatárolják a slothoz kötheto objektumokat. Egy osztályt eggyel magasabb szintűnek tekintve, mint a slotjainak tipusait definiáló osztályokat, definiálhatjuk a PART-OF szervezési dimenzióit, mely

az összetett osztályokat feljebb és az atomi osztályokat alul helyezi el.

IS-A: a fogalmakat specializációikhoz/általánositásaikhoz kapcsolja. Mint osztályok közti részben-rendezés un. általánositó vagy IS-A hierarchiát definiál. Az IS-A hierarchiák kulcskérdése az általuk támogatott szokásos slot-öröklés fajtája.

Nem emlitettük még a programokat. Minden program valójában egy osztály, melynek slotjai határozzák meg

- a paramétereit,
- az előfeltételeit, melyeknek minden végrehajtás előtt igaznak kell lenniük, és
- a tevékenységeit, melyeket minden végrehajtás alatt kivitelezni kell

Mivel egy program egy osztály, vannak saját általánositásai vagy programjai, melyekből örökölhet paramétereket, előfeltételeket és tevékenységeket előiró slotokat [PAS'82].

Tsotsos [TS'80] más strukturáló mechanizmust vezetett be, hogy felismerésre is használhassanak PSN tudásbázist. Ez a mechanizmus, amit Minsky is javasolt [MIN'75], osztályok közötti hasonlósági összekötő részeket alkalmaz. Amikor egy adott osztály és az input adat közti illesztés eredménytelen, az eredménytelenségtől függően kivétel (exception) következik be, ami meghatározza, hogy melyik hasonlósági összekötő rész felhasználásával kell más osztályok illesztését javasolni.

4.2.2.2 A PSN megvalósitása

A nyelv jelenlegi megvalósitása rétegezett. PSN/0 olyan tudásbázis létrehozására ad lehetőséget, ami csak az INSTANCE-OF relációval strukturált. PSN/1 ezt az IS-A-val és a PART-OF egy egyszerű változatával bőviti. PSN/2 a PART-OF egy mesterkéltebb és drágább változatát vezeti be, hasonlósági összekötő részekkel és kivételekkel. A későbbiekben megvalósitandó rétegek feladata lesz pl., hogy a programokat osztályokként kezelje, ne atomi objektumokként, és ábrázolja a világi és okozati fogalmakat, pillanatnyilag ugyanis a slotok értékeinek korlátaiként kezeli ezeket a rendszer. A réteges megvalósitás fontos tulajdonsága, és ezt továbbra is meg akarják őrizni, hogy $j < i$ esetén egy PSN/i tudásbázis egyben PSN/j tudásbázis is.

A PSN felhasználót számos program segiti, vizuálissá téve az általa szerkesztett tudásbázist és elősegitve a további munkát:

- képernyős és szerkesztési funkciók, futási idejű támogatás, változat-vezérlő;

- a tudásbázis tartalmára vonatkozó kérdésekhez korlátozott, természetes nyelvű front-end lehetőség. A feleletek alakja némi PSN kód, egyszerű természetes nyelvű mondat, vagy a tudásbázis struktúrájára vonatkozó kérdések esetén a képernyőn pl. az IS-A vagy PART-OF hierarchiák, a hasonlósági háló, vagy az osztályok közti világi összefüggések grafikus, szines megjelenitése.

4.2.2.3 PSN/2 alkalmazás. Az Alven szakértő, számítógépes felismerő rendszer.

Plasztikai műtét - koronária áteresztés vagy billentyű protézis - után vizsgálják az emberi szív bal kamrájának működését, és igy a műtét hatékonyságát. A műtét alatt minden bal kamrafalba kilenc vékony tantalum spirált ültetnek be, durván egy síkban. Az aortához hivatkozási pontokként két szorító bilincset erősitenek. A műtét utáni utókezelő vizsgálat viszonylag egyszerű mozgó röntgenográfiát foglal magába.

Az Alven inputja a bal kamráról nyert képsorozat, de a módszer a dinamikus szivképek más formáira is alkalmazható. A cél hármas:

- e gazdag világi tárgykör szövegkörnyezetében a vizuális mozgás megértésére alkalmas módszertanokkal kisérletezni, és e módszertanokat tanulmányozni;

- ellentmondás mentesen és tárgyilagosan értékelni a bal kamra teljesitményét; és

- az idevágó kardiológiai ismeretek összeállitása és finomitása, hogy klinikai és kutatási eszközt adjon a kardiológusoknak.

Az Alven projekt 1976-ban kezdődött és 1983-ban implementálták másodszor egy jóval kiterjedtebb tudásbázissal, és olyan lehetőségekkel, melyek klinikai kipróbálásra is alkalmassá tették.

Az ábrázolt tudás néhány speciális jellegzetessége:

- az adatok gyakran "fuzzy" minőségűek, és az elfogadható értékek tartománya átfedheti különböző fogalmak megfelelő tartományait;

- az ábrázolásnak erős leíró alappal kell rendelkeznie, ami minőségi fogalmakat mennyiségi fogalmakkal hoz kapcsolatba, mivel a rendszer valódi jel-adatokkal dolgozik;

- a tudásbázisnak különböző absztrakciós szintű fogalomleírásokat kell magában foglalnia, a tudást bevivő személy szakértelmétől függően.

Az eseményeket PSN osztályokként, az események szemantikus összetevőit - pl. mozgó tárgy, térfogatváltozások, gyorsaságok, pályagörbék - slottokként ábrázolják. Ez utóbbiak várt jellemzői a kényszerek. Valahányszor sikertelen az illesztés, azaz az adott kényszer nem teljesül a megfelelő kivétel feltétel bekövetkezik. Az absztrakció különböző szintjeinek ábrázolására IS-A és PART-OF relációkat használnak.

Az időt intervallum-alapu sémával kezelik: minden eseménynek egy időtartam slot az egyik összetevője. A Time-Interval osztálynak három slotja van, melyek egy esemény kezdetének, végének vagy fennállásának idejét ábrázolják. A fennállás idejére azért van szükség, mert sok esetben nem ismert az esemény kezdetének vagy végének az időpontja, csupán az időtartamára vonatkozó kényszerek állnak rendelkezésre. Ez a séma megengedi, hogy

- az események idővonalán rugalmasan lehessen hivatkozni bármely pontra;

- egy sajátos esemény esetén e három időtartam sajátosság bármelyikére vonatkozó világi kényszereket ábrázolják; és

- az események között olyan relációkat lehessen kialakitani, mint During, Overlapping, Before és Simultaneous.

A rendszer kb. 80 általános mozgás fogalmat ismer fel, beleértve a kétdimenziós haladó mozgások és körmozgások bő választékát; a terület-, hossz- és alakváltozásokat; és az összetett mozgásokat, melyeket szimultán, átlapoló vagy sorozatos mozgások halmazaként definiáltak.

Hogy helyesen hasonlitsuk össze a szakértő rendszerekben lévő tudásbázisok méretét, figyelnünk kell az egyes rendszerek "tudás szemcsésségére". Azt találták, hogy az Alven osztályai általában több információt tartalmaznak, mint a tudásábrázoló nyelvként production rendszereket használó szakértő rendszerek ábrázolási egységei.

A felismerést vezérlő struktúra két paradigmán alapszik:

- a hipotézisek közti versenyen és együttműködésen, és
- a hipotézisek felállitásán és vizsgálatán.

A vezérlő struktúrát a tudásbázis struktúrája irányitja. A felismerés az általánostól az IS-A mentén halad a specifikus felé, a primitivtől a PART-OF mentén az összekapcsolt felé, a kölcsönösen kizáró események

halmazán keresztül a hasonlósági reláció mentén, és végül időben előre.

4.2.2.4 PSN/2 alkalmazás. CAA (Casual Arythmia Analysis)

Az EKG olyan "antennával" vett jelnek tekinthető, ami a szív elektromos tevékenységének egészét fogja fel, és így nincs egyszerű megfeleltetés a jelek sajátságai és a szív egyedi elektromos kisülései között. A főként az EKG-ból megfigyelhető szívrendellenességek fontos kategóriájába tartoznak a szívritmus zavarok. A ritmuszavarok észleléséhez tipikusan 24 órán keresztül gyűjtött EKG szükséges.

Az EKG elemzés évek óta jelentős figyelmet kapott. A jelenlegi rendszerek az összes EKG 80%-át helyesen értelmezik. Az e rendszerekben használt módszerek, beleértve a jelátalakitásokat és az egyszerű kontúr elemzést, úgy tűnik nem használhatók a maradék 20% elemzésében, különösen akkor nem, ha ritmuszavarok vannak a jelben.

A CAA rendszer megpróbálja a szív átvezető rendszerének fiziológiai ismeretét a jel ismeretével együtt alkalmazni a ritmuszavarok felismerésére és leirására. A munka 1979-ben kezdődött.

A CAA sokat átvett az Alven vezérlő struktúrájából, beleértve az IS-A, PART-OF és hasonlósági kapcsolatok használatát a hipotézis felállitásában. A legérdekesebb CAA sajátosság talán az okozati információ ábrázolása, azaz az egyik megkülönböztethető eseményből egy másikba

vezető hatásfolyam leirása. A CAA okozati összekötő részei két fontos módon hoznak kölcsönviszonyba eseményeket:

- megszabják egy okozott eseménynek az őt okozó esemény(ek)től való egzisztenciális függését;

- előirják az okozó és okozott események közti világi kényszereket.

Az okozati összekötő részek szerepe az, hogy helyi integritási feltételeket adjanak azokhoz az eseményekhez, amiket összefüggésbe hoznak. Igy ha egy eseményt azonositott a rendszer, az okozati összekötő részekkel keres időben előre vagy hátra más okozatilag összefüggő eseményeket, és megjósolja ezek valószinű időbeli helyét. Ezeket az eseményeket akkor hagyja jóvá, ha megfigyelhető alakmásuk megtalálható a jel jósolt időbeli helyein.

Ha egy megjósolt esemény nem figyelhető meg közvetlenül, a rendszer az eseménynek alapértelmezés szerinti slot értéket ad. Pl. ha egy nem megfigyelhető esemény indulási idejét más megfigyelt eseményekkel határozza meg, akkor időtartamát és igy befejeződésének idejét, az eseményre vonatkozó statisztikai információ visszakeresésével és a pillanatnyi szövegkörnyezetbe illesztésével becsli.

A CAA rétegezett tudásbázis tartalmaz

- egy morfológiai tudásbázist, ami az EKG hullámformák olyan oldalait irja le, mint alakjuk és időtartamuk; és

- egy fiziológiai tudásbázist, ami a szív ingervezetési rendszerére vonatkozó információt tartja karban.

Az ingervezetési események egy vetítési mechanizmuson keresztül függnek össze morfológiai alakmásukkal.

A hullámforma felismerés az EKG jelekben lévő kiemelkedő hullámformák felderitésével kezdődik. Ennek eredményeként áll elő a hipotézisek kiindulási halmaza. Ezután az Alven felismerő stratégiáját alkalmazva erre a halmazra, további morfológiai hipotéziseket állit elő és értékel ki a rendszer. Ezen hipotézisek előállitásakor a CAA megfelelő fiziológiai hipotéziseket akar felállitani a vetitési összekötő részeken keresztül, amelyek egy morfológiai eseményosztályt a lehetséges fiziológiai eseményosztályok választékával hoznak összefüggésbe, mindegyik kötési feltételekkel. Az általános ritmuszavar felismerésnek egy ütéssorozatot magában foglaló ritmus hipotézissel kell kezdődnie. A rendszer ezt a folyamatot használja fel az egyes vetitett osztályok vizsgálatára, és annak eldöntésére, vajon az osztályt összetevő hipotézisként tartalmaznia kell-e a pillanatnyi globális hipotézisnek. Ezt a folyamatot használva a rendszer rekurziv módon hipotézist állit fel az egymást követő ütés-eseményekre, és a megfelelő hullámsorozattal összehasonlitva az ellentmondásmentesség fokát kategorizálja.

4.3 Tudásbázisok építését/használatát támogató szoftverkörnyezetek

4.3.1 SIR (Semantic Information Retrieval) [RA'68]

A rendszer "szemantikus", mert

- a szövegből kivont és a program által tárolt aktuális információ szándék szerint megközeliti az anyag nyelvészeti "szemantikus tartalmát" vagy "jelentését"; és

- a SIR-ben használt számitógépes információábrázolás a formális matematikai logika "szemantikus" modell struktúráiból származik.

"Információ visszakeresésről" van szó, mert a SIR állitások gyüjteményein működik; tényeket keres vissza, hogy válaszoljon a kérdésekre. A SIR modell adatok dinamikus gyüjteménye, hiszen az új információ előidézheti az adatok automatikus bővitését vagy változását.

A tervezés követelményei:

- a modell szervezése legyen elég általános ahhoz, hogy széles tárgykörben hasznos legyen, de a tárolt információ legyen elég specifikus ahhoz, hogy valódi segitséget jelentsen a kérdések megválaszolásakor.

- a kérdésekre válaszoló eljárás kapcsán ügyelni kell az inputnak a modellben való kódolására és a válaszoknak a modellből való visszakeresésére. Egyik feladat sem lehet akadályozóan bonyolult vagy időigényes.

E követelmények kielégitésére a legjobbnak a szavakon és szó asszociációkon alapuló modellek tűnnek.

A SIR modellben a szavak maguk ábrázolják a szavakkal jelzett objektumokat vagy osztályokat, és a szavak közti asszociációk sajátos fajtái ábrázolják ezen objektumok vagy osztályok közti relációkat.

Az asszociációkat "leirólisták" (description list) ábrázolják. A leirólista elempárok sorozata, és az egész lista egy sajátos objektummal kapcsolatos. Minden pár első eleme egy objektum osztályra alkalmazható attributum neve, a második pedig ezen attributum értéke a leirt objektum esetén. Ha pl. az objektum a "3" szám, akkor leirólistája az attributumok és a hozzájuk kapcsolt értékek következő sorozatát tartalmazhatná (az attributumokat aláhúzzuk):

SUCCESSOR,4,ODD,YES,SHAPE,CURVY,...

Azt, hogy a "3" páratlan, egyszerűen az "ODD" attributum jelenléte - tetszőleges vagy semmilyen hozzá kapcsolt értékkel - jelezhetné, ha a rendszer képes az érték nélküli attributum felismerésére. A "macskák" osztályát leirhatná a következő lista:

SOUND,MEW,COLOUR,(BLACK,WHITE,YELLOW,BROWN),

LEGGEDNESS,4,...

ahol a COLOUR attributumhoz a lehetséges macska szinek listája kapcsolódik.

A LISP-ben a leirt objektumok egyedi szavak vagy "atomi szimbólumok", amikhez egy "tulajdonságlista" kapcsolódik. A SIR modell a tulajdonságlistákat használja.

A SIR csak egyszerű mondatokkal dolgozik, melyek valódi objektumokat vagy objektum-osztályokat jelölő szavakból és az objektumok és objektum-osztályok közötti sajátos kapcsolatokat kifejező szavakból áll. Ha az objektumokat és osztályokat egy formális rendszer egyedi elemeinek tekintjük, akkor ezek az objektumok és osztályok közti kapcsolatok a formális logika relációihoz hasonlók.

A számitógépes ábrázolásban mind az alapvető objektumok, mind a reláció nevei egyszerűen szavak. Tegyük fel, hogy az x szót a modellben az R reláció az y szóval társitja. Ez olyan mondatot ábrázol, ami azt "jelenti", hogy az x-szel jelölt objektumot vagy osztályt az R nevű reláció az y-nal jelölt objektummal vagy osztállyal társitja.

A SIR a következő relációkat tartalmazza:

- halmazbafoglalás (set-inclusion),
- rész-egész reláció,
- rész-egész relációhoz kapcsolt numerikus mennyiség
- halmaz elem reláció,
- balról jobbra térbeli relációk,
- birtoklás (ownership).

Egy angol mondatot a mondatforma-felismerő program értelmez; azt állítva, hogy az x-nek és y-nak nevezett objektumok és osztályok között az R-reláció áll fenn, e kapcsolatot úgy ábrázolja, hogy attributum érték párokat helyez mind az x, mind az y tulajdonság listájára. Mivel a relációk általában nem szimmetrikusak, az R-relációt olyan R1, R2 relációkra osztják, hogy

$\langle x,y \rangle \in R$ akkor és csak akkor, ha

$y \in R1(x)$ és $x \in R2(y)$.

Az x-ből y-ba vezető R1 attributumu összekötő rész:

1-tipusú, ha $\forall x \; \exists! y \quad y \in R1(x)$.
Térbeli relációk eseetén pl. csak egyetlen objektum lehet "közvetlenül jobbra" egy másiktól.

2-tipusú, ha R1 értéke objektumnevek listája.
A SZEMÉLY tulajdonságlistáján a SUBSET attributum értéke pl. (FIÚ,LANY,DIAK), ha függetlenül tudjuk meg, hogy minden fiú személy, minden lány személy és minden diák személy.

3-tipusú, ha R1 értéke tulajdonságlisták listája. A rész-tulajdonságlisták első eleme a PLIST flag, mutatva hogy tulajdonságlista következik. Minden rész-tulajdonságlistán szerepel a NAME attributum, aminek értéke 1-tipusú, és ez a lista legfőbb objektuma. Pl. az "A person has two hands" és az "A finger is part of a person" mondatok feldolgozása után a

PERSON tulajdonságlistája a következő attributum-érték párt tartalmazza:

SUBPART((PLIST,NAME,HAND,NUMBER,2)(PLIST,NAME, FINGER)).

A SIR úgy oldja meg a szemantikus elemzést, hogy csupán néhány mondatformát ismer fel, melyek mindegyike sajátos módon felel meg bizonyos relációknak. A megengedett input nyelvet szabályok listája definiálja. Minden szabály egy sajátos angol mondatformát ismer fel és ezen működik. A SIR által kapott mondatokat a lista minden szabálya megvizsgálja. A mondatra alkalmazható első szabály határozza meg a rendszer által végzendő tevékenységet, és azonnal meghivja a tevékenységet végrehajtó programot. Ha nincs alkalmazható szabály, a rendszer nem veszi figyelembe a mondatot, kivéve ha a rendszer megfelelően válaszol. A szabályok listája bővíthető.

A szabály alkotórészei:

- a minta;
- a mintában megjelenő változók listája;
- az alkalmazhatósági vizsgálatok listája; és
- a tevékenységlista, amit akkor kell végrehajtani, ha a mondat minden vizsgálatot kielégit.

Nézzünk egy példát:

((X IS A Y)(X Y)(ART ART)(SETR CAR CADR))

"ART" egy függvény neve, amely megvizsgálja, hogy

argumentuma két szimbólumból álló lánc-e, melyek közül az első határozatlan névelő. Ha igen, akkor értéke a lánc második szimbóluma, különben értéke "NIL".

"SETR": egy SIR függvény, ami halmazbafoglalás reláció létezését mutató összekötő részt hoz létre két argumentum között. "CAR" és "CADR" a "SETR" argumentumait adják.

Elemezzük a következő mondatot:

(A BOY IS A PERSON).

- X megfelelője "A BOY", Y megfelelője "A PERSON"
- art[A BOY]="BOY, art [A PERSON]="PERSON", és a rendszer létrehozza ezen értékek listáját: (BOY,PERSON)
- setr[BOY;PERSON] végrehajtása után a rendszer tartalmazza az elemzett mondatból e szabály által kivont relációs információt.

A kijelentő és kérdő mondatoknak külön-külön formái és megfelelő tevékenység függvényei vannak. A kérdőjel a "Q" szimbólum.

A SIR bizonyos esetekben egyetlen mintát (és szabályt) használ, noha számos alak lenne szükséges az igényelt tevékenységek egyedülálló módon való meghatározásához. A SIR rendszerben nincs "Every x is a y" és "The x is a y", csupán "x is a y". "Every boy is a person" pl. azt irja elő, hogy a "boy" halmaz a "person" halmazban van. "The boy is a person" viszont azt irja elő, hogy a "boy" halmaz néhány sajátos eleme a "person" halmaznak is eleme. E formai kétértelműség alkalmazhatósági

vizsgálattal feloldható.

A szemantikai kétértelműség részletesebb alakok használatával nem oldható fel. Ilyen kétértelműségre példa a

"John has ten fingers"

mondat, ahol a "have" jelentése "have attached as parts", vagy

"John has three marbles"

esetén "have" jelentése "own". E szemantikai kétértelműségek feloldhatósága a korábbi nem kétértelmű mondatok alapján létrehozott modellbeli szó-asszociációktól függ.

A SIR válaszadó mechanizmusa beépitett válaszformák halmazát foglalja magában. Ezek néhánya teljes, előre elkészitett mondat, mig másokat az őket használó programoknak kell még kinyomtatásuk előtt kiegésziteniük.

A SIR az igazság vizsgálatára egy heurisztikus tételbizonyitó programot használ. Ez az igazság vizsgálatát a leglényegesebb axiómákkal és modell kapcsolatokkal kezdi, és csak szükség esetén vezet be további tényeket. A modell struktúrája diktálja, hogy mi számit "leglényegesebbnek".

Angol kérdésre akkor és csak akkor "yes" a válasz, ha a tételbizonyitó a modellt illetően bizonyitani tudja a kérdésnek megfelelő SIR-mondat igazságát. A

tételbizonyitó az itéletkalkulus véges tartományában működik, és nincs felkészitve igazi minősitéses következtetésekre, mint pl. arra sem, hogy általánosan bizonyitsa:

$$(\exists y)(\forall x)P[x;y] \quad (\forall x)(\exists y)P[x;y].$$

4.3.2 Számitógép hardverének diagnosztizálása [DS'83]

Rosszul működő számitógép viselkedéséből a gép struktúrájára következtetve kell kinyomoznia a rendszernek a problémát.

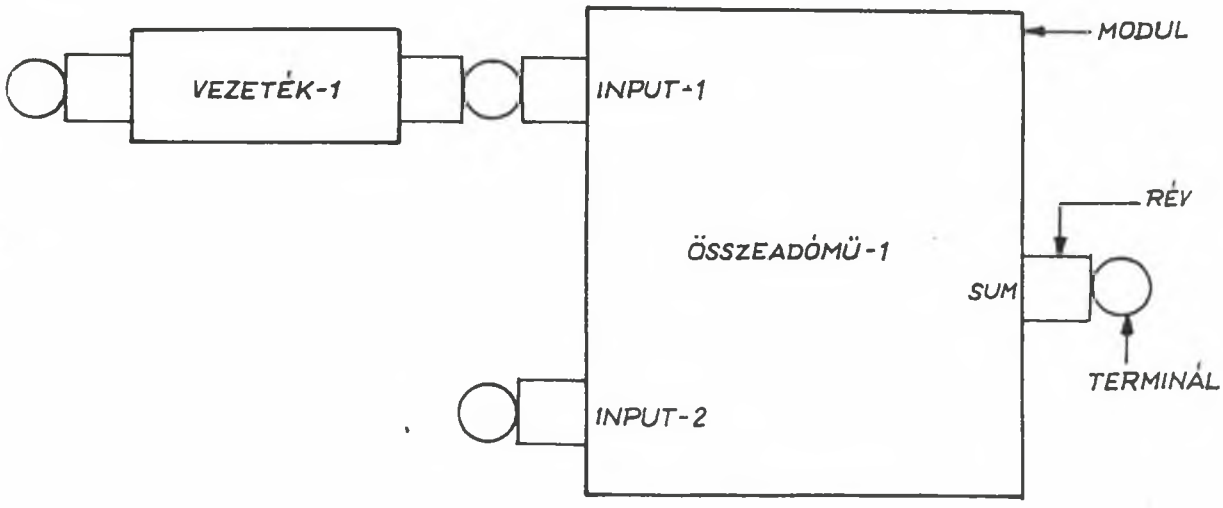
4.3.2.1 A struktúra ábrázolása

A strukturális leirás legalapvetőbb szintje három fogalomra épül:

- a modulokra, ezek szabványos fekete dobozok;
- a révekre (port), ezeken a helyeken áramlik az információ a modulba, vagy a modulból; és
- a terminálokra, melyek olyan primitiv elemek, ahol a réveken az eszközbe vagy az eszközből áramló információ megvizsgálásával próbálkozhatunk, de nem tartoznak a minket érdeklő struktúrához.

Minden révnek legalább két terminálja van, egy kivül és egy vagy több belül. A révek nyújtják az absztrakciós szint eltolásának fontos funkcióját. A révekben olyan gépezet van, ami megengedi, hogy külső termináljaikra

érkező információt belső terminál jaikra képezzenek. Két modul terminál jai egymásra illesztéssel kapcsolódnak össze (23.ábra)).



23.ábra. Alapvető strukturális fogalmak

Minden komponensnek van funkcionális és fizikai leírása. A funkcionális szervezés leírásánál pl. az összeadómű slice-okból áll, melyeket viszont half-adder-ek alkotnak stb. A fizikai szervezés leírásánál ugyanezeket a modul, rév és terminál fogalmakat használjuk, de a leírásokban cabinet-ek, board-ok és chip-ek szerepelnek.

A nyelv a szokásos értelemben hierarchikus: bármely szinten lévő modulnak lehet alstruktúrája. Leírásuk a funkcionális hierarchiában a logikai kapuk szintjén, a fizikai hierarchiában a chip-ek szintjén ér véget, mivel

ezek lényegében fekete dobozok, csak helyes vagy helytelen viselkedésük számit. A két hierarchia terminális szógpontjainál kapcsolódik össze.

A funkcionális szervezésben bármely nem terminális entry fizikai helyét összes levele fizikai helyzetének aggregálásával határozzák meg. A két leirás közti kapcsolatok alapján felelhet a rendszer bizonyos hasznos kérdésekre, pl. arra, hogy fizikailag hol található a cimforditó regiszter (mint működő rész), vagy hogy milyen funkció(ka)t hajt végre egy bizonyos quad-and gate chip.

Egy modul strukturális leirása a megépitésére vonatkozó parancshalmaz. Ezeket a parancsokat a rendszer végrehajtja, és olyan adatstruktúrákat hoz létre, melyek modellezik az összes alkotórészt és kapcsolatot, mert izomorfak a kérdéses struktúrákkal. Az adatstruktúrák LISP értelemben ugyanúgy vannak összekapcsolva, mint a modul objektumai. Az információ áramlás az összekapcsolások eredménye.

Egy számitógép teljes leirása egészen nagy lehet. Ezért un. "lusta" példával való szemléltetést alkalmaznak. Egy modul példával való szemléltetésekor csak a "héját" épitik meg, a külső dobozt a révekkel, a belső struktúrát pedig csak akkor, ha szükség van rá.

A leiró nyelv a DPL (Design Procedure Language) egy részhalmazának alapjára épült. A DPL eredeti implementálása szerint sajátosan VLSI tervezésre készült, de viszonylag könnyű volt "lehántani" a nyelv felső szintjét, ami chip-tervezéssel foglalkozott, és az

igy nyert bázisra felépiteni a korábban leirt nyelv új rétegeit.

Kifejlesztettek egy egyszerű áramkört rajzoló rendszert, ami lehetővé teszi a 23.ábrához hasonló képek interaktiv bevitelét. Az áramkörök bevitele egérmozgatások és billentyű leütések kombinációjával történik; az igy kapott struktúrák azután elemzéssel a nyelvben fogalmazódnak meg.

4.3.2.2 A viselkedés ábrázolása

A viselkedés leirására használnak

- egyszerű szabályokat, ha az eszköz viselkedése nem bonyolult;

- Petri-hálókat, ha párhuzamos események modellezésére összpontositanak;

- megszoritás nélküli kódot, mint utolsó segédeszközt, amikor a kifejezés strukturáltabb formái túl korlátozónak vagy borzalmasnak bizonyulnak; és

- ezek különféle konbinációit.

A kezdeti megvalósitás kényszerek használatán alapult. A kényszer egyszerűen egy kapcsolat; pl. egy összeadómű viselkedését kifejezheti, hogy az input-1, input-2 és sum réveken lévő terminálok logikai szintjeinek nyilvánvaló kapcsolata van. A kapcsolat gyakorlati

megvalósitásához olyan szabályhalmazt definiálnak, ami lefedi az összes különböző számitást, és ezek mint démonok figyelik a megfelelő terminálokat.

Egy modul teljes leirása tehát strukturális leirásából és viselkedési leirásából áll. Ez utóbbi a modul terminál jainak logikai szintjeit egymással kapcsolatba hozó szabályok formájában adott.

A megvalósitás különbséget tesz az elektromosság áramlását ábrázoló szimulációs szabályok, és a következtetés áramlását ábrázoló szabályok között.

A szabályfuttató mechanizmus olyan könyvelést vezet, amivel meghatározhatjuk, hogy egy terminálon megjelenő érték hogyan került oda. Ehhez minden terminál esetén nyomon követ

- minden szabályt, ami ezt a terminált inputként használja;

- minden szabályt, ami erre a terminálra értéket adhat, azaz outputként használja; és feljegyzi, hogy
- melyik szabály adta valójában a terminál pillanatnyi értékét.

Ez az információ-gyüjtemény az un. függőségi hálózat (dependency network).

A függőségi hálózat sokféleképpen használható. Elindulhatunk T_1 terminháltól, és a szabályokon keresztül visszafelé nyomozhatunk, hogy lássuk, milyen más

terminálok vettek részt a T_1-nél lévő érték kiszámitásában; vagy ellenkezőleg, előre nyomozva láthatjuk, hogy milyen más terminálok kiszámitásához kellett a T_1-nél lévő érték. Könnyen válaszolhatunk néhány kérdésre szimulációval: az eszközt bizonyos állapotba hozzuk, és azután feltárjuk a jövő alternativáit. Ha befejeztük, a megfelelő értékek törlésével minden tőlük függő feljegyzés is törlődik.

A rendszer memóriával rendelkező eszközöket is tud ábrázolni és modellezni. Először csak egy egyszerű globális óra létezését tételezték fel, majd kiterjesztették a szabályokat, hogy engedjenek meg relációkat egy terminálhalmaz jelenlegi és jövőbeli következő értékei között. Az egyszerű (érték, szabály) pár helyett (érték, szabály, időbélyegző) hármas utal az érték érvénybe lépésére. Minden terminálhoz egy ilyen hármasokat tartalmazó verem tartozik. Ennek alapján kiderithető, hogy pl. a 235. és 281.időpillanat között az összeadómű input-1 révjén az 1-es érték volt.

Az állapotváltozás modellezéséhez először a terminál pillanatnyi (érték, szabály, időbélyegző) hármasát teszik a terminál history listájának tetejére, majd ezt megismétlik minden olyan terminálra, melynek értéke az első terminál értékétől függ. Igy gyakorlatilag feljegyzi az eszköz pillanatnyi állapotának pillanatfelvételét. Ezután növeli a globális időszámlálót, új értéket ad annak a terminálnak, ahol az állapotváltozást kezdeményezték, és továbbadja azt az összes alkalmazható szabályon keresztül.

A viselkedés-leíró mechanizmus korlátja, hogy

- nincs nyilvánvaló támogatás a "terminálok közti kapcsolat" szemléletén kívüli hibás viselkedés kifejezésére;
  Egy busz-protokoll pl. további gépezetet igényel a protokollt leíró állapot-átmeneti hálózat ábrázolásához.

- nehézségeket okoz, ha szimbolikus kifejezésekkel kell dolgozni.
  Ha pl. tudjuk, hogy egy OR-kapu outputja 1, de egyik input értékét sem ismerjük, akkor egy szabály kifejezheti ugyan az egyik input értékét a másik input értékével, de problémát okoz, ha ezt a kifejezést máshol próbáljuk használni.

4.3.2.3 Az ábrázolás felhasználásai

A hibakeresés elsődleges alkotórésze a szimuláció és a következtetési szabályok kölcsönhatása. Egy eszközre ismertek a szimulációs és következtetési szabályok. Bizonyos input értékeket feltételezünk, és ezek alapján várjuk az output értékét. Ha a kérdéses outputot mérjük, és nem egyezik meg a várt értékkel, akkor ellentmondás van a szimulációs szabályok jóslása és az eszköz jelenlegi működése között. Az eszköz vagy rossz vagy helytelen inputot kapott. Ennek megfelelően a rendszer a jelöltek listájára teszi, az eszköz következtetési szabályaival következtet az input vonalak értékeire, és megnézi, hogy nincs-e ellentmondás elvárásunk és a kérdéses modul outputja között. Ha igen, akkor

megismétli az eljárást. A szimulációnak és a következtetésnek ez a kölcsönhatása megalapozza a hibakeresést, és olyan rendszert ad, ami sok érdekes képességgel rendelkezik. Az egyszerű logikai kapu szintű alkotórészeknél bonyolultabb eszközökön is bemutatták e megközelítés használatát.

Egy adott modulhoz kapcsolt viselkedés leírásai megengedik e modul viselkedésének szimulálását; a struktúra leírásában megadott modulok összekapcsolása előidézi az egyik modul által kiszámított eredmények átadását a másik modulnak. Így egyszerűen maga a leírás "futtatható". Hierarchikus megközelítéssel és a modul-rév-terminál szótárral nagyon könnyű a sokszintű szimuláció. Bármely modul szimulálásakor futtatható a

- modulhoz kapcsolt viselkedés, a modult szimulálva egy egyszerű lépésben; vagy a
- modul alstruktúrája, az eszközt struktúra szintjének megfelelően szimulálva.

A szimulációs szint változtatása a gyorsaság szempontjából hasznos, mert nem kell szimulálni az igazolt részstruktúrát, és egyszerű ellenőrzést is ad a struktúra és a viselkedés specifikációira. Összehasonlíthatjuk a modul viselkedési specifikációja által előállított eredményeket az eggyel alacsonyabb szimulációs szinten előállított eredményekkel. Az eltérések tipikusan hibát jelentenek az alacsonyabb szintű struktúrális specifikációban.

Struktúrális leírással jelezhetjük az információáramlás tervezett irányát, de ez a szimulációt nem vezeti félre.

Ez a megkülönböztetés fontos a hibakeresésben, mert a nehezebben felfedezhető hibák közé tartoznak azok, melyek hatására az eszközök nem úgy viselkednek, ahogy szerintünk "kellene", hanem úgy, ahogy elektromosan képesek. A tervezés és a megvalósitás elkülönitésére olyan prototipus modellek szolgálnak, melyek struktúrája és viselkedése bármely áramkörtől függetlenül definiált. Összeadóművek, vezetékek, AND-kapuk stb. külön-külön definiáltak. Eszközök létrehozásakor e prototipusok példával való szemléltetéseit szerkesztjük és kapcsoljuk össze. Bár ez a megközelités nem nyújt formális garanciát arra, hogy viselkedési definicióink pontosak, de egy egyedi modul viselkedési definicióját áramkörben tervezett használatától világosan megkülönböztetve, és az azonos tipusu modulokat ugyanazon viselkedési specifikáció használatára kényszeritve olyan környezetet ad, ami erősen felhivja erre a figyelmet.

A viselkedés leirása kényelmes mechanizmus hibabeszúrásra is. Egy nullán akadt vezeték modellezéséhez pl. olyan viselkedési specifikációt adunk, ami termináljait nulla logikai szinten tartja, minden megváltoztatásukra irányuló kisérlet ellenére. Hidak, osztottságok stb. hasonlóképpen könnyen modellezhetők.

4.3.3 Krypton [BFL'83]

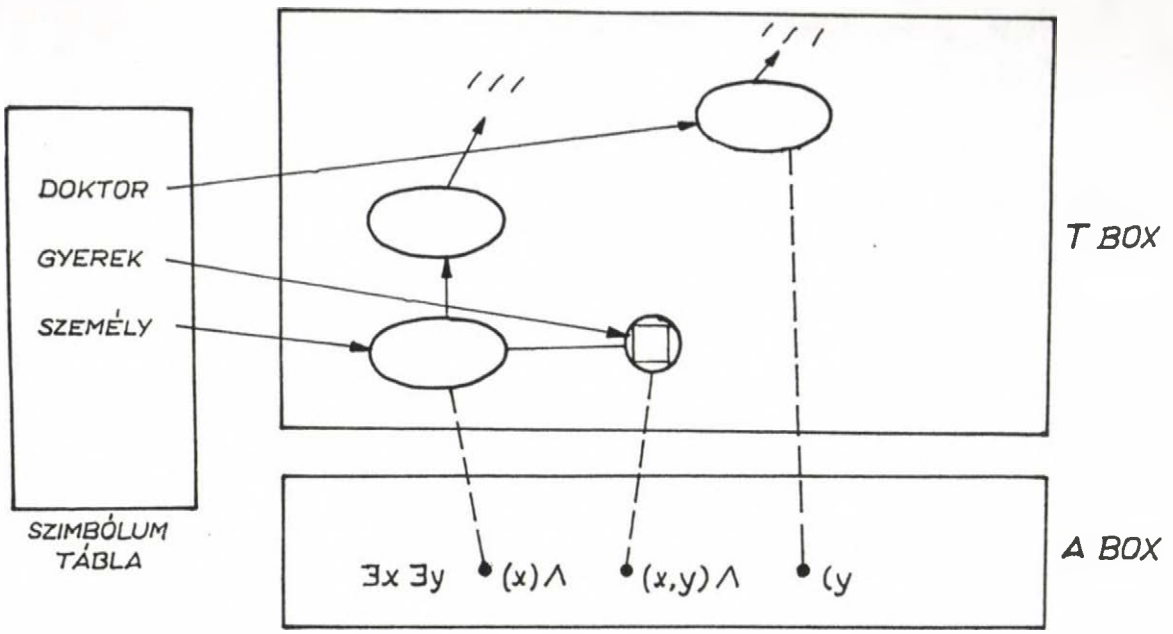
A rendszer tervezői a tudásbázis funkcionális specifikációjára összpontositottak. Arra a kérdésre próbáltak válaszolni, hogy milyen operációra van szükség egy tudásbázissal való együttműködéshez anélkül, hogy

bármit is feltételeznénk belső struktúrájáról. Olyan szoftverkörnyezetet akartak teremteni, melyben a felhasználó arra összpontosithat, amire tudásbázisba való, és nem kényszerül az implementációs részletek megismerésére.

4.3.3.1 A Krypton szerkezete

Az ábrázolórendszer két fő alkotórészből áll: egy terminológiai és egy állitási részből (24.ábra).

A T Box a rendszer terminológiai része. Struktúrált termek taxonómiájának kialakitására és a termek közti analitikus kapcsolatokra vonatkozó kérdések megválaszolására szolgál. Főnévi kifejezések formális ekvivalenseivel foglalkozik, mint pl. "a person with at least three children", és megérti, hogy ez a kifejezés alárendeltje az "a person with at least one child"-nak, és diszjunkt az "a person with no child" kifejezéssel. A felhasználónak nincs módja egy term taxonómiabeli helyének előirására.



24.ábra. A Krypton szerkezetének áttekintése. A T Box-ban rendszertanilag szervezett, strukturált termek vannak, az A Box olyan elsőrendű állitásokat tartalmaz, melyek predikátumai T Box termjei, és a szimbólumtábla tartja karban a T Box-beli termek neveit.

Az A Box a rendszer állitási része. Az érdeklődési tartományok leiró elméleteinek felépitésére, és ezen tartományokra vonatkozó kérdések megválaszolására szolgál. Mondatok formális ekvivalenseivel dolgozik, pl. "Every person with at least three children owns a car", és megérti az állitások logikai értelemben vett jelentését. A felhasználónak nincs módja előirni azt a tényt, hogy mely itéletek logikai következményei más, A Box-beli logikai itéleteknek.

4.3.3.2 A reprezentáció két nyelve

A T Box nyelv két kifejezéstipust támogat: a frame-eknek megfelelő un. fogalom kifejezéseket és a slotoknak megfelelő un. szerep kifejezéseket. A fogalmak és szerepek általában más fogalmak és szerepek egyesitésével vagy korlátozásával képződnek.

Pl. a "nőtlen szimbólum úgy definiálható, mint

(Conj Generic nem-házas-személy férfi)

feltéve, hogy a "nem-házas személy" és "férfi" szimbólumoknak van megfelelő definiciójuk. A

(VR Generic személy gyermek nőtlen)

jelentése "olyan személy, akinek minden gyermeke nőtlen".

Az NC Generic operátor egy adott szerep esetén a megtöltők halmazának számosságát korlátozza, pl.

(NR Generic személy gyermek 1 3)

jelentése "személy, akinek legalább egy, és legfeljebb három gyermeke van."

A szerepek is definiálhatók más szerepek specializációiként. Pl. a "fiú" a következő kifejezéssel definiálható

(VR Diff Role gyermek férfi),

ahol adott a "gyermek" szerep- és a "férfi" fogalom-term.

A T Box nyelv

Kifejezés	Értelmezés	Leirás
Fogalmak		
(Conj Generic $c_1 \ldots c_n$)	"c_1 és...és c_n"	Konjunkció
(VR Generic c_1 r c_2)	"c_1,melynek minden r-je c_2"	Értékmegszoritás
(NR Generic c r n_1 n_2)	"c, n_1 és n_2 közötti r-rel"	Számmegszoritás
(Prim Generic c i)	"i-ik féle c"	Primitiv részfogalom
(Decomp Generic c i j disjoint?)	"i-ik tipusu c a j-ik [kizárásos dekompozicióból"]	Dekompozició
Szerepek		
(VR Diff Role r c)	"r ami c"	Szerep megkülönböztetés
(Role Chain $r_1 \ldots r_n$)	"r_1 r_2-jének...r_n-je	Szereplánc
(Prim Role r i)	"i-ik féle r"	Primitiv részszerep
(Decomp Role r i j disjoint?)	"i-ik tipusu r a j-ik [kizárásos dekompozicióból"]	Dekompozició

A term-alakitó operátorok a megfelelő termek helyén is szerepelhetnek, pl.

```
(VR Generic (Conj Generic nem-házas-személy férfi)
            (VR Diff Role testvér férfi)
            (NR Generic személy gyerek 1   )).
```

Ez a kifejezés úgy olvasható, hogy "nőtlen férfi, akinek minden fivére gyermekes".

A Prim Generic és Prim Role operátorok egy fogalom vagy szerep primitiv specializációjára szolgálnak. Egy primitiv fogalom tartalmazó fogalma alá tartozik, de nincsenek elégséges feltételek annak meghatározására, hogy valamit leir-e. A család fogalma a következőképpen fejezhető ki:

```
(Prim Generic
 (Conj Generic
  (NR Generic
   (VR Generic szociális-struktúra apa férfi)
   apa 1 1)
  (NR Generic
   (VR Generic szociális struktúra anya nő)
   anya 1 1)
   (VR Generic szociális struktúra gyerek személy)))
```

Az A Box nyelv állitásos ábrázolási nyelv. Egy ilyen nyelvben a kifejező erő kérdése valójában annak kérdése, hogy a nem teljes tudás milyen mértékig ábrázolható. Ez a kérdés indokolja a diszjunkció, negáció és az egzisztenciális kvantor szabványos logikai itéleti konstrukcióit. Hogy rendszeresen lehessen foglalkozni a nem teljes tudással, és hogy valamelyest ellensúlyozzák a T Box állitási lehetőségektől való mentességét, az A Box nyelv struktúrája olyan, mint egy elsőrendű predikátum kalkulus nyelvé. Az itéletalkotó operátorok tehát a szokásosak: Not, Or, There Exist stb. A szabványos elsőrendű logikai nyelvtől a primitiv állitásokban tér el. Egy szabványos logikai nyelv nem-logikai, azaz predikátum-szimbólumai - és ha ilyen egyáltalán van, akkor funkció-szimbólumai - független, primitiv, tárgykörtől függő termek. Mivel a T Box ad lehetőséget a tárgykörtől függő kifejezések gyűjteményének leirására, a T Box nyelv termjeit tették az A Box nyelv nem-logikai szimbólumaivá. Amikor frame-ek és slotok nyelvét predikátum kalkulusba forditjuk, a frame-ek és slotok rendre egy- és kéthelyű predikátumokká válnak [HA'79]. A Kryptonban az igy eredményezett predikátumok nem primitivek;

definició szerint egymással kapcsolatban lehetnek, bármely A Box nyelven kifejezett elmélettől függetlenül.

4.3.3.3 A Krypton alkotóreszeinek működése

A felhasználó a T Box és A Box nyelvek feletti operációk rögzitett halmazához férhet hozzá. A felhasználó és a Krypton tudásbázis között minden kölcsönhatást ezek az operációk közvetitenek. Az operációk két csoportra oszthatók: az un. Tell operációk a tudásbázis nagyobbitására, mig az un. Ask operációk információnyerésre szolgálnak. Az operációk mindkét esetben lehetnek definiciósak vagy állitásosak.

T Box

Tell: egy szimbólumot egy T Box termmel társit. A tudásbázist úgy változtatja, hogy szótára magába foglalja a term definiálta szimbólumot.

Ask: Ask_1: egy T box term magában foglal-e egy másikat.

Ask_2: egy T Box term fogalmilag diszjunkt-e egy másikkal.

A Box

Tell: egy A Box kijelentésről azt állitja, hogy igaz. A tudásbázist úgy változtatja, hogy világelmélete foglalja magába ezt a kifejezést.

Ask: egy kijelentésről megkérdezi, igaz-e. Az eredményt a tudásbázis pillanatnyi elmélete és a kijelentésben használt, T Box-ban definiált szótár határozza meg.

Egy tudásbázisban természetesen további Ask operációknak is kell lenniük. Az A Box-ban meg kell találni, hogy mely egyedek adott tulajdonságúak, a T Box-ban pedig valahogy információt kell kapnunk a definiciókból, amit a fenti kérdések nem adnak meg. Pl. mennyi egy "háromszög" "szögeinek" száma?

A Tell és Ask megvalósitására használt valóságos szimbolikus struktúrák a felhasználó számára nem elérhetők. A tudásbázist úgy kezelik, mint egy absztrakt adattipust, amit inkább egy operáció-halmazzal jellemzünk, semmint egy bizonyos implementációs struktúrával.

4.3.3.4 A Box létrehozása

Ha a predikátum szimbólumok valóban T Box termek, akkor az A Box következtető mechanizmusának hozzá kell férnie e termek T Box-beli definicióihoz. Pl. ha megmondták,

hogy Bimbó tehén, akkor az A Box-nak is tudnia kell, hogy Bimbó állat, és nem bika.

Az A Box predikátumok nem egyszerűen összekapcsolatlan primitivek - mint az elsőrendű logikában -, így ha szabványos elsőrendű következtetési technikát kívánnak használni, valahogy létre kell hozni a T Box-ban lévő összekapcsolásokat. A Kryptonban a szabványos következtetési szabályokat úgy terjesztették ki, hogy vegyék számításba a T Box definiciókból származtatható predikátumok közti függőségeket.

Pl. egy szabványos rezolúciós tételbizonyító következtetése annak észrevételén múlik, hogy $\varphi(x)$ előfordulása az egyik klauzulában inkonzisztens $\neg\varphi(x)$-szel egy másikban. Ezt felismerve a két klauzula egy eloszlató hatású klauzula következtetésére használható. E következtetési szabály érvényességi tartománya növelhető, ha két literál összeegyeztethetetlenségének felismerésére további eszközként felhasználjuk a T Box-ból az általánositási és diszjunktivitási információt. Azaz, ha φ és ψ diszjunkt, akkor $\varphi(x)$ és $\psi(x)$ inkonzisztens, és ha φ magában foglalja ψ-t, akkor $\neg\varphi(x)$ és $\psi(x)$ inkonzisztens. A helyzetet bonyolitja, hogy a T Box definiciók "feltételes" inkonzisztenciákat is magukba foglalnak. Tegyük fel pl., hogy a "téglalap" definició szerint "olyan sokszög, melynek minden szöge derékszög". "Sokszög"(x) csak akkor inkonzisztens $\neg$"téglalap"(x)-szel, amikor x minden szöge derékszög. Ilyenkor a feltételesen inkonzisztens literálokat tartalmazó klauzulák még mindig megoldhatók lehetnek, ha a feltétel negációját a feloldó hatásba beleértjük. Így, ha a T Box-ot megkérdezik, hogy a "sokszög" és a $\neg$"téglalap" disz-

junkt-e, azt a választ kellene kapniuk, hogy "csak akkor, ha minden szög derékszög".

4.3.3.5 T Box létrehozása

Az A Box következtető mechanizmusának a T Box-beli általánositási és diszjunktivitási információt a következtetési lépések között kell elérnie, ezért az A Box-nak szükséges időhöz képest elfogadhatóan gyorsan kell végrehajtani a T Box operációkat.

Tetszőleges "lambda-definiálható" predikátumokat megengedő nyelv esetén a tartalmazásra nincs teljes, még kevésbé hatékony algoritmus. A T Box nyelvet - a már tárgyalt operátorokkal - "frame-definiálható" predikátumokra korlátozva van némi esélyünk, hogy használható algoritmust kapjunk, és még mindig nyújtsuk a term-alakitó lehetőségek olyan halmazát, ami a mesterséges intelligencia alkalmazásokban hasznosnak bizonyult. A megoldás még messze van. Úgy tűnik, hogy a termek közti általánosítás kiszámitásának bonyolultsága végtelenül érzékeny a term-alakitó operátorok megválasztására. Pl. VR Diff Role operátor nélküli egyszerű T Box nyelv esetén a termek közti általánositás algoritmusa láthatóan a legrosszabb esetben $O(n^2)$; ezzel az operátorral azonban a problémának nincs nem-exponenciális megoldása.

Az általánositó kapcsolatokra a definiált szimbólumok kifejezett taxonómiáját tartják karban. Módszereket fejlesztenek ki, hogy ez mind abszolut, mind feltételes diszjunktivitási információt is tartalmazzon a T Box termekről. A kulcskérdés még megválaszolatlan: hogyan

kell meghatározni a szimbólumok közt definiálható összes feltételes kapcsolat hasznos részhalmazát.

Átvették a KL-One-beli osztályozó ötletét [SL'83]. A taxonómia és az osztályozás azonban csupán implementációs stratégia. A T Box nyelv jelentése és az operátorok definíciója egyáltalán nem függ a taxonómiától, vagy az osztályozó használatának előnyeitől.

III. SZEMANTIKUS ADATMODELLEK ÉS FOGALMI LEIRÓ NYELVEK

1. Szemantikus adatmodellek meghatározása

1.1 A modellezési folyamat a számitástechnikában

Az információfeldolgozás a múltban és a jövőben is a számitógépek egyik legfontosabb és legjellemzőbb felhasználási területe volt és lesz. Ennek az információfeldolgozási munkának egyik központi kérdése a megfelelő adatmodell elkészitése, ami röviden a következő módon foglalható össze:

Kiindulva egy már létező, intenziv adatfeldolgozással járó tevékenység követelményrendszeréből (pl. egy adott vállalat könyvelésének leirásából), készitsük el ennek a tevékenységnek egy olyan reprezentációját, amely alkalmas a számitógépes megvalósitásra, és képes arra, hogy a folyamatban előforduló összes lényeges feladat végrehajtását támogassa. Mivel a modellbe épülő számitógépes információs rendszert általában hosszabb időn át kivánjuk használni, ezért ezen túl ezen modellnek arra is képesnek kell lennie, hogy az időben változó körülményekhez egy bizonyos mértékig alkalmazkodni legyen képes.

Sajnos a fenti feladat csak egészen egyszerű, mechanikus tevékenységekre végezhető el tökéletesen. Ennek alapvető oka az, hogy feladatok végrehajtásában az emberek sokkal rugalmasabbak, mint a számitógépek, és ezen kivül képesek arra, hogy a kitűzött problémákra megoldó

stratégiákat dolgozzanak ki, illetve tanuljanak meg, márpedig ezen stratégiák számitógépes reprezentációja igen nehéz kérdés.

Ezzel a kérdéssel foglalkoznak a tanulmány második fejezetében leirt "tudásábrázoló rendszerek", de ezek részletes kidolgozása még hosszú kutatómunkát kiván, és a jelenleg forgalomban lévő számitógépes adatfeldolgozó rendszerekeben a már hozzáférhető tudásábrázoló rendszerek sem igen nyertek még alkalmazást. Igy az adatmodellezési munkában általában kompromisszumot kell kötnünk, és ezért a jelenlegi számitógép információs rendszerek felhasználói kényszeritve vannak arra, hogy gondolkodásukat többé-kevésbé hozzáidomitsák a számitógéphez, és kénytelenek olyan fogalmakkal is megbirkózni, amelyek nem az adatok természetéből, hanem a jelenlegi számitógépek felépitéséből fakadnak. A fogalmi modellezés (=conceptual modelling), és ezen belül az un. szemantikus adatmodellek kidolgozásának a legfontosabb célja éppen az, hogy az adatmodelleket közelitsük a természetes fogalmakhoz, és másrészt a megfelelő modell elkészitését, tehát magát a modellezési folyamatot is megkönnyitsük.

Tekintsük át röviden az adatmodellek, illetve ezek használatára épülő adatbázis kezelő rendszerek legfontosabb alapfogalmait.

1.2 A modellek általános felépitése

Adatmodellnek nevezzük a jól meghatározott fogalmak egy olyan gyűjteményét, amely segit valamely probléma tulajdonságainak leirását elkésziteni. Itt egy adatmodell meghatározásáról általában megköveteljük, hogy teljesen szabatos legyen, azaz matematikai értelemben vett definició legyen. A modellezendő folyamatról pedig azt tesszük föl, hogy a következő (egyelőre nem teljesen egyértelmű) elemekkel jól leirható legyen:

- állandó (=static) alkotórészek. Ilyenek a modellezendő feladatban szereplő objektumok /=entity/, az ilyen objektumok bizonyos tulajdonságai, és bizonyos kapcsolatok az objektumok között;

- változó (=dynamic) alkotórészek. Ilyenek az objektumokon végrehajtandó műveletek, az ilyen műveletek tulajdonságai;

- bizonyos szabályok illetve megszoritások, amelyek részint a folyamat alapvető tulajdonságaiból, felépitéséből következnek, részint pedig a folyamatban résztvevő objektumokra illetve műveletekre előirt (pl. mennyiségi) követelményekből.

Ennek megfelelően a modellezési folyamat eredményének, azaz magának az adatmodellnek két, többé-kevésbé elkülönülő része lesz az állandó illetve változó elemekre való felbontásnak megfelelően (a figyelembe veendő szabályokat a megfelelő módon irjuk le):

1. Egy matematikai fogalmi rendszer, ami képes az adatobjektumok tulajdonságait kifejezni, és a köztük lévő kapcsolatokat leirni.

2. Műveletek halmaza, amelyekkel végre tudjuk hajtani a megkivánt tevékenységeket az adatok halmazán.

A szokásos terminológiában az 1.pontban szereplő fogalmi rendszert sémának nevezzük. Igy a séma tartalmazza az adatmodell objektumainak leirását, a köztük lévő kapcsolatok definicióját, és a köztük kötelezően meglévő, állandóan fennálló szabályok felsorolását. Itt egy-egy objektum tulajdonságainak listáját a megfelelő objektum tipus definiciójának nevezzük.

Maga a séma nem más, mint egy terv a modellezendő folyamat egy aktuális példánya modelljének elkészitésére. Ezt a konkrét modellt (amely egy számitógépen ábrázolható adathalmaz) egy adatbázis példánynak (=instance of the database) nevezzük. Ez az adatbázis tehát tartalmaz objektumokat, amely objektumok eleget tesznek a sémában előirt valamely objektum-tipus definicónak. Ezen objektumok között természetesen fennállnak kapcsolatok, és ezen kapcsolatok is eleget tesznek a séma előirásainak.

Igen fontos észrevétel az, hogy a műveletek bizonyos természetes osztályainak nem szükséges, sőt, néha nem is szabad (pl. titkossági okok miatt) az összes sémában definiált objektummal foglalkozni. Igy szükségünk van arra, hogy a séma bizonyos részeit kijelölhessük. A séma egy ilyen, előre meghatározott részét alsémának nevezzük, és igen hasznos, ha a műveletek egy-egy

osztályán a megfelelő alsémát könnyen meghatározhatjuk és leirhatjuk.

1.3 A modellekhez szükséges fogalmak részletes áttekintése

1.3.1 A séma felépitése

Tekintsük át röviden azokat az alapvető fogalmakat, amelyek szükségesek a séma fölépitésében, azaz az állandó tulajdonságok modellezésében. Először is a séma tartalmazza az adatobjektumok leirását. Ezen objektumok lehetnek egyszerűek, vagy összetettek. Az egyszerű objektumok olyan adathordozó egységek, amelyek felbonthatatlanok, és képesek az önálló létezésre. (Például a klasszikus rekord-alapu rendszerekben az egyszerű objektumok leirását éppen a rekord-definiciók alkotják.) Az összetett objektumok két vagy több alkotórészből felépitett egységek, és természetesen csak akkor létezhetnek, ha minden alkotórészük létezik.

1.3.1.1 Objektumok

Annak meghatározása, hogy pontosan mik legyenek az egyszerű objektumok, és belőlük milyen összetett objektumok épüljenek fel, a modellezési folyamat egyik fontos és nehéz döntése.

Az objektumokra természetesen létezhetnek különböző megszoritások. Ezek a legegyszerűbb esetben bizonyos

mennyiségi előírások, amelyek segítenek a hibás adatokat még a bevitelkor kiszűrni, és így a modell korrektségét megőrizni. Másik példa az, hogy gyakran előírjuk, hogy egy objektum-típuson belül egy konkrét objektumot egyértelműen meghatározzon egy kulcs, azaz az objektumnak legyen egy vagy néhány olyan tulajdonsága, amely két különböző objektumra biztosan eltér.

Egy objektum egy elemi, tovább nem bontható állandó tulajdonságát attribútumnak nevezzük, és ha az objektum egy konkrét példányáról beszélünk, akkor a megfelelő aktuális értéket attribútum-értéknek nevezzük.

Így egy attribútum önmagában nem is létezik, és csak a megfelelő objektum-típushoz kapcsoltan értelmes beszélni róla. Például, ha a rendszerünkben személyi nyilvántartást akarunk vezetni, akkor a sémában valószínűleg lesz egy "személy" nevű objektum-típus, amelyhez kapcsoltan felsoroljuk a személyek számunkra fontos adatait (például "név", "személyi szám" stb.). Itt a "név", "személyi szám" stb. a "személy" objektum-típus egy-egy attribútuma.

A sémához természetesen annak megadása is hozzátartozik, hogy egy objektum egy adott attribútumra milyen értéket vehet fel, azaz le kell írnunk az attribútumok típusait. Általában feltételezzük, hogy rendszerünkben felhasználhatjuk egy magas szintű programnyelv (pl. Pascal, C, Fortran) minden szolgáltatását, és így az alapvető numerikus (egész és valós), illetve karakter-sorozat (fix- vagy változó-hosszú) típusokat adottnak tekintjük. Az egyszerűbb rendszerekben az attribútumok csak ilyen elemi típusúak lehetnek, de az általunk vizsgált fejlet-

tebb adatmodellekben az attributumok tipusa lehet bonyolultabb adattipus is, amelyet természetesen az adott adatmodell terminológiáját használva pontosan le kell irnunk.

Az attributumok meghatározásában a megengedett attributum-tipusokon kivül is jelentős különbségek vannak az egyes adatmodellek között. Például egyesek előirják, hogy egy objektum minden attributumának jól meghatározott értéknek kell lennie, mig mások megengedik, hogy bizonyos attributumok nem definiáltak legyenek (azt általában a speciális "null" értékkel jelölhetjük meg). Egyes rendszerek nem engedik meg, hogy egy objektum élete során az attributumainak értékeit megváltoztassuk, mig másokban ez lehetséges. Egyes rendszerek biztositják, hogy az attributum értéke a megfelelő attributum-tipusu elemek egy halmaza is lehet (többértékű attributumok). A nagyobb erejű modellek általában jobb és könnyebb modellezést tesznek lehetővé, de természetesen nehezebben implementálhatók.

Fontos már most megjegyeznünk, hogy a fentiek ilyen szigorú formában elsősorban az un. klasszikus adatmodellekre, és azok továbbfejlesztéseire igazak. Amint látni fogjuk a későbbiekben, néhány újabb adatmodell (pl. a funkcionális adatmodell) elkerüli az attributum fogalmát, és helyette más módszert kinál az adatobjektumok tulajdonságainak leirására.

1.3.1.2 Az objektumok közötti kapcsolatok leírása

A sémában természetesen nem elegendő az adatobjektumok tipusait külön-külön leirni, mivel általában a modellezendő folyamatban az egyes elemek összetett módon kapcsolatban állnak egymással, és éppen ezek a kapcsolatok alkotják a folyamat lényegét. Ezeknek a kapcsolatoknak az ábrázolása az adatmodell egyik kritikus módja.

Az adatmodellek három jól - elsősorban a kapcsolatok leirásának módja szerint - körülhatárolható csoportját klasszikus adatmodellnek fogjuk nevezni. Ezek a hierarchikus, hálózatos, illetve relációs adatmodellek. Ez az elrendezés indokolt, mert egyrészt ezen adatmodellek elméleti megalapozása jó [ULL'82], másrészt a jelenleg kereskedelmi forgalomban lévő számitógépes információs rendszerek ezen adatmodellek egyikére (vagy esetleg ezek egy kombinációjára) épülnek.

1.3.2 A műveletek meghatározása

A séma után pedig meg kell adnunk az adatmodell dinamikus tulajdonságait, azaz az objektumokon végrehajtható műveletek leirását. Fontos éreznünk, hogy a műveletek két csoportba oszthatók, egyrészt olyanok vannak, amelyek az adatobjektumok megváltoztatására szolgálnak (ezek az un. adatmanipulációs műveletek), másrészt vannak olyanok, amelyek a rendszerben tárolt információ visszanyerésére (un. lekérdező műveletek) használtatnak. Ez egy természetes különválasztás, amely jelentősen megkönnyiti a műveletek leirását, noha néhány újabb adatmodell elkerüli ezt a megkülönböztetést.

1.3.2.1 Adatváltozás

Az adatmanipulációs műveletek megoldása általában a következő módon lehetséges: Vannak bizonyos, a rendszer által kínált elemi műveletek, amelyeknek a jelentése egyszerű és örökre meghatározott. Ezeket a szokásos szóhasználattal primitíveknek nevezzük. (Tipikus példa erre a sok rendszerben megtalálható "insert" parancs, amellyel egy új adatobjektumot tudunk létrehozni.) A sémát felépítő személy ezekből összeállíthat bonyolultabb műveleteket, amelyet tranzakcióknak nevezünk. (Tehát az adatmanipulációs műveletek összeállítását úgy is elképzelhetjük, mint ha egy assembler nyelvben programoznánk, ahol az elemi műveletek a primitívek.) A modell korrektségének megőrzését megkönnyítendő, a tranzakciók általában un. atomos módon hajtódnak végre, ami azt jelenti, hogy ha a tranzakció valamely komponensének végrehajtása nem sikerül, akkor az egész tranzakció meghiusul, és az esetleges addigi részleges végrehajtásának eredménye törlődik.

Természetesen az adatmanipulációs műveletek között is szükségünk van kapcsolatokra. Részint ezekkel, részint az un. dinamikus megszorításokkal tudjuk biztosítani a művelet helyes végrehajtását. A dinamikus megszorításokat általában egy-egy művelet elé és után helyezzük el, az előbbi az un. előfeltétel, az utóbbi az eredmény, vagy utófeltétel. Az előfeltétellel tudjuk biztosítani, hogy a rendszer csak akkor hajtsa végre a műveletet, ha az értelmes (azaz megvannak a feltételei), az utófeltétellel tudjuk leírni a művelet pontos hatását. Ezekkel a feltételekkel könnyen le tudjuk írni

az egymás után következő műveletek közötti kommunikációt.

Másrészt ilyen módon el tudjuk kerülni az implementációs részleteket a műveletek megadásánál. Ez a 70-es évek közepétől egyre nagyobb szerepet játszik). mivel így bizonyos nem lényeges részleteket el tudunk rejteni. (Ez motiválta az un. absztrakt adattipusok bevezetését [lásd AHU'83], ahonnan a megfelelő fogalmak és eljárások átkerültek a fogalmi modellezésbe is.)

1.3.2.2 Információ visszanyerése

A lekérdező műveletek megadása általában úgy történik, hogy meg kell határoznunk egy logikai kifejezést, a sémában megadott objektum leirásokat és kapcsolat meghatározásokat felhasználva. A lekérdező művelet eredménye az adatbázisban éppen aktuálisan szereplő objektumok azon halmaza, amelyek eleget tesznek ezen logikai kifejezésnek. (Bizonyos lekérdező műveletek eredménye lehet valamilyen összegyűjtött mennyiségi információ is, pl. hogy hány olyan személy van, aki 1950 és 60 között született.)

1.4 Az előzőleg vázolt fogalmak megvalósitása

A fent felsorolt modellezési folyamat végrehajtásához természetesen a rendszernek kinálnia kell a megfelelő eszközöket. Ezek általában nyelvek, amelyekkel definiálni, manipulálni, lekérdezni lehet az adatbázis aktuális állapotát. Igy a legtöbb jelenlegi rendszerben megtalálható egy adat-definiáló nyelv (a szokásos angol rövidítéssel DDL) séma és alsémák meghatározására, egy adat-manipulációs nyelv (DML), és egy lekérdező nyelv (QL) az adatmanipulációs illetve lekérdező műveletek megirására. Ezen felbontás nem kötelező, sok rendszerben az adat-manipulációs és a lekérdező nyelv egy egységet alkot, sőt néha a három nyelvet egy egységgé vonják össze.

1.5 A szemantikus adatmodellek bevezetésének indokai

Most pedig lássunk néhány indokot, amelyek a szemantikus adatmodellek bevezetését alátámasztják, és néhány célt, amelyeket meg akarunk velük valósitani.

Először is, maga a szemantikus adatmodell kifejezés egy nem preciz fogalom. A szakirodalomban többen [pl. TL'82] megállapitották, hogy igen nehéz szabatos definiciót adni rá. Mi egyszerűen egy adatmodellt szemantikus adatmodellnek fogunk nevezni, ha gazdagabb, jobban kifejező fogalmakat kinál, mint a klasszikus adatmodellek, és ezekkel a modellezendő folyamat jelentésének (szemantikájának) több oldalát lehet megragadni.

Tehát az indokok és célkitűzések:

- A modellezési erő növelése.

Ez természetesen az elsődleges szempont. A klasszikus adatmodellek közül a hierarchikus és a hálózatos adatmodell leggyengébb pontja a kapcsolatok ábrázolása, amely kizárólag speciális próbaállitások módján (az un. sok-egy bináris reláció segitségével) történhet. Ez azt jelenti, hogy egy kapcsolatban csak két komponens vehet részt, és egy-egy kapcsolaton belül az egyik résztvevőnek fixnek kell lennie. (A hierarchikus modellben még további megszoritás az, hogy egy kapcsolaton belül a pároknak fákba szervezetten kell felépülniük.) Ez igen komoly megszoritás, emiatt részint a lekérdező műveleteket csak meglehetősen primitiv módon (egy időben egy rekordot tekintő eljárással) tudjuk felépiteni, másrészt a felhasználóknak a fizikai adatelhelyezésből fakadó problémákkal kell megküzdeniük.

A relációs adatmodell a matematikai reláció fogalmára épül, amely nem más, mint egy rögzitett n-re rendezett n-esek egy halmaza. Ezek a rendezett n-esek összetett adatobjektumok, és adatobjektumok közötti kapcsolatok reprezentálására is alkalmasak. Igy ebben a modellben direkt módszer van a több-komponensű kapcsolatok ábrázolására, és az un. relációs predikátum kalkulus [ULL'82, 156-168.old.] egy természetes és igen kifejező eszköz a lekérdező műveletek felépitésére. Azon-

ban ez a modell sem kinál segitséget a modellezési folyamat végrehajtásához, és itt is bizonyos mûveletek (pl. az egyesitès, vagy a normalizáció) a számitógép jellegébôl ered, és nem a természetes fogalmakból, igy a relációs adatmodell - leglábbis eredeti formájában - sem elegendô a fogalmi modellezéshez.

- Az absztrakció támogatása

Ez röviden és egyszerûsitve azt jelenti, hogy a modellnek támogatnia kell az információ megszervezését a modellezési folyamat során. Ennek egyik fontos eleme az, hogy legyen mód az éppen aktuális problémához fontos részletek hangsúlyozására, és az éppen jelentéktelen részletek elrejtésére. (Például lehetôség legyen a modellt lépésrôl-lépésre fölépiteni, elôször az alapvetô, meghatározó fogalmakat megragadva, azután egyre több részletet leirva. Ez megfelel a strukturált programozás un. fentrôl-lefelé tervezési elvének [WI'71])

Vagy vegyük az adatobjektumok közötti kapcsolatok leirását. A klasszikus adatmodellek közül egyedül a relációs adatmodellben találunk közvetlen módszert a több paraméteres kapcsolatok leirására, de ott sincs semmi biztositék arra, hogy alkalmazás-orientáltan támogassák a kapcsolatok ábrázolását, és biztositsák azt, hogy értelmesek legyenek.

- Természetes objektum- és művelet fogalmak

Minden adatmodelltől elvárjuk, hogy kifejező legyen, azaz a modellre támaszkodva a felhasználó képes legyen az összes fontos állandó, vagy dinamikus tulajdonságot leírni. Azonban az is igen fontos, hogy ez egyszerű és célirányos módon történjék, azaz az alkalmazás egy természetes fogalma megjelenjék, mint a modell egy fogalma, és ne kelljen bajlódni azzal, hogy a modell több fogalmából kelljen a modellen kívül összekombinálni.

Másrészt lehetőleg el kell kerülni a mesterséges, erőltetett fogalmakat a modellben, azaz az olyanokat, amelyek csak a modell megszervezéséhez, felépítéséhez kellenek, de nincs direkt jelentésük. (Erre példa az, hogy ha a hierarchikus vagy hálózatos modellben a szokásos módon ábrázolunk egy több paraméteres kapcsolatot, akkor létrehozunk egy fiktív "adatobjektum típust", amely csak a kapcsolat egy eleme komponenseinek összefogására szolgál.

Fontos észrevennünk, hogy bizonyos szempontok nem játszottak szerepet a szemantikus adatmodellek létrehozásánál. Először is, általában nem törődünk a modell nem redundáns voltával. (Azaz azzal, hogy egy elemi információ csak egyszer szerepeljen a konkrét adatbázis példányban.) Ez valószínűleg távol áll az emberi gondolkodástól, és a szemantikus adatmodellek megalkotóinak többsége elvetette a természetesebb ábrázolásmód kedvéért. (A számítógépek fizikai

paramétereinek látványos fejlődése valószínűleg ezt indokolttá is teszi.)

Másrészt ez a téma napjainkban inkább még a kutat szintjén van, és a javasolt - igen nagyszámú - szemant kus adatmodell túlnyomó többsége még csak elképzelés Így aztán az implementálhatóság nehézsége eddig ne játszott nagy szerepet ezen modellek létrehozásánál, sőt nagyon kevés modellt vizsgáltak meg ebből a szempontból. A közeljövőben ez a helyzet nyilván látványosan változni fog, és egy modell implementálhatósága döntően meg fogja határozni a modell jogosultságát.

2. Szemantikus adatmodellek áttekintése

2.1 A legfontosabb elem: az absztrakció

2.1.1 Az absztrakció meghatározása

Az alábbiakban megkíséreljük a szemantikus adatmodellek részletesebb leirását. Ez nem könnyű feladat, mivel - mint azt már előzőleg is emlitettük - az elmúlt pár évben igen nagyszámú javaslat született. Ezek nagyrésze azonban csak egy ötletre épül, és a részletesebb kifejtés hiányzik, és egyébként is ezen modellekben igen sok hasonló gondolat van. Igy aztán az alapvető eszméket sikerül majd fölsorolnunk.

Véleményünk szerint a jelenlegi modellek legfontosabb eleme az absztrakció támogatása, igy azt most kiemelve tárgyaljuk. Maga az absztrakciós folyamat egyszerűen azt jelenti, hogy az éppen aktuális nézőpont szerint fontos részleteket hangsúlyozzuk, a többi részletet pedig elfedjük.

2.1.2 Az osztály fogalma

A legtöbb szemantikus adatmodell az absztrakciós folyamat támogatására az un. "osztály" fogalmát használja fel, és néhány alapvető konstrukciót, amelyekkel a már létező osztályokból új osztályokat hozhatunk létre. Ezen elmélet tulajdonképpen két régebbi elképzelés egyesitéséből született.

2.1.2.1 Előzmények

Az első a programozási nyelvekből származik, és az un. absztrakt adattipus fogalma. Ez a hetvenes évek elején, közepén kristályosodott ki, és a fogalom megalkotásában a Simula programozási nyelv "osztály" fogalmának nagy jelentősége volt [DMN'68].

Ez az absztrakt adattipus kifejezés azt jelenti, hogy az illető programozási nyelvben módunkban áll új tipusokat létrehoznunk, mégpedig a következő módon: A program deklarációs részében definiáljuk a tipust úgy, hogy megadjuk az azonositásra szolgáló nevét, és azt, hogy milyen alkotórészekből milyen módon épül fel, és még azt is, hogy milyen műveleteket lehet végrehajtani rajta. (A fizikai megvalósitás megszervezése természetesen a fordítóprogram feladata és nem a programot iró személyé.) Ezután magában a programban ezt a tipust ugyanúgy használhatjuk, mint a szokásos (egész, valós stb.) tipusokat, pl. definiálhatunk ilyen tipusu változókat, amelyekkel egyszerű név szerinti hivatkozással végrehajthatjuk a műveleteket.

A második befolyásoló fogalom a mesterséges intelligencia kutatásból származik, nevezetesen a szemantikus hálók fogalmából, igy most nem részletezzük.

2.1.2.2 A fogalom meghatározása

Most pedig lássuk az "osztály" fogalmát közelebbről, ahogyan a legtöbb szemantikus adatmodell használja.

Tekintsük az adatbázis összes lehetséges példányát. Ezek elemei együtt alkotnak egy halmazt, amelyet az illető adatmodell adatobjektumai halmazának nevezünk.

Ezen objektumok bizonyos csoportjainak lehetnek olyan közös tulajdonságai, amelyek miatt szeretnénk őket együttesen is egy fogalomnak tekinteni. Ezt úgy tehetjük meg, hogy a sémában definiálunk egy "osztályt", amelynek elemei éppen a kérdéses objektumok.

Ezzel máris meghatároztuk az adat absztrakció első fajtáját:

Osztályozás:

> Ez azt jelenti, hogy definiálunk egy osztályt (ami egy objektum-tipus), mint az objektumok egy halmazát. Így ez a művelet hozza létre a kapcsolatot a séma és az adatobjektumok között, amely kapcsolatra az "előfordulása" elnevezést fogjuk használni. (Ez a szakirodalomban meghonosodott "instance-of" megnevezés fordítása.)
>
> Például, ha adatmodellünk személyek adatainak nyilvántartásával foglalkozik, akkor nyilván lesz egy "személy" osztályunk. Ha az éppen létező aktuális adatbázis példányban szerepelnek "Kovács János" adatai, akkor az egyén adatait tartalmazó objektum

egy "előfordulása" a "személy" osztálynak.

Tehát az előzőeknek megfelelően az "osztály" jelentése kettős: egyrészt jelenti az osztály elemeiből álló adatobjektum halmazt (ez a modell szintjén a jelentése), másrészt jelenti azt a sémabeli specifikációt, amellyel leirjuk és meghatározzuk az előező objektumokat (ez a séma szintjén a jelentése).

2.1.3 Az absztrakciós konstrukciók áttekintése

A következőkben azon műveletekkel foglalkozunk, amelyekkel a sémát fölépithetjük, azaz azokkal a konstrukciókkal, amelyekkel a már létező osztályspecifikációkból új osztály leirásokat készithetünk. (Ez fölveti annak a kérdését, hogy ezt az épitkezést hogyanis tudjuk elkezdeni, azaz mik azok az osztályok, amelyeket kezdetben adottnak tekinthetünk. Általában az a megoldás, hogy a sémát készitő személy megadhat néhány elemi osztályt, amelyek jelentése világosnak tekintett az adott alkalmazási környezetben - lásd pl. az úgynevezett SDM modell "alaposztály" fogalmát [HM'81]. Van néhány rendszer, ahol az un. specializáció művelet - amelyről később lesz szó - olyan erős és kifejező, hogy kezdetben csak egy osztályt tekintünk adottnak, ami az összes szóbajövő adatobjektum halmaza, amit hivhatunk pl. a "DOLGOK" osztálynak (lásd a TAXIS modellt [MBW'81].))

2.1.3.1 Összekapcsolás

Az első és legegyszerűbb ilyen művelet a fix formátumú összetett adatobjektumok létrehozására szolgál. (Így ez a művelet szoros rokonságban van a programozási nyelvek rekordot létrehozó parancsával, pl. Pascal-ban a "record", vagy C-ben a "struct" konstrukcióval.)

Összekapcsolás (= aggregation)

Ez a sémabeli megoldásban azt jelenti, hogy osztályok egy (véges) gyűjteményét fölsoroljuk, és ennek a listának nevet adva a továbbiakban már osztálynak tekintjük.

Az adatbázis szintjén ennek az összetett osztálynak az elemeit pedig egyszerűen az alkotó osztályok elemeiből képzett rendezett listák alkotják, azaz ennek az összetett osztálynak az objektum-halmaza az alkotó osztályai objektumhalmazainak Descartes-féle szorzata.

Egy alkotó osztály és az összetett osztály közötti kapcsolatot a továbbiakban "alkotórésze" kapcsolatnak fogunk hívni. (Ez az angol "part-of" elnevezés fordítása.)

Például, ha az előzőleg említett személyi nyilvántartásban csak az emberek nevét és címét tároljuk, akkor a "személy" osztály lehet összekapcsolása a "név" és a "cim" osztálynak, ahol a "név" pl. lehet egy specializációja a "betűsorozat" alaposztálynak, a "cim" pedig lehet az "irányítószám", "város",

"utcanév", "házszám" osztályok összekapcsolása, amely osztályok egy-egy specializációját alkotják az "egész", "betűsorozat", "betűsorozat", illetve "egész" alaposztályoknak.

2.1.3.2 Specializáció / általánositás

Ezután lássuk azt a műveletet, amely a jelenlegi modellekben leginkább támogatja az absztrakciót. (Igy természetesen ezen művelet pontos meghatározásában van a legnagyobb különbség az egyes rendszerek között.) Ezen művelet segitségével fel tudjuk épiteni bővebb illetve szűkebb osztályok egy hierarchiáját, amely hierarchia - a feltételeket jól megválasztva - tükrözni fogja a modellezendő folyamat természetes fogalmait. A hierarchia felsőbb elemei - azaz a bővebb osztályok - fogják jelképezni az általánosabb fogalmakat, mig a hierarchia alsó elemei fogják ábrázolni a több részletet leiró, specifikus fogalmakat.

2.1.3.2.1 A művelet meghatározása

Specializáció / általánositás

Természetesen ezek a műveletek is a sémán belüli épitkezésre szolgálnak, azaz egy vagy több adott osztály leirásából megragadjuk a közös jellemzőket, de bizonyos részleteket elhanyagolunk, és igy létrehozunk egy általánosabb osztály leirást.

Általában ennek a műveletnek a forditottja játszik

nagyobb szerepet, amit specializációnak nevezünk. Tehát a specializáció azt jelenti, hogy kiindulva egy vagy több osztály leirásból, újabb részleteket adunk azokhoz, és igy létrehoznak egy új, részletesebb osztályt, amelyet a kiinduló osztályok egy specializáltjának nevezünk.

Az előzőekből következően, ha tekintünk egy specializációt, akkor a specializált osztály meghatározása formálisan többet követelő, mint az eredeti osztályok bármelyikének leirása, és igy - az adatbázis szintjén - ennek kevesebb objektum tesz eleget, azaz a specializált osztály objektum-halmaza szűkebb, mint az eredeti osztályok objektumhalmazai.

A szakirodalomban meghonosodott kifejezéssel az általánosabb leirású osztályt a szűkebb leirású osztály egy szuperosztályának nevezzük, és a közöttük fennálló kapcsolatot - forditott irányban - a korábbiaknak megfelelően IS-A kapcsolatnak hivjuk. (Ez a bevett angol kifejezés.) Ha tehát a "személy" osztálynak a "nő" osztály egy specializáltja, akkor? Egy "nő" egyben IS-A "személy".

A különböző modellekben számos módot találunk arra, hogy pontosan hogyan is hajthatjuk végre ezt a műveletet. A régebbi rendszerekben, pl. az úgynevezett RM/T rendszerben [CO'79] a specializált osztály képzéséről csak egy eredeti, általános osztályból indulhatunk ki, és az a módszer, hogy pl. egy attributum értékét rögzithetjük. Ez több szempontból nem látszik elegendőnek. (Például

vegyük az első megkötést. Ha tekintünk egy egyetemi személyi nyilvántartást, akkor lesz egy "személy" osztályunk, amely tartalmazza az összes számbaveendő egyént. Ennek az osztálynak a "férfi", illetve "nő" egy-egy természetes specializáltja, de ugyanúgy természetes specializáltak a "tanár" illetve "diák" osztályok is. Ha ezután a férfi diákokat le szeretnénk irni egy osztályba, akkor erre a kézenfekvő mód az, hogy vesszük a "férfi" és a "diák" osztályok közös specializáltját.) Igy azután az ujabb, nagyobb erejű rendszerek (elsősorban a TAXIS modell) megengedik, hogy több általánosabb osztályból induljunk ki, és a specializáció megadására tetszőleges, az eredeti osztályok attributumaiból felépithető predikátum szolgál.

2.1.3.2.2 Tulajdonságok öröklődése

A fentiekben vázolt hierarchia egyik legfontosabb előnye az, hogy a specializált osztályok öröklik az általánosabb osztály leirását, és igy nincs szükségünk az általános részleteket újból felsorolnunk. Ez egyrészt közelebb áll a természetes fogalomalkotáshoz, másrészt bizonyos szintaktikus hibák lehetőségét csökkenti. (Ha egy osztály lehet több általános osztály specializáltja, akkor ez az osztály az összes általánosabb osztály leirását örökli, ez az un. többszörös öröklés. Ez jelenthet zavart az egyes tulajdonságok elnevezésében, ha egy attributum nevet több általános osztály leirásában használtunk. Igy ebben az esetben a rendszernek rá kell kényszeritenie a felhasználót egy

elnevezés konvenció használatára.)

Egyes rendszerek között eltérések vannak abban, hogy a tulajdonságok öröklődése pontosan hogyan is történjék. Először is, mivel a specializált osztály részletesebb leírásra szolgál, így természetes, hogy bevezethetünk új tulajdonságokat, és valóban ezt az összes modell támogatja. Abban azonban már eltérések vannak, hogy a régi tulajdonságok hogyan is kerüljenek át a specializált osztály leirásába. A Simula nyelv az úgynevezett szigorú öröklést kinálja, ami azt jelenti, hogy a régi, általános tulajdonságok szóról-szóra megjelennek az új leirásban. A TAXIS rendszer viszont lehetőséget ad arra, hogy ezeket a tulajdonságokat átdefiniáljuk, azaz egy-egy általános tulajdonságot valamely specializáltjával helyettesitsük. (Igy természetesen az igaz marad, hogy a specializált osztály egy példánya egyben legális eleme lesz az általános osztálynak is.)

Meg kell jegyeznünk, hogy noha a szemantikus adatmodellek túlnyomó többsége a fentiekben vázolt módját használja a specializációnak, van egy másik kinálkozó lehetőség is, amelyet elsősorban a mesterséges intelligencia kutatásban használunk, ahogyan a tanulmány első felében láttuk. Arról van szó, hogy az általunk megadott követelmények szerint, ha az A osztály egy specializáltja a B osztálynak, akkor az A osztály minden elemének eleget kell tennie a B meghatározásának. Ez a matematikai gondolkodásban mindig igy is van, és az osztályok hierarchiáját világossá teszi, azonban az emberi gondolkodásban ez nem feltétlenül teljesül. Például a köznapi nyelvben az alábbi két állitást igaznak tartjuk (ez a példa azonos a tanulmány első

felében lévővel, ahol vele szintén az IS-A hierarchia egyértelmű értelmezésének nehézségeit ábrázoltuk):

"A madarak repülnek."

"A struccok madarak."

Az általunk megadott szigorú öröklődési szabályból ekkor az következne, hogy a struccok tudnak repülni, ami egyszerűen nem igaz.

Ennek a problémának két megoldása lehet. Vagy feladjuk a szigorú öröklődést, és a specializációnál az általános tulajdonságok csak akkor öröklődnek a specializált osztályra, ha felül nem definiáljuk azokat (ez a bevett angol kifejezéssel a "default inheritance"), vagy megtartjuk a szigorú öröklést, és a fentihez hasonló eseteket úgynevezett kivételnek tartjuk, és külön kivétel-kezelő eljárásokat vezetünk be. Az előbbit a mesterséges intelligencia kutatásban használjuk gyakran, bár van néhány konkrét rendszer is, amely erre épül. (Például a Smalltalk programozási nyelv.) Az utóbbi, amelyet az előzőekben részletesen ismertettünk, található meg a legtöbb szemantikus adatmodellben. (Lásd a későbbiekben a TAXIS rendszer részletes leirását, különösen a kivétel-kezelő eljárásokat.)

2.1.3.2.3 A megvalósitás nehézsége

Nem árt felhivnunk a figyelmet arra, hogy egy IS-A kapcsolatokból felépülő osztály/hierarchia megvalósitása milyen komoly implementációs nehézségeket jelent.

Például, ha van egy "személy" nevű osztályunk, amelynek van egy "életkor" nevű attributuma, amelynek értéke lehet egy nemnegatív egész, és van továbbá ennek az osztálynak egy "gyermek" nevű specializáltja úgy, hogy azt az osztályt azon elemek alkotják a "személy" osztályból, akiknek az életkora 0 és 14 közé esik, akkor, ha létrehozunk egy "személy" osztályba tartozó objektumot, akkor ha ennek életkora pl. 11 év, akkor ezt az objektumot automatikusan el kell helyeznünk a gyermek osztályba is. A másik oldalról pedig, ha létrehozzuk a "gyermek" osztálynak egy új elemét, akkor ezt be kell illesztenünk a "személy" osztályba is.

2.1.4 Asszociáció: egy nem általánosan elfogadott absztrakciós művelet

Az előzőekben megadott absztrakciós módszerek mind megtalálhatók a szemantikus adatmodellek közül azokban, amelyek ezzel a terminológiával támogatják az absztrakciós folyamatot. Így ezeket tekinthetjük a standard absztrakciós lépéseknek, mivel ezek az elnevezések és jelentésük is többé-kevésbé egységes a szakirodalomban. Ezen konstrukciók a legtöbb modellezési feladatra elégségesek, de nem az összesre. Tekintsük pl. a következő kérdést: Ismét egy személyi nyilvántartássunk van, amelyben a "személy" nevű osztálynak van egy "kereset" attributuma. Ha érdekel minket a vállalaton belüli átlagkereset, akkor bevezethetünk egy "átlagkereset" nevű attributumot, azonban ez nem az egyes személyek tulajdonsága lesz, hanem az egész "személy" osztálynak. Ez mutatja, hogy bizonyos feladatok azt kívánják, hogy bizonyos osztályoknak legyenek tulajdonságaik, amelyek

nincsenek meg az őket alkotó elemekben. Ez úgy érhető el, hogy bevezetünk egy olyan osztályt, amelynek elemei maguk is osztályok. Ez tulajdonképpen azt jelenti, hogy az "osztályozás" műveletet végrehajtjuk az osztályokra is, mint elemekre, és igy - a szokásos terminológiát használva - ugynevezett metaosztályokat kapunk. Ezeknek a metaosztályoknak megvannak a maguk meghatározásaik, és igy pl. lehetnek tulajdonságaik, amelyek újak, azaz az őket alkotó osztályok alkotóelemeiben nem találhatunk meg. (Az előző példánál maradva, az "átlagkereset" lehet egy attributuma bármely személyekből álló osztálynak, de az egyes személyeknek nincs "átlagkereset" nevű tulajdonságuk.) Mivel a metaosztályok maguk is osztályok, igy az előzőekben vázolt absztrakciós lépések rájuk is vonatkoznak, igy pl. a metaosztályok is egy "egy" hierarchiába szervezhetők.

Egyes szerzők [BR'84] bevezették az osztályok közötti osztályozásra az "asszociáció" elnevezést, ez azonban még nem nyert polgárjogot a szakirodalomban.

Jelenleg általánosan elfogadott, hogy az absztrakciónak ez a két szintje elégséges a legtöbb alkalmazáshoz, és nincs szükség arra, hogy a metaosztályokból bizonyos "meta-metaosztályokat" épitsünk fel. [MBW'80, TL'82 - 10.fejezet] A TAXIS rendszer szigorúsága erre az álláspontra épül, és közvetlenül támogatja a kétszintű hierarchia felépitéséhez szükséges lépéseket.

2.2 Fejlesztési lehetőségek, kutatási irányok

2.2.1 A szokásos matematikai formalizmus

A következőkben megkíséreljük röviden áttekinteni azokat az irányzatokat, amelyeket a jövőbeli, nagyobb erejű szemantikus adatmodelleknek támogatniuk kell. Ebben a problémakörben sok a nyitott kérdés, és így ennek a területnek a kutatása igen indokolt.

Elsősorban a Buroughs cég kutatóközpontjának eredményeire fogunk támaszkodni [ST'85].

Először is a szükséges matematikai alapokat kell tisztázni. Az adatmodellek felépítésében eddig általában a természetes predikátum kalkulusra támaszkodtunk, azaz a matematikai logika nyelvén egy jól megválasztott elsőrendű nyelv néhány formulája jelenti az adatbázis sémájának megadását, mégpedig a következő módon: A változók és a konstansok adatobjektumokat, az egyes predikátumok (azaz a relációjelek a nyelvben) pedig az objektumok közötti kapcsolatokat jelölik a lehető legtermészetesebb módon, azaz ha az R relációjel jeleniti meg a K kapcsolatot, akkor itt az R és K aritása (változóinak száma) egyenlő, mondjuk n. Ha pedig az x_1, x_2, ..., x_n változók, vagy konstansok az obj 1, ..., obj n objektumokat ábrázolják, akkor az $R(x_1, ..., x_n)$ igazság-értéke pontosan akkor igaz, ha az obj 1, ..., obj n objektumok között a K kapcsolat fennáll (például [CO'79], 3.pont).

2.2.2 A standard matematikai logika kiterjesztései

Ez a formalizmus természetes és igen kifejező, azonban van két pont, ahol mégis a kifejező ereje nem elégséges.

Először is, mint az előzőekben tárgyaltuk, indokoltnak látszik az un. metaosztályokat is bevezetni. Ez a megfelelő nyelvben azt jelenti, hogy bizonyos változóknak és konstansoknak halmazt kell jelölniük, és igy pl. jogunkban áll az egzisztenciális és univerzális kvantort is használni ezekre a változókra. Igy pl., ha minden személyekből álló osztálynak van egy főnöke, és megállapitottuk, hogy egy főnök mindig legalább kétszer annyit keres, mint az osztályának átlagfizetése, akkor ezt a tényt kifejezhetjük a következő formában:

$\forall$ személy(x) (kereset (főnök(x)) > 2* átlagkereset(x))

Az ilyen nyelveket a matematikai logikában másodrendű nyelveknek hivjuk, tehát elvárjuk, hogy a sémát felújitó nyelv (az ugynevezett objektum nyelv) egy legalább másodrendű nyelv legyen.

Másodszor: a klasszikus formalizmus egy állitás igazságértékére két lehetőséget kinál, egy állitás vagy igaz, vagy nem. Ha például tekintjük az "anyjának lenni" kapcsolatot a személyek között, akkor ezt ábrázolhatjuk egy kétváltozós "anyja(x,y) predikátummal, ahol az x,y személy tipusú adatobjektumokat jelölnek, és ez a predikátum pontosan akkor igaz, ha az x-nek megfelelő személy édesanyja az y-nak megfelelő személynek. Ha ezt a kapcsolatot pl. egy táblázatban tároljuk, akkor az "anyja(x,y)" igazságértékének megállapitása úgy

történik, hogy végignézzük a táblázatot, és ha megtaláljuk benne az x,y bejegyzést, akkor ez a predikátum igaz, egyébként hamis. Ilyen módon minden állításról eldönthetjük, hogy igaz, vagy hamis, ezt a szakirodalomban az un. "zárt világ feltevésnek" nevezik [CO'79, 3.pont]. Ezt elôször R. Reiter [RE'78] fogalmazta meg, és ez elmúlt években ennek a feltevésnek jogosultsága intenziv kutatás tárgya volt (például [LE'84]). Tény, hogy a jelenleg kereskedelmi forgalomban lévô rendszerek mind erre a feltevésre épülnek, és ezt a feltevést használva a lekérdezéseket világosan és elegánsan meg tudjuk fogalmazni, azonban valószinû, hogy egy erôs szemantikus modellben nem tarthatjuk meg ezt az egyszerû formalizmust. Ennek oka az, hogy a valódi életben tudásunk hiányos, és kivánatos, hogy ez a modellben is tükrözôdjék. Igy egy szemantikus adatmodelltôl elvárhatjuk, hogy három igazságértéket támogasson, az "igaz", "hamis", "nem ismert"-et. Természetesen ennek megvalósitásához meg kellenek, hogy legyenek a megfelelô eszközök. R. A. Stachowitz szerint erre alkalmas mód egy "meta-nyelv" bevezetése, ahol bizonyos objektum-nyelvbeli állitásokat expliciten kijelenthetünk illetve cáfolhatunk. Igy egy objektum-nyelvbeli formula igazságértéke akkor "igaz" vagy "hamis", ha ezt le tudjuk vezetni a meta-nyelvben expliciten kijelentett vagy tagadott állitásokból, egyébként pedig "nem ismert". Ez az un. nyitott világ feltevés", és valószinû, hogy a jövôbeli nagy erejû modelleknek erre kell épülniük.

2.2.3 Nem létező és nem ismert objektumok

A következő szempont szintén az esetleg hiányos tudásunk ábrázolásának igényéből fakad. Az adatkezelésben közismert probléma az, hogy egy objektum létrehozásának pillanatában nem ismerjük minden tulajdonságát, azaz ennek az objektumnak vannak olyan attributumai, amelyeknek értékét aktuálisan nem ismerjük. Ezt általában úgy oldjuk meg, hogy valamilyen módszerrel az attributum értékének helyénél jelezzük, hogy egyelőre nincs adatunk. Erre a szokásos eljárás az, hogy engedélyezünk speciális "null" értékeket, és ha valamely attributum értékét nem ismerjük, akkor neki ezt a különleges "null" értéket adjuk. Ez az eljárás intenziv kutatás tárgya volt az elmúlt években (például [RE'84]) annak érdekében, hogy a "null" értékek szerepét tisztázzuk a matematikai megalapozásban. Részletekbe nincs módunk belemenni, azonban mutatunk egy példát, ami megvilágitja a nehézségek egy részét.

Vegyünk egy adatobjektum tipust és annak két példányát úgy, hogy ezen két objektum minden attributum értéke rendre megegyezik. Ebben az esetben eddig ezt a két objektumot egyenlőnek tartottuk, ha azonban a "null" értéket is megengedjük attributum-értéknek, akkor ez nyilván nem fog teljesülni. Ha például két "személy" tipusba tartozó objektum neve egyelőre nem ismert, akkor mindkét objektum "név" attributuma "null" lesz, és előfordulhat, hogy az összes többi attributum-értékük is egyenlő. Ebből azonban nem következik, hogy a két objektum egyenlő, hiszen lehet két személy, akiknek a nevén kivül minden tárolt adata (pl. születési év, lakóhely) azonos. Ez a példa könnyen elháritható

látszik, azonban a "null" értékekkel végre kell tudnunk hajtani az összes adatmanipulációs és lekérdező műveletet, és ez komoly nehézségeket is okozhat (újra [RE'84]).

Így egy szemantikus adatmodelltől elvárjuk, hogy támogassa a "null" (azaz nem létező) objektumok kezelését. Ehhez hasonlók az R. A. Stachowitz által bevezetett "nem ismert" objektumok, amelyek egy adott objektum tipus egy maghatározott elemét jelölik, azonban amelyek azonositásához egyelőre nincs elegendő információ. Az általa adott példa egy rendőrségi nyilvántartás, amelyben van egy "tettesként szóba jövő" tipus, amely a személyekre vonatkozó szokásos adatokon kivül a személyleirást is tartalmazza, és ebbe az osztályba bevezetik mindazok adatait, akik valamilyen oknál fogva (például büntetett előélet) számitásba veendők, mint bünelkövetők. Minden pillanatban a rendszerben van néhány ismeretlen bünelkövető, akik azonositása a nyilvántartás fő feladata. Ezen - egyelőre ismeretlen - személyeket jelölhetjük a "nem ismert" objektumokkal, úgymint pl. u1, u2, ... stb. Itt az i-edik ismeretlen tettes minden ismert adatát (pl. szeme szine, körülbelüli testsúly) bevezetjük az ui leirásába, és igy ez a leirás szolgál az információ összegyüjtésére. Ha befut néhány új adat ui-re vonatkozólag, akkor az adatok bevitele után egy lekérdező művelettel megkisérelj̈ük az azonositást. Ha sikerül, akkor megtesszük a szükséges lépéseket, és utána ui-t töröljük a "nem ismert" objektumok listájáról.

Ezen "nem ismert" objektumok egy másik kérdésben is segitségünkre lehetnek, amit a szakirodalomban "unió-

információnak" (disjunctive information) neveznek. Ez is bizonyos hiányos tudásból fakad, ugyanis előfordul, hogy két állitás logikai uniójáról tudjuk, hogy igaz, de a két állitásról külön-külön nem. Például előfordulhat, hogy egy adott György nevű gyerek matematika tanáráról azt tudjuk, hogy vagy János, vagy Péter, de hogy pontosan ki, azt nem. Ennek az információnak megjelenitése szintén gondot okoz a klasszikus formalizmusban, mivel a két állitásnak: "János a tanár", "Péter a tanár" külön-külön meghatározott igazságértéket kell adni, de csak azt tudjuk, hogy ezen két igazságérték kizáró logikai egyesitése igaz. (Feltesszük, hogy egy tanárról van szó, azaz a diáknak nincs két matematika tanára.)

Egy megoldás az, hogy eset szétválasztással külön-külön modellben megvizsgáljuk a szóbajövő "i", "h" illetve "h", "i" igazságérték párokat, azonban a komolyabb feladatoknál óhatatlanul hatalmas mennyiségű munkát jelent, az esetek számának kombinatorikai robbanása miatt.

A "nem ismert" objektumok megoldhatják ezt a kérdést is, feltéve, hogy van a meta-nyelvben két állitásunk, az "összehasonlitható", illetve az "összehasonlithatatlan", amelyek közül az első azt fejezi ki, hogy az első paraméterként adott "nem ismert" objektum valódi értéke lehet a másodikként megadott konkrét objektum, a második művelet pedig azt, hogy nem. A következő meta-nyelvi állitássorozattal leirhatjuk a fenti únió-információt. (Itt u egy eddig nem használt "nem ismert" személy objektum, amely igy kezdetben minden személlyel összehasonlitható.)

∀ személy(x) (összehasonlithatatlan(u,x))

u = tanára (György)

összehasonlitható(u, János)

összehasonlitható(u, Péter)

Talán az előzőek is már megmutatták, hogy bizonyos szituációkban a "nem ismert" objektumok alaposan leegyszerűsitik a rendszer felépitését, és igy ezen objektumok kezelésének megadását is számba kell vennünk a modern szemantikus adatmodellek megalkotásánál.

2.2.4 Következtetési szabályok beépitése

A következőkben talán a legfontosabb új elemet tekintjük át, amely a modellek egész szerkezetét átalakithatja. Arról van szó, hogy a tanulmány II.pontjában megismert tudásábrázolási módszereket szeretnénk beépiteni az adatmodellbe, mégpedig olymódon, hogy a modellre épülő rendszer képes legyen bizonyos következtetéseket levonni, azaz képes legyen olyan információkat levezetni, amelyek nincsenek expliciten tárolva az adatbázisban.

Jól tudjuk, hogy ennek megvalósitása nem könnyű kérdés, és a tudásábrázoló rendszerekben az is probléma volt, hogy ott általában nem tételezhetünk fel "apriori", mindenképp igaz következtetési szabályokat, amelyek fennállását garantáltnak vehetjük. A szemantikus adatmodelleknél viszont megelégszünk azzal, hogy olyan

módszereket kínálunk, amelyekkel legalább a modellezési folyamatban felismert, biztosan igaz szabályokat beépithetjük a modellbe, és nem törődünk azzal, hogy ezeket hogyan is lehet megtalálni csupán néhány konkrét szituáció (azaz itt adatbázis példány) ismeretében. Igy ezen szabályok megtalálását teljes mértékben a modellezést végző személyre háritjuk.

Az előzőekben említett "meta-nyelv" erre is egy kézenfekvő lehetőséget kinál, hiszen abban expliciten kijelenthetjük egy következtetési osztályról, hogy igaz. Ezzel például bizonyos megszoritásokat könnyen le tudunk irni:

$$\forall x \ (\text{személy}(x) \quad (0 < \text{életkor}\ (x) < 120))$$

Stachowitz említett cikkében részletesen tárgyalja, hogy pontosan milyen következtetési szabályokat tart célszerűnek beépiteni a modellbe. Az általános okok:

"feltétel" "következtetési szabály" "következmény"

Itt természetesen az objektumokból és tulajdonságaikból felépitett klasszikus szillogizmusok a legkézenfekvőbb szabályok, ahol a fenti "következtetési szabály" a "logikai következtetés", például

$$\forall x \ (\text{személy}(x) \Rightarrow \text{halandó}(x))$$

Itt nincs helyünk a részletesebb elemzésbe belemenni, röviden csak azt állapithatjuk meg, hogy az objektumokra, osztályokra, illetve metaosztályokra vonatkozó összes szóbajövő, formailag helyes

következtetési szabály hasznosnak bizonyulhat, és igy érdemes számba venni.

Fontos megjegyeznünk, hogy ha a modellünkbe beletartozik egy, a fentiekben vázolt következtető egység, akkor igen szigorúan meg kell akadályoznunk azt, hogy ellentmondó adatok kerüljenek az adatbázisba. A logikában egy ellentmondásból minden állitás következik, igy ellentmondó adatok esetén a következtető egység összeomlasztja az adatbázist.

Annak, hogy a modellünkben meg lehet ilyen következtetési szabályokat adni, sok kézenfekvő előnye van. Ezek közül a legfontosabb az, hogy előfordulhat, hogy bizonyos adataink közvetlenül következtetési szabályként fogalmazhatók meg, és ennek természetes megjelenitése az, hogy egy az egyben beépitjük a modellbe. Másrészt ügyesen megválasztott szabályokkal jelentősen csökkenthetjük a fizikailag tárolt adatok mennyiségét, és gyors, kifejező lekérdező műveletekre nyilik lehetőség.

2.3 Konkrét szemantikus adatmodellek általános osztályozása

A következőkben megkiséreljük röviden áttekinteni a szemantikus adatmodelleket. Ez a modellek nagy száma miatt nem könnyű feladat, elsősorban M. L. Brodie tanulmányára fogunk támaszkodni.

A máig kifejlesztett modellek négy, világosan meghatározott csoportba oszthatók:

- a klasszikus modellek direkt kiterjesztései

- matematikai modellek

- irreducibilis adatmodellek

- szemantikus hálózati modellek.

2.3.1 Klasszikus modellek közvetlen kiterjesztései

2.3.1.1 Az entity relationship modell

Ez a modell [CH'76] a legjobban elismert továbbfejlesztése a klasszikus modelleknek, és ezen modell terminológiája többé-kevésbé szabványosnak tekinthető a szakirodalomban, és mi is ezt használtuk az 1.pontban, ahol az adatmodellek általános jellemzőit ismertettük.

Tehát ennek a modellnek a jelentősége óriási, azonban mivel a számítógéptudományban általánosan ismert, igy csak vázlatosan ismertetjük.

Ebben a modellben két tipusu információhordozó van, egyrészt az entity-k, másrészt az ezek között fennálló kapcsolatok.

Magát az entity fogalmát nem definiáljuk (hasonlóan a matematikai halmaz fogalmához), ez egy mindent magába foglaló elnevezés: minden, aminek objektiv léte van, vagy képesek vagyunk rá világosan gondolni, tekinthető egy entity-nek. Modellünkben az adathordozó objektumok

az entity-k lesznek.

Bizonyos entity-ket hasonlóknak tartunk, ezt úgy tudjuk kifejezni, hogy azonos tipusba tartoznak. Itt az entity tipust inkább entity halmaznak nevezzük.

Ezen entity halmazoknak nem kell diszjunktoknak lenniük, azaz egy adott entity halmazok között fennállhat az általánositás/specializáció kapcsolat, tehát lehet, hogy egy speciálisabb enetity-halmaz része az általánosabb entity-halmaznak.

Az entity-k tulajdonságainak leirására az attributumok szolgálnak, itt egy attributum egy tulajdonságot jelöl. Egy attributum egy függvény egy entity-halmaz és a szóbajövő attributum értékek halmaza között, ahol az utóbbit az illető attributumhoz tartozó érték-halmaznak nevezzük. Bizonyos attributumok lehetnek többértékűek, azaz az illető érték-halmaz egy részhalmazát jelölik ki. Természetesen előfordulhat, hogy különböző attributumokhoz ugyanaz az értékhalmaz tartozik, várható, hogy az "egészek" érték-halmaz több, numerikus jellegű attributumhoz fog tartozni.

Az entity-k közötti kapcsolatok leirására az entity-halmazok közötti relációk szolgálnak, azaz ebben a modellben egy kapcsolat egy névvel kitüntetett matematikai reláció bizonyos entity-halmazok között, amelynek egy eleme (azaz az entity-k egy n-ese) a kapcsolat egy eleme, vagy megvalósulása. A relációk teljesen általánosak lehetnek, tehát lehetnek több mint kétváltozósak, és a hierarchikus és hálózatos adatmodellekre jellemző megszoritásokat (azaz, hogy egy

kapcsolat egy rögzitett komponense csak egyértelmű lehet) sem tételezzük fel.

A kapcsolat megadásánál lehetőségünk van arra, hogy jelezzük az adott reláció adott komponensében a megengedett entity-k maximális számát, és igy módunkban áll az alkalmazás szempontjából fontos, speciális eseteket megjelölnünk, pl. az előző esetet, amikor az egyik komponensnek egyértelműnek kell lennie.

Az alkotó entity-knek egy reláción belül lehet egy bizonyos szerepük, pl. a "főnök-beosztott" kapcsolat egy reláció a "személy" és újra a "személy entity-halmaz között, ahol az első komponens szerepe a "főnök", a másodiké a "beosztott". Ezeknek a szerepeknek a megadására is természetes mód van a séma megadásakor.

Lehetőségünk van arra is, hogy bizonyos kapcsolatokat rekurzivan adjunk meg, természetesen ennek csak akkor van értelme, ha a megfelelő entity tipusok azonosak.Az előző példánál maradva elég csak csak a közvetlen főnök-beosztott kapcsolatokat megadni - például osztályvezető-munkatárs, főosztályvezető-osztályvezető, igazgató-főosztályvezető -, és a rendszer ebből rekurzivan felépiti a teljes főnök-beosztott kapcsolatot.

Egy olyan attributomot, vagy attributomok egy olyan halmazát, amelyek értéke egyértelműen meghatározza a hozzájuk tartozó entity-t az entity halmazon belül kulcsnak nevezünk. Jogunk van bármilyen entity halmazhoz egy kulcsot definiálni (például egy személyi nyilvántartásban természetesen kinálkozik a személyi szám, mint kulcs), azonban nem követeljük meg, hogy legyen kulcs

minden entity halmazhoz, sőt az is előfordulhat, hogy van egy olyan entity halmazunk, amelyben két különböző entity minden attributuma egyenlő. Természetesen a modellen belül valahogyan meg kell különböztetni ezt a két elemet, de előfordulhat, hogy ez csak a különböző kapcsolataikon keresztül lehetséges [TL'82,8.fejezet] Az ott szereplő példa egy orvosi nyilvántartás, amelyben szerepelnek a diagnózishoz szükséges rutinvizsgálatok leirásai, például egy vércukorteszt végrehajtásának módja. Ezt az entity halmazt sok entity alkotja - nevezetesen az egyes betegeken végrehajtott konkrét tesztek -, de ezek tulajdonságai mind azonosak, csak a mondjuk "beteg" entity-kkel való kapcsolatuk más.)

Ezen modellen belül lehetőségünk van még az úgynevezett gyenge entity-halmazok létrehozására, amelyekbe tartozó entity-k csak akkor létezhetnek, ha egy megadott kapcsolaton keresztül létezik rájuk hivatkozás. (A személyi szám entity halmazba csak azok a számok tartoznak, amelyekre történik hivatkozás a személy entity halmazból.)

Az entity-relationship modell sémáját a következő szemléletes módon tudjuk összegezni, és megjeleniteni :

A sémát egy ábrába rajzoljuk fel, amelyet entity-relationship diagramnak nevezünk.

Ebben az ábrában a téglalapok entity halmazokat jelképeznek, és a téglalapra ráirjuk az illető entity halmaz nevét.

Az attributumok érték-halmazait körként rajzoljuk le, és egy éllel összekötjük azon entity halmazokkal, amelyek-

ben ez az attributum szerepel.

A kapcsolatokat álló rombuszként jelenitjük meg, amelyet éllel kötünk össze a relációt alkotó entity halmazokkal, amely élekre ráirhatjuk az entity esetleges szerepét a kapcsolaton belül. Arra is van mód, hogy az élen számmal jelezzük az adott reláción belül megengedett maximális különböző értékek halmazát. Igy például 1 jelöli az egyértelmű komponenst, N jelöli azt, hogy nincs megszoritás.

Az entity-relationship modell egy igen általános módot ad a modell elkészitésére, és nagy népszerűsége annak köszönhető, hogy általánosan elfogadott, hogy a fenti entity fogalom, és a közöttük fennálló kapcsolatok természetes modellezési eszközök. Ezen modell legnagyobb hiányossága a dinamikus megszoritások - azaz a műveletek támogatásának - hiánya, másrészt nincsenek a kapcsolatok közötti viszony leirására sem eszközök. A modell továbbfejlesztésében [például SH'78] már kinálnak ezekre a problémákra megoldást.

2.3.1.2. A strukturális modell

Ez a modell [WE'80] a relációs adatmodell továbbfejlesztése, és felépitésében sok vonást átvett az entity-relationship modellből.

Ezen modell megalkotásásban alapvető szempont volt az, hogy a relációs adatmodell azon szabadsága, amivel a az adatobjektumokat és a közöttük fennálló kapcsolatokat ugyanazzal a formalizmussal (az n-változós relációkra

épülő halmazelmélettel és kalkulussal) kezelhetjük inkább káros, mint előnyös. Igy ebben a modellben feltételezzük az attributumok értékhalmazainak létét, és az objektumokat az ezekből alkotott relációk jelentik. Az objektumok közötti kapcsolatot egy egészen más szó, a kapcsolat (connection) jelöl, amely szintén egy matematikai reláció, de a formalizmus segiti a világos megkülönböztetést.

A modellben pontos előirásokat találunk arra, hogy az attributumok milyen relációi jelölhetnek adatobjektumot, nevezetesen öt tipusa van az ilyen relációknak. Például lehetőség van arra, hogy egy objektum tipuson belül az adatobjektumokat alkotó relációk paritása (változóinak száma) ne legyen egyenlő, ez teljesen új vonás. Például a gyerekek számától függetlenül egy elemmel az apja(apa, fiú1, fiú2,...,fiún)-nel le tudjuk irni az apa fiú kapcsolatokat. Lehetőség van arra is, hogy a nagyon fontos, kétváltozós egy-egyértelmű relációkat - amelyeket itt lexikonnak hivunk - külön kezeljük.

Az objektumok közötti kapcsolatok felépitésében is megvan ez a precizitás. Például, természetes mód van arra, hogy az attributumokból felépitett kulcs segitségével bizonyos esetekben egy kapcsolat tulajdonosát definiáljuk, ez a CODASYL megközelités megvalósitását segiti.

Ezek a részletes előirások segitik az adatbázis tervezését, és megadják a szükséges pontosságot a modell kezeléséhez.

2.3.1.3. Az objektum-szerep modell

Ez a modell [FA'76] elsősorban azért jelentős, mert bevezette a "szerep" fogalmát, amely azután más modellekben is nagy fontosságra tett szert.

Ez a modell a relációs adatmodell olyan továbbfejlesztése, amelyben a relációk csak az objektumok közötti kapcsolatok leírására szolgálnak.

Itt az objektumok maguk nem hordoznak információt, csupán annyit, hogy léteznek, és lehet rájuk egy egyértelmű névvel hivatkozni.

Ha egy objektumról információt szeretnénk kapni azon felül, hogy létezik, akkor meg kell adni egy "szerepet", és válaszul meg fogjuk kapni azokat a tulajdonságokat, amelyekkel ay adott nevű objektum ebben a "szerepben" bir. (Esetleg azt a választ kapjuk, hogy ennek az objektumnak nincs ilyen szerepe.)

Az előzőekből természetes módon következik, hogy a kapcsolatok megadásánál meg kell adnunk a komponensek szerepeit, csak igy értelmes a kapcsolatról beszélni.

A modellben a séma megadásakor az objektumok közötti "legkisebb" kapcsolatokat kell először megadni, azaz azokat, amelyek szemantikusan irreducibilisek (lásd 2.3.pontot) - ezekre a szerző egy külön elnevezést: "asszociáció" vezetett be -, és ezekből lehet felépiteni az általános kapcsolatokat.

A szerep fogalmának felismerése igen fontos, azzal, hogy

egy objektum több "szerepben" hordoz hasznos információt, csökkenthető a redundáns információ mennyiségét, és közvetlen probléma-orientált modellezésre nyilik lehetőség.

2.3.2. Matematikai modellek

Ezek a modellek általában a relációs adatmodell továbbfejlesztéséhez szükséges matematikai alapokat kivánják felépiteni. A relációs adatmodell azért jelenti a kiindulópontot, mert ennek elmélete (az úgynevezett relációkalkulus) egyszerű, de kifejező, és matematikai értelemben is preciz.

Ezen modellek általában nem jelentenek igazából új modellt, mivel az esetek többségében a már létező, de nem szabatos fogalmakat kisérelik meg a modellek felépitéséhez szükséges matematikai logikába beépiteni.

Ilyen például az előzőekben emlitett, nem elsőrendű, hanem magasabb rendű nyelvek elmélete, vagy a "null", és "nem ismert" objektumok szerepe a predikátum kalkulusban.[RE'84]

A szükséges matematikai eszközök megitélésében eltérések vannak az egyes szerzők között, azaz az elmélet még nem kristályosodott ki. A kiindulópont természetesen az elsőrendű matematikai logika, és az ahhoz kapcsolódó halmazelmélet, és ezt kell kiterjeszteni a modellek kezeléséhez szükséges matematikai formalizmussal.

Összefoglalva, ezen modelleknek közvetlem gyakorlati

jelentőségük nincs, hanem az elvi alapok felépitéséhez kinálnak egy szilárd alapot.

2.3.3. Irreducibilis modellek

Ezen modellek fő motivációja az, hogy a sémát bizonyos alapvető, egyszerű adatokból kiindulva épithessük fel. Itt tehát az a cél, hogy felismerjük az információ "atomjait", azaz azokat az információ darabokat, amelyek egy elemi fogalmat jelölnek, és igy az információ megőrzésével tovább már nem bonthatók. Ezekből az "atomokból" aztán felépithetjük az összetett fogalmakat is. Ezen modellek szerzőinek meggyőződése az, hogy jól megválasztott atomokkal elérhető az, hogy az atomok függetlenek legyenek, azaz, amikor néhány atomból összeállitunk egy összetett objektumot, akkor az alkotórészeket külön-külön, egymástól teljesen függetlenül megváltoztatva az összetett objektum értelmes marad. Ennek a feltevésnek a jogosultsága, valamint az elemi információk megkeresésének támogatása még további kutatást igényel. Valószinű, hogy a teljes függetlenséget nem tudjuk elérni [BRO'84], de bizonyos korlátok között igen, és ez nagymértékben megnövelheti a modellezési precizitást.

2.3.3.1. A bináris-reláció modell

A bináris-reláció modell [BPP'76] egy tipikus irreducibilis adatmodell, ez a relációs adatmodellnek egy olyan speciális változata, amelyben csak kétváltozós relációkat engedünk meg . Ezek a kétváltozós relációk a legkisebb ábrázolható információ egységek a rendszerben, és

ezek általában egy objektum és egy egyszerű attributum-érték közötti kapcsolatot irják le, és ezeket tekintjük atomi tényeknek. Ezen modell legnagyobb előnye a kevés és világos felépitésű adatkonstrukció, hátránya, hogy az általános kapcsolatok ábrázolásához mesterséges "összekötő" tipusokat kell definiálnunk.

2.3.3.2. Az irreducibilis reláció modell

Az irreducibilis reláció modell [HOT'76] a relációs adatmodell olyan változata, ahol a relációk lehetnek n-változósak, és azokat a relációkat tekintjük atomi tényeknek, amelyek szemantikusan irreducibilisek. Ez azt jelenti, hogy nem lehet őket két vagy több kisebb reláció egyesitésére bontani, hogy közben információ el ne vesszék, azaz, ha a felbontás elemeit kombináljuk, akkor nem kapjuk vissza az eredeti relációt. Például, ha van egy leltárt számontartó rendszerünk, és abban van egy mennyiség(raktárterem,árufélesség,darabszám), ezt valószinűleg nem lehet felbontani valami R1(raktárterem,árufélesség) és R2(árufélesség,darabszám) egyesitésére, mivel egy áru a különböző raktártermekben különböző mennyiségben fordul elő.

2.3.3.3. A funkcionális adatmodell

Ez a modell a legjobban kidolgozott irreducibilis adatmodell, és ez sikeresen épiti be a funkcionális programozás elméletét az adatmodellezésbe.

A funkcionális adatmodellek alkotóelemei az entity-k, és

az ezek közötti függvények. Igy az egyes objektumok tulajdonságainak leirására a függvények szolgálnak, és ezen modellek legvonzóbb tulajdonsága az, hogy természetes módon, a függvények kompoziciójával tudunk új, származtatott adatokat létrehozni, és ezen konstrukciókkal egyszerű lekérdező müveletekre is lehetőség nyilik.

Ezen modellek közül kiválasztottunk egyet, nevezetesen a DAPLEX adatmodellt [SH'81], és kissé részletesebben is megvizsgáljuk.

Először is a DAPLEX nem csupán egy adatmodell, hanem kinálja azt a felhasználói interface-t (fogalmi leiró nyelvet) is, ami a természetes adatbázis kezeléshez szükséges.

Az előzőeknek megfelelően a DAPLEX-ben két alapvető konstrukció van, az egyik az entity, a másik a függvény. Az előző az adatobjektumok, az utóbbi az objektumok tulajdonságainak megjelenitésére szolgál.

Itt a függvények a lehető legáltalánosabbak lehetnek, azaz lehetnek többváltozósak, és lehetnek több értékűek is. Igy egy érdekes formai egyszerűsitésre nyilik lehetőség, mivel vannak változó nélküli,többértékű függvények is (ezen függvények tulajdonképpen entity-tipusokat jelölnek), igy nincs szükség külön tipusokat definiálni. (Feltéve, hogy már kezdetben is van egy nagy halmazunk - mondjuk ENTITY -, amely tartalmazza az összes objektumot, és igy használhatjuk, mint általános értelmezési tartomány. A továbbiakban ezt mindig feltételezzük, továbbá azt is, hogy a szokásos programozási nyelvekből megszokott "EGéSZ","STRING","BOOLE" tipusok is garantál-

tak.)

A DAPLEX-ben két sémaépitő művelet van a DECLARE, és a DEFINE. Az első új függvények létrehozására szolgál, a második segitségével pedig a már létező függvényekből új, származtatott függvényeket állithatunk elő. Az adatmanipulációs, és lekérdező műveletek nem tesznek különbséget a két fajta függvény között, igy ügyesen megválasztott származtatott függvényekkel támogathatjuk a különböző felhasználói nézőpontok létrehozását. A pontos szintaxis ismertetése túl sok helyet foglalna, igy inkább egy egyszerű példát mutatunk.

A példában egyszerű nyilvántartásról van szó, amelyben tároljuk néhány programozó személyi adatait, továbbá azt, hogy aktuálisan milyen projekteken dolgoznak. A projekteknek van egy egyértelmű vezetője.

Először definiáljuk a személyi adatokat. Ezt úgy tehetjük meg, hogy egy változó nélküli, többértékű függvénnyel létrehozzuk a "személy" entity tipust, azután definiáljuk a személyek nyilvántartásba veendő adatait. Tehát :

```
DECLARE személy() ==>> ENTITY

DECLARE név(személy) => STRING
DECLARE cim(személy) => STRING
stb.
```

Itt a kettős nyil többértékű, az egyes nyil egyértékű függvényt jelez.

Ezután úgy tudjuk jelezni, hogy a programozókra kérünk személyi nyilvántartást, hogy ezen tipust igy hozzuk létre :

```
DECLARE programozó() =>> személy
```

Látható, hogy igy egyszerűen meg tudtunk valósitani egy specializációt.

Most pedig hozzuk létre a projekt tipust, és tulajdonságait :

```
DECLARE project() =>> ENTITY

DECLARE vezető(project) => programozó

DECLARE feladat(programozó) => project
```

Tekintsük most a feladat és a vezető függvények kompozicióját. Látható, hogy ennek a függvénynek teljesen világos szemantikai jelentése van, ez irja le a főnök-besztott kapcsolatot, igy:

```
DEFINE főnök(programozó) =>> vezető(feladat(programozó))
```

A származtatott függvények felépitésénél még sok egyéb lehetőségünk is van a kompozició függvény képzésén kivül, például használhatjuk az inverz-függvényt, vagy több függvényből, mint komponensből többdimenziós függvényt képezhetünk.

Még ki kéne térnünk a manipulációs, és lekérdező művelétek szintaxisára, de az tulságosan messze vezetne, és

reméljük, hogy a fenn vázoltak elegendők egy gyors betekintésbe a DAPLEX rendszerbe, és igy a funkcionális adatmodellbe is.

2.3.4 A szemantikus hálózati modellek

A szemantikus adatmodellek többsége ebbe a csoportba tartozik, és nehéz részletes áttekintést adni az egyes konkrét modellekről. Mindenesetre ezen modellek a szemantikus hálók fogalmára épülnek (amelyek olyan gráfokat jelentenek, ahol az élek megvannak cimkézve és jelentéssel birnak), és lényegüket az "osztály" fogalmára épülő absztrakciós mechanizmusok jelölik. Igy a 2.1 pont alapján elég világos képünk van ezen modellek szerkezetéről.

A továbbiakban a legjobban kidolgozott modell, a TAXIS lényegét ismertetjük. Egyéb szemantikus modellek az RM/T [CO'79]. SDM [HM'81]. SHM+ [B-R'84].

A TAXIS modell az előzőekben emlitett kétszintes hierarchiára épül. Igy az alkotórészei a következők:

- az adatobjektumok, amelyeket itt "token"-nek nevezünk,
- a tokenekből álló osztályok,
- a metaosztályok, amelyeknek elemei osztályok.

Mindhárom objektum fajtára név szerint lehet hivatkozni, és az osztályok és metaosztályok formális leirással birnak, amely meghatározza az elemei tulajdonságainak fajtáját. (Ezt a szerzők úgy fejezik ki, hogy a tulajdonságok definiciója az osztály leirásban "indukálja" az

egyes elemek "tényleges" tulajdonságait.)

Egy osztály, illetve metaosztály definiciójában jelezhetjük, hogy ez az osztály "változó", ez azt jelenti, hogy az ilyen osztályokba beilleszthetünk egy új tokent (ill. osztályt), illetve törölhetünk belőle egy elemet.

Az osztályok és metaosztályok külön-külön egy IS-A hierarchiába vannak szervezve, amelyben a lehető legáltalánosabb specializáció művelettel hozhatunk létre új osztályokat, definiálhatunk új attributumokat, régieket pótolhatunk egy erősebb definicióval, és a speciálisabb osztály kijelölése egy tetszőleges logikai állitással történhet (ez az un. test-defined osztály).

Ez a formalizmus annyira erős, hogy a modellbeli műveleteket is természetes módon egy IS-A hierarchiába tudjuk szervezni. A műveleteknél lehetőség van a szokásos előfeltétel megadására.

A TAXIS rendszeren belül a szigorú öröklődés áll fenn, azaz a speciális osztály minden eleme egyben eleme lesz az általános osztálynak is. Tudjuk, hogy ez problémákat jelent, ennek megoldása a rendszeren belül egy külön kivétel-kezelő egység felépitése. Ehhez szükséges az, hogy világos tesztekkel magukat a kivételeket is egy IS-A hierarchiába szervezzük, és igy, amikor egy kivétel felmerül (pl. egy tranzakciót kisérelünk meg végrehajtani, de az előfeltétel nem teljesül), akkor az aktivizálódó kivétel-kezelő ezek alapján megtehesse a szükséges lépéseket.

IRODALOM

AHU'83

Aho, A.V., Hopcroft, J.E., Ullman, J.D.
Data Structures and algorithms.
Addison-Wesley, Reading, Mass., 1983.

BFL'83

Brachman, R.J., Fikes, R.E., Levesque, H.J.
Krypton: A Functional Approach to Knowledge Representation.
Computer, Vol.16, No.10, October 1983, pp.67-73.

BL'68

Black, F.
A Deductive Quesion Answering System
In: Minsky, M./ed./ Semantic Information Processing
MIT Press, Cambridge, Mass.,1968, pp.354-402.

BPP'76

Bracchi, G., Paolini, P., Pelagatti, G.
Binary Logical Associations in Data Modelling. In: G. M. Nijsen /ed./ Modelling in Database Management Systems
North Holland, 1976.

BR'79

Brachman, R.J.
On the Epistemological Status of Semantic Networks.
In: Findler, N.V. /ed./ Associative Networks: Representation and Use of Knowledge by Computers
Academic Press, New York, 1979, pp.3-50.

BR'83
Brachman, R. J.
What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks
Computer, Vol.16, No.10, October 1983, pp.30-36.

BRO'84
Brodie, M.L.
On the Development of Data Models. In: Brodie, M.L., Mylopoulos, J., Schmidt, J.W. /eds./ On Conceptual Modelling
Springer Verlag, 1984.

B-R'84
Brodie, M.L., Ridjanovic, D.
On the Design and Specification of Database Transactions. In: Brodie, M.L., Mylopoulos, J., Schmidt, J.W. /eds./ On Conceptual Modelling
Springer Verlag, 1984.

BS'84
Brachman, R.J., Schmolze, J.
An Overview of the KL-One Knowledge Representation System Cognitive Science /in press/.

BWE'80
Bobrow, R.J., Webber, B.L.
Knowledge Representation for Syntactic/Semantic Processing
Proc.First Ann. Nat'l Conf. Artificial Intelligence, American Association for Artificial Intelligence, 1980, pp. 316-323.

BWI'76
Bobrow, D.G., Winograd, T.
An Overview of KRL: A Knowledge Representaion Language
Techn.Report CSL-76-4, Xerox Palo Alto Res.Center, Palo Alto, Cal., 1976.

CH'76
Chen, P.P.
The Entity-Relationship Model
ACM Trans. on Database Systems, Vol.1, No.1, March 1976.

CO'79
Codd, E.F.
Extending the Database Relational Model to Capture More Meaning
ACM Trans. on Database Systems, Vol.4, No.4, December 1979, pp.397-434.

CR'66
Craig, J.A. et al.
DEACON: Direct English Access and Control
AFIPS Conf. Proc.,Vol.29, 1966 FJCC, pp.365-380.

DMN'68
Dahl, O.J., Myrhang, B., Nygaard, K.
SIMULA '67 Common Base Language
Pub.S-22, Norwegian Computing Center, Oslo, 1968.

FA'79
Fahlam, S.E.
A System for Representing and Using Real-World Knowledge. In: Winston, P.H., Brown, R.H./eds./ AI: An MIT Perspective, Vol.1.
MIT Press, Cambridge, Mass., 1979, pp.453-470.

FI'79
Findler, N.V. /ed./
Associative Networks: Representation and Use of Knowledge by Computers
Academic Press, New York, 1979.

FU'76
Funt, B.V.
WHISPER: A Computer Implementation Using Analogoues in Reasoning
Techn.Report TR-76-9, Dept.of Computer Science, Univ. of British Columbia, Cancouver, BC. 1976.

GO' 79
Goodwin, J.W.
Taxonomic Programming with KL-One
Techn. Report LITH-MAT-R-79-5, Informatics Laboratory, Linkoeping University, Sweden, 1979.

HA'77
Hayes, P.J.
In Defence of Logic
Proc. 1977 Int'l Joint Conf. Artificial Intelligence, pp.559-565.

HA'79
Hayes, P.J.
The Logic of Frames. In: Metzing, D./ed./ Frame Conceptions and Text Understanding.
Walter de Gruyter and Co., Berlin, 1979, pp.46-61.

HA'84
Hayes, P.J.
On Semantic Nets, Frames and Associations
Natural Language, Vol.5, pp.99-107.

HEN'79
Hendrix, G.G.
Encoding Knowledge in Partitioned Networks. In: Findler, N.V. /ed./ Associative Networks: Representation and Use of Knowledge by Computers.
Academic Press, New York, 1979, pp.51-92.

HEW'72
Hewitt, C.
Description and Theoretical Analysis (Using Schemata) of Planner
MIT AI Memo 251, Cambridge, Mass., April 1972

HM'81
Hammer, M., McLead, D.
Database Description with SDM: A Semantic Database Model
ACM Trans. on Database Systems, Vol.6, No.3, September 1981, pp.351-386.

HOT'76
Hall, P., Owlett, J., Todd, S.J.
Relations and Entities. In: Nijssen, G.M./ed./
Modelling in Database Management Systems
North Holland, 1976.

IB'81
Israel, D.J., Brachman, R.J.
Disctinction and Confusions: A Catalogue Raisonne
Proc. 1981 Int'l Joint Conf. Artificial Intelligence, pp. 452-459.

IS'83
Israel, D.J.
The Role of Logic in Knowledge Representation
Computer, Vol.16, No.10, October 1983, pp.37-41.

KS'78
Klahr, D., Siegler, R.
The Representation of Children's Knowledge. In:
Reese, H.W., Lipsett, L.P./eds./ Advance in Child
Development, Vol.12.
Academic Press, New York, 1978.

KO'82
Kolata, G.
How Can Computers Get Common Sense
Science, Vol.217, September 24, 1982, pp.1237-1238.

LA'82
Langley, P.
Strategy Acquisition Governed by Experimentation
Proc. European Conf. Artificial Intelligence, 1982,

pp. 171-176.

LA'83
Langley, P.
Representational Issues in Learning Systems
Computer, Vol.16, No.10, October 1983, pp.47-51.

LBS'83
Langley, P., Bradshaw, G., Simon, H.A.
Rediscovering Chemistry with the Bacon System. In:
Michalski, R., Carbonell, J.G., Mitchell, T.M./eds./
Machine Learning: An Artificial Intelligence
Approach
Tioga Press, Palo Alto, Cal., 1983.

LE'82
Levesque, H.J.
A Formal Treatment of Incomplete Knowledge Bases
Techn.Report 3, Fairchild Laboratory for Artificial
Intelligence Research, Palo Alto, Cal., February
1982.

LE'84
Levesque, H.J.
The Logic of Incomplete Knowledge Bases. In: Brodie,
M.L., Mylopoulos, J., Schmidt, J.W./eds./ On Concep-
tual Modelling
Springer Verlag, 1984.

MBW'80
Mylopoulos, J., Bernstein, P.A., Wong, H.K.
A Language Facility for Designing Database-Intensive
Application

ACM Trans.on Database Systems, Vol.5, No.2, June 1980, pp.185-207.

MCC'83
McCalla, G., Cercone, N.
Guest Editors' Introduction: Approaches to Knowledge Representation
Computer, Vol.16, No.10, October 1983, pp.12-18.

MCA'80
McCarthy, J.
Circumscription - A Form of Non-Monotonic Reasoning
Artificial Intelligence, Vol.13, No.1-2, April 1980, pp.27-39.

MIN'75
Minsky, M.
A Framework for Representing Knowledge. In: Winston, P./ed./ The Psychology of Computer Vision.
McGraw Hill, 1975.

MIN'82
Minsky, M.
Why People Think Computers Can't
AI Magazine, Vol.3, No.4, 1982. pp.3-15.

MIT'81
Mitchell, T.M. et al.
Learning Problem Solving Heuristics Through Practice
Proc. Seventh Int'l Joint Conf. Aritificial Intelligence, 1981, pp.127-134.

MST '83
Mylopoulos, J., Shibahara, T., Tsotsos, J.K.
Building Knowledge-Based Systems: The PSN Experience
Computer, Vol.16, No.10, October 1983, pp.83-89.

NA'83
Nau, D.
Expert Computer Systems
Computer, Vol.16, No.2, February 1983, pp.63-84.

NS'72
Newell, A., Simon, H.A.
Human Problem Solving
Prentice-Hall, Englewood Cliffs, N.J., 1972.

NIJ'81
Nijssen, G.M.
An Architecture for Knowledge Base Systems
Proc.of the SPOT-2 Conference, Stockholm, September 1981.

NIL'82
Nilsson, N.
Artificial Intelligence: Engineering, Science or Slogan?
AI Magazine, Vol.3, No.1, 1982, pp.2-8.

PS'81
Papalaskaris, M.A., Schubert, L.K.
Parts Inference in Closed and Semi-Closed Partitioning Graphs
Seventh Int'l Joint Conf. AI, 1981, pp.304-309.

PAP'82
Papalaskaris, M.A.
Special-Purpose Inference Methods
Masters Thesis, Dept. of Computing Science University of Alberta, Edmonton, 1982.

PAS'82
Patel-Schneider, P. et al.
PSN: An Extensible Knowledge Representation Scheme
Proc. Canadian Soc. Computational Studies Intelligence Conf., 1982.

QU'68
Quillian, M.R.
Semantic Memory. In: Minsky, M./ed./ Semantic Information Processing. Cambridge, Mass., 1968, pp.216-270.

RA'68
Raphael, B.
SIR: Semantic Informantion Retrieval. In: Minsky, M./ed./ Semantic Information Processing. Cambridge, Mass. 1968, pp.33-134.

RE'78
Reiter, R.
On Closed World Data Bases. In: Gallaire, H., Minkler, J./eds./ Logic and Data Base.
Plenum Press, New York, 1978.

RE'84
Reiter, R.
Towards a Logical Reconstruction of Relational Database Theory. In: Brodie,M.L., Mylopoulos, J., Schmidt, J.W./eds./ On Conceptual Modelling
Springer Verlag, 1984.

RO'81
Robson, D.
Object-Oriented Software Systems
Byte, Vol.6, No.8, August 1981, pp.74-86.

RU'86
Ruttkay, Zs.
Tudáskezelő software eszközök. - Döntéselőkészitő tanulmány software eszközök vásárlásához.
MTA SZTAKI Intelligens Rendszerek Osztály, Budapest, 1986. május.

SL'83
Schmolze, J.G., Lipkis, T.A.
Classification in the KL-One Knowledge Representation System
Proc. Int'l Joint Conf. Artificial Intelligence, 1983.

SGC'79
Schubert, L.K., Goebel, R.G., Cercone, N.J.
The Structure Organization of a Semantic Net for Comprehension and Inference. In: Findler, N.V./ed./ Associative Networks: Representation and Use of Knowledge by Computers.
Academic Press, New York, 1979.

SCU'79
Schubert, L.K.
Problems with Parts
Proc. Sixth Int'l Joint Conf. AI, Tokyo, 1979, pp.778-784.

SH'78
Shoshani, A.
CABLE: A Language Based on the Entity-Relationship Model
Lawrence Berkeley Lab., Berkeley, Cal.,1976.

SHI'81
Shipman, D.W.
The Functional Data Model and the Data Language DAPLEX
ACT Trans. on Database Systems, Vol.6, No.1, March 1981, pp.140-173.

SPT'83
Schubert, L.K., Papalaskaris, M.A., Taugher, J.
Determining Type, Part, Colour and Time Relationships
Computer, Vol.16, No.10, October 1983, pp.53-60.

ST'85
Stachowitz, R.A.
A Formal Framework for Describing and Classifying Semantic Data Models
Inf. Systems, Vol.10, No.1, 1985, pp.77-96.

SW'71
Shapiro, S.C., Woodmansee, G.H.
A Net Structure Based Relational Question-Answerer
Proc. Int'l. Joint Conf. AI, Washington, DC, 1971, pp.325-346.

TL'82
Tsichritzis, D.C., Lochovsky, F.H.
Data Models
Prentice Hall, 1982.

TS'80
Tsotsos, J.
A Framework for Visual Motion Understanding
PhD Thesis, Dept.of Computer Science, University of Toronto, 1980.

UL'82
Ullman, J.D.
Principles of Database Systems
Computer Science Press, 1982.

WE'80
Wiederhold, G., El-Masri, R.
The Structural Model for Database Design. In: Chen, P.P./ed./ Entity-Relationship Approach to Systems Analysis and Design
North Holland, Los Angeles, 1979.

WHR'78
Waterman, D.A., Hayes-Roth, F.
An Overview of Pattern-Directed Inference Systems
In: Waterman, D.A., Hayes-Roth, F./eds./ Pattern-

Directed Inference Systems
Academic Press, New York, 1978, pp.3-22.

WI'71
Wirth, N.
Program Development by Stepwise Refinement
Communications of the ACM, Vol.14, No.4, April 1971, pp.221-227.

WIN'75
Winston, P.
Learning Structural Descriptions from Examples
In: Winston, P./ed./ The Psychology of Computer Vision
McGraw Hill, New York, 1975.

WO'75
Woods, W.A.
What's in a Link: Foundations for Semantic Networks
In: Bobrow, D.G., Collins, A.M./eds./ Representation and Understanding
Academic Press, New York, 1975, pp.35-82.

WO'80
Woods, W.A.
Cascaded ATN Grammers
Am. J. Computational Linguistica, Vol.6, No.1, Jan.-March, 1980, pp.1-15.

WO'83
Woods, W.A.
What's Important About Knowledge Representation?
Computer, Vol.16, No.10, October 1983, pp.22-27.

YO'76
Young, R.M.
Seriation by Children: An Artificial Intelligence Analysis of a Piagetian Task
Birkhauser, Basel, 1976.

YOS'81
Young, R.M., O'Shea, T.
Errors in Children's Subtraction
Cognitive Science, Vol.5, 1981. pp.153-177.

1986-BAN MEGJELENTEK:

179/1986 Terlaky Tamás: Egy véges criss-cross módszer és alkalmazásai

180/1986 K.N. Čimev: Separable sets of arguments of functions

181/1986 Renner Gábor: Kör approximációja a számitógépes geometriai tervezésben

182/1986 Proceedings of the Joint Bulgarian-Hungarian Workshop on "Mathematical Cybernetics and Data Processing" Scientific Station of Sofia University, Giulecica /Bulgaria/, May 6-10, 1985 /Editors: J. Denev, B. Uhrin/ Vol I

183/1986 Proceedings of the Joint Bulgarian-Hungarian Workshop on "Mathematical Cybernetics and Data Processing" Scientific Station of Sofia University, Giulečica /Bulgaria/, May 6-10, 1985 /Editors: J. Denev, B. Uhrin/ Vol II

184/1986 HO THUAN: Contribution to the theory of relational databases

185/1986 Proceedings of the 4th International Meeting of Young Computer Scientists IMICS'86 /Smolenice, 1986/ /Editors: J. Demetrovics, J. Kelemen/

186/1986 PUBLIKÁCIÓK - PUBLICATIONS 1985 Szerkesztette: Petróczy Judit

187/1986 Proceedings of the Winter School on Conceptual modelling /Visegrád, 27-30 January, 1986/ /Editors: E. Knuth, A. Márkus/

188/1986 Lengyel Tamás: A Cluster analizis néhány kombinatorikai és valószinüségszámitási problémája

189/1986 Bernus Péter: Gyártórendszerek funkcionális analizise és szintézise

190/1986 Hernádi Ágnes: A tipus fogalma, és szerepe a modellezésben

191/1986 VU DUC THI: Funkcionális függőséggel kapcsolatos néhány kombinatorikai jellegü vizsgálat a relációs adatmodellben

192/1986 Márkusz Zsuzsanna: P a p e r s on Many-stored logic as a tool for modelling

193/1986 KNVVT Conference on Automation of Information Processing on Personal Computers
Budapest, May 5-9, 1986 Vol I.
Szerkesztette: Ratkó István

194/1986 KNVVT Conference on Automation of Information Processing on Personal Computers
Budapest, May 5-9, 1986 Vol II.
Szerkesztette: Ratkó István